STATA MULTILEVEL MIXED-EFFECTS REFERENCE MANUAL
RELEASE 19



A Stata Press Publication
StataCorp LLC
College Station, Texas



Copyright © 1985–2025 StataCorp LLC
All rights reserved
Version 19

Published by Stata Press, 4905 Lakeway Drive, College Station, Texas 77845

ISBN-10: 1-59718-435-7
ISBN-13: 978-1-59718-435-9

This manual is protected by copyright. All rights are reserved. No part of this manual may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means—electronic, mechanical, photocopy, recording, or otherwise—without the prior written permission of StataCorp LLC unless permitted subject to the terms and conditions of a license granted to you by StataCorp LLC to use the software and documentation. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document.

StataCorp provides this manual "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. StataCorp may make improvements and/or changes in the product(s) and the program(s) described in this manual at any time and without notice.

The software described in this manual is furnished under a license agreement or nondisclosure agreement. The software may be copied only in accordance with the terms of the agreement. It is against the law to copy the software onto DVD, CD, disk, diskette, tape, or any other medium for any purpose other than backup or archival purposes.

The automobile dataset appearing on the accompanying media is Copyright © 1979 by Consumers Union of U.S., Inc., Yonkers, NY 10703-1057 and is reproduced by permission from CONSUMER REPORTS, April 1979.

Stata, STATA, Stata Press, Mata, MATA, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC.

Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations.

StataNow is a trademark of StataCorp LLC.

Other brand and product names are registered trademarks or trademarks of their respective companies.

For copyright information about the software, type `help copyright` within Stata.

The suggested citation for this software is

> StataCorp. 2025. *Stata 19*. Statistical software. StataCorp LLC.

The suggested citation for this manual is

> StataCorp. 2025. *Stata 19 Multilevel Mixed-Effects Reference Manual*. College Station, TX: Stata Press.

Contents

me . . . . . Introduction to multilevel mixed-effects models 1

estat df . . . . . Calculate degrees of freedom for fixed effects 49
estat group . . . . . Summarize the composition of the nested groups 53
estat icc . . . . . Estimate intraclass correlations 54
estat recovariance . . . . . Display estimated random-effects covariance matrices 57
estat sd . . . . . Display variance components as standard deviations and correlations 59
estat wcorrelation . . . . . Display within-cluster correlations and standard deviations 61

mecloglog . . . . . Multilevel mixed-effects complementary log–log regression 67
mecloglog postestimation . . . . . Postestimation tools for mecloglog 80

meglm . . . . . Multilevel mixed-effects generalized linear models 86
meglm postestimation . . . . . Postestimation tools for meglm 118

meintreg . . . . . Multilevel mixed-effects interval regression 136
meintreg postestimation . . . . . Postestimation tools for meintreg 148

melogit . . . . . Multilevel mixed-effects logistic regression 157
melogit postestimation . . . . . Postestimation tools for melogit 181

menbreg . . . . . Multilevel mixed-effects negative binomial regression 193
menbreg postestimation . . . . . Postestimation tools for menbreg 210

menl . . . . . Nonlinear mixed-effects regression 215
menl postestimation . . . . . Postestimation tools for menl 320

meologit . . . . . Multilevel mixed-effects ordered logistic regression 333
meologit postestimation . . . . . Postestimation tools for meologit 346

meoprobit . . . . . Multilevel mixed-effects ordered probit regression 353
meoprobit postestimation . . . . . Postestimation tools for meoprobit 367

mepoisson . . . . . Multilevel mixed-effects Poisson regression 374
mepoisson postestimation . . . . . Postestimation tools for mepoisson 391

meprobit . . . . . Multilevel mixed-effects probit regression 398
meprobit postestimation . . . . . Postestimation tools for meprobit 412

mestreg . . . . . Multilevel mixed-effects parametric survival models 418
mestreg postestimation . . . . . Postestimation tools for mestreg 443

metobit . . . . . Multilevel mixed-effects tobit regression 455
metobit postestimation . . . . . Postestimation tools for metobit 466

mixed . . . . . Multilevel mixed-effects linear regression+ 474
mixed postestimation . . . . . Postestimation tools for mixed 554

Glossary 576

Subject and author index 582

Cross-referencing the documentation

When reading this manual, you will find references to other Stata manuals, for example, **[U] 27 Overview of Stata estimation commands**; [R] **regress**; and [D] **reshape**. The first example is a reference to chapter 27, *Overview of Stata estimation commands*, in the *User's Guide*; the second is a reference to the `regress` entry in the *Base Reference Manual*; and the third is a reference to the `reshape` entry in the *Data Management Reference Manual*.

All the manuals in the Stata Documentation have a shorthand notation:

[GSM]	*Getting Started with Stata for Mac*
[GSU]	*Getting Started with Stata for Unix*
[GSW]	*Getting Started with Stata for Windows*
[U]	*Stata User's Guide*
[R]	*Stata Base Reference Manual*
[ADAPT]	*Stata Adaptive Designs: Group Sequential Trials Reference Manual*
[BAYES]	*Stata Bayesian Analysis Reference Manual*
[BMA]	*Stata Bayesian Model Averaging Reference Manual*
[CAUSAL]	*Stata Causal Inference and Treatment-Effects Estimation Reference Manual*
[CM]	*Stata Choice Models Reference Manual*
[D]	*Stata Data Management Reference Manual*
[DSGE]	*Stata Dynamic Stochastic General Equilibrium Models Reference Manual*
[ERM]	*Stata Extended Regression Models Reference Manual*
[FIN]	*Stata Financial Statistics Reference Manual*
[FMM]	*Stata Finite Mixture Models Reference Manual*
[FN]	*Stata Functions Reference Manual*
[G]	*Stata Graphics Reference Manual*
[H2OML]	*Machine Learning in Stata Using H2O: Ensemble Decision Trees Reference Manual*
[IRT]	*Stata Item Response Theory Reference Manual*
[LASSO]	*Stata Lasso Reference Manual*
[XT]	*Stata Longitudinal-Data/Panel-Data Reference Manual*
[META]	*Stata Meta-Analysis Reference Manual*
[ME]	*Stata Multilevel Mixed-Effects Reference Manual*
[MI]	*Stata Multiple-Imputation Reference Manual*
[MV]	*Stata Multivariate Statistics Reference Manual*
[PSS]	*Stata Power, Precision, and Sample-Size Reference Manual*
[P]	*Stata Programming Reference Manual*
[RPT]	*Stata Reporting Reference Manual*
[SP]	*Stata Spatial Autoregressive Models Reference Manual*
[SEM]	*Stata Structural Equation Modeling Reference Manual*
[SVY]	*Stata Survey Data Reference Manual*
[ST]	*Stata Survival Analysis Reference Manual*
[TABLES]	*Stata Customizable Tables and Collected Results Reference Manual*
[TS]	*Stata Time-Series Reference Manual*
[I]	*Stata Index*
[M]	*Mata Reference Manual*

me — Introduction to multilevel mixed-effects models

Description Quick start Syntax Remarks and examples
Acknowledgments References Also see

Description

Mixed-effects models are characterized as containing both fixed effects and random effects. The fixed effects are analogous to standard regression coefficients and are estimated directly. The random effects are not directly estimated (although they may be obtained postestimation) but are summarized according to their estimated variances and covariances. Random effects may take the form of either random intercepts or random coefficients, and the grouping structure of the data may consist of multiple levels of nested groups. As such, mixed-effects models are also known in the literature as multilevel models and hierarchical models. Mixed-effects commands fit mixed-effects models for a variety of distributions of the response conditional on normally distributed random effects.

Mixed-effects linear regression

`mixed`	Multilevel mixed-effects linear regression

Mixed-effects generalized linear model

`meglm`	Multilevel mixed-effects generalized linear models

Mixed-effects censored regression

`metobit`	Multilevel mixed-effects tobit regression
`meintreg`	Multilevel mixed-effects interval regression

Mixed-effects binary regression

`melogit`	Multilevel mixed-effects logistic regression
`meprobit`	Multilevel mixed-effects probit regression
`mecloglog`	Multilevel mixed-effects complementary log–log regression

Mixed-effects ordinal regression

`meologit`	Multilevel mixed-effects ordered logistic regression
`meoprobit`	Multilevel mixed-effects ordered probit regression

Mixed-effects count-data regression

`mepoisson`	Multilevel mixed-effects Poisson regression
`menbreg`	Multilevel mixed-effects negative binomial regression

Mixed-effects multinomial regression

Although there is no `memlogit` command, multilevel mixed-effects multinomial logistic models can be fit using `gsem`; see [SEM] **Example 41g**.

Mixed-effects survival model

`mestreg`	Multilevel mixed-effects parametric survival models

Nonlinear mixed-effects regression

`menl`	Nonlinear mixed-effects regression

Postestimation tools specific to mixed-effects commands

`estat df`	Calculate and display degrees of freedom for fixed effects
`estat group`	Summarize the composition of the nested groups
`estat icc`	Estimate intraclass correlations
`estat recovariance`	Display the estimated random-effects covariance matrices
`estat sd`	Display variance components as standard deviations and correlations
`estat wcorrelation`	Display within-cluster correlations and standard deviations

Quick start

Linear mixed-effects models

Linear model of `y` on `x` with random intercepts by `id`

```
mixed y x || id:
```

Three-level linear model of `y` on `x` with random intercepts by `doctor` and `patient`

```
mixed y x || doctor: || patient:
```

Linear model of `y` on `x` with random intercepts and coefficients on `x` by `id`

```
mixed y x || id: x
```

Same model with covariance between the random slope and intercept

```
mixed y x || id: x, covariance(unstructured)
```

Linear model of `y` on `x` with crossed random effects for `id` and `week`

```
mixed y x || _all: R.id || _all: R.week
```

Same model specified to be more computationally efficient

```
mixed y x || _all: R.id || week:
```

Full factorial repeated-measures ANOVA of `y` on `a` and `b` with random effects by `field`

```
mixed y a##b || field:
```

Generalized linear mixed-effects models

Logistic model of y on x with random intercepts by id, reporting odds ratios

```
melogit y x || id: , or
```

Same model specified as a GLM

```
meglm y x || id:, family(bernoulli) link(logit)
```

Three-level ordered probit model of y on x with random intercepts by doctor and patient

```
meoprobit y x || doctor: || patient:
```

Nonlinear mixed-effects models

Nonlinear mixed-effects regression of y on x1 and x2 with parameters {b0}, {b1}, {b2}, and {b3} and random intercepts U0 by id

```
menl y = ({b0}+{b1}*x1+{U0[id]})/(1+exp(-(x2-{b2})/{b3}))
```

Same as above, but using the more efficient specification of the linear combination

```
menl y = ({lc: x1 U0[id]})/(1+exp(-(x2-{b2})/{b3}))
```

Same as above, but using define() to specify the linear combination

```
menl y = {lc:}/(1+exp(-(x2-{b2})/{b3})), define(lc: x1 U0[id])
```

Include a random slope on continuous variable x1 in the define() option, and allow correlation between random slopes U1 and intercepts U0

```
menl y = {lc:}/(1+exp(-(x2-{b2})/{b3})),     ///
  define(lc: x1 U0[id] c.x1#U1[id]) covariance(U0 U1, unstructured)
```

Specify a heteroskedastic within-subject error structure that varies as a power of predicted mean values _yhat

```
menl y = {lc:}/(1+exp(-(x2-{b2})/{b3})),     ///
  define(lc: x1 U0[id] c.x1#U1[id])      ///
  covariance(U0 U1, unstructured) resvariance(power _yhat)
```

Three-level nonlinear regression of y on x1 with random intercepts W0 and slopes W1 on continuous x1 by lev2 and with random intercepts S0 and slopes S1 on x1 by lev3, with lev2 nested within lev3, using unstructured covariance for W0 and W1 and exchangeable covariance for S0 and S1

```
menl y = {phi1:}+{b1}*cos({b2}*x1),        ///
  define(phi1: x1 W0[lev3] S0[lev3>lev2]    ///
       c.x1#(W1[lev3] S1[lev3>lev2]))   ///
  covariance(W0 W1, unstructured) covariance(S0 S1, exchangeable)
```

Syntax

Linear mixed-effects models

`mixed` *depvar fe_equation* [`||` *re_equation*] [`||` *re_equation* ...] [, *options*]

where the syntax of the fixed-effects equation, *fe_equation*, is

[*indepvars*] [*if*] [*in*] [*weight*] [, *fe_options*]

and the syntax of a random-effects equation, *re_equation*, is the same as below for a generalized linear mixed-effects model.

Generalized linear mixed-effects models

`me`*cmd* *depvar fe_equation* [`||` *re_equation*] [`||` *re_equation* ...] [, *options*]

where the syntax of the fixed-effects equation, *fe_equation*, is

[*indepvars*] [*if*] [*in*] [, *fe_options*]

and the syntax of a random-effects equation, *re_equation*, is one of the following:

for random coefficients and intercepts

levelvar `:` [*varlist*] [, *re_options*]

for random effects among the values of a factor variable in a crossed-effects model

levelvar `:` `R.`*varname*

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

Nonlinear mixed-effects models

`menl` *depvar* `=` <*menlexpr*> [*if*] [*in*] [, *options*]

<*menlexpr*> defines a nonlinear regression function as a substitutable expression that contains model parameters and random effects specified in braces `{}`, as in `exp({b}+{U[id]})`; see *Random-effects substitutable expressions* in [ME] **menl** for details.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Using mixed-effects commands
Mixed-effects models
  Linear mixed-effects models
  Generalized linear mixed-effects models
  Survival mixed-effects models
  Nonlinear mixed-effects models
  Alternative mixed-effects model specification
  Likelihood calculation
  Computation time and the Laplacian approximation
  Diagnosing convergence problems
  Distribution theory for likelihood-ratio test
Examples
  Two-level models
  Covariance structures
  Three-level models
  Crossed-effects models
  Nonlinear models

Introduction

Multilevel models have been used extensively in diverse fields, from the health and social sciences to econometrics. Mixed-effects models for binary outcomes have been used, for example, to analyze the effectiveness of toenail infection treatments (Lesaffre and Spiessens 2001) and to model union membership of young males (Vella and Verbeek 1998). Ordered outcomes have been studied by, for example, Tutz and Hennevogl (1996), who analyzed data on wine bitterness, and De Boeck and Wilson (2004), who studied verbal aggressiveness. For applications of mixed-effects models for count responses, see, for example, the study on police stops in New York City (Gelman and Hill 2007) and the analysis of the number of patents (Hall, Griliches, and Hausman 1986). Rabe-Hesketh and Skrondal (2022) provide more examples of linear and generalized linear mixed-effects models. Nonlinear mixed-effects (NLME) models are popular in, for example, population pharmacokinetics, bioassays, and studies of biological and agricultural growth processes.

For a comprehensive treatment of mixed-effects models, see, for example, Searle, Casella, and McCulloch (1992); Verbeke and Molenberghs (2000); Raudenbush and Bryk (2002); Hedeker and Gibbons (2006); McCulloch, Searle, and Neuhaus (2008); and Rabe-Hesketh and Skrondal (2022). For NLME models, see, for example, Davidian and Giltinan (1995); Vonesh and Chinchilli (1997); Demidenko (2013); Pinheiro and Bates (2000); and Davidian and Giltinan (2003).

Shayle R. Searle (1928–2013) was born in New Zealand. He obtained his PhD in animal breeding from Cornell University in 1958, with a minor in statistics. Prior to moving to New York, he worked as a research statistician for the New Zealand Dairy Board, which provided the data that he would analyze for his thesis. After completing his doctoral degree, he worked as a research associate and published several articles. He later returned to his post as a statistician in New Zealand, a position which would have a lasting influence on his career.

Through his analysis of dairy production data, Searle made advancements in estimation methods for unbalanced data and published a book on this topic. He later returned to Cornell University, teaching courses in matrix algebra, linear regression models, and estimation of variance components. Searle was one of the first few statisticians to use matrices in statistics, and he wrote a couple of books applying matrix algebra to economics and statistics. In 2001, he published a book on mixed models, which proved to be a significant contribution considering that not many statisticians were well acquainted with random effects in the 1950s. His contributions did not go unnoticed: he was awarded the Alexander von Humboldt US Senior Scientist Award and was elected a fellow of the Royal Statistical Society and of the American Statistical Association.

George Casella (1951–2012) was born in Bronx, New York. After obtaining a PhD in statistics from Purdue University, he went on to join the faculty at Rutgers University, and later Cornell University, where he taught for 19 years, and the University of Florida. He published on topics such as confidence estimation, Bayesian analysis, and empirical Bayes methods. In general, his work was motivated by applications to science, and in particular, his work on variable selection and clustering was motivated by genetics. Casella coauthored a book with Roger Berger that introduced many graduate students to mathematical statistics. He coauthored another book with Christian P. Robert on Monte Carlo methods. In addition to his own published work, Casella was an editor for three journals: *Statistical Science*, *Journal of the American Statistical Society*, and *Journal of the Royal Statistical Society*.

Casella's many contributions are reflected in his election to fellowship on behalf of four different associations and institutes and being made a foreign member of the Spanish Royal Academy of Sciences. He acquired the Spanish language during a year he spent in Spain for sabbatical and even gave talks on Monte Carlo methods in Spanish. Aside from his academic accomplishments, Casella completed 13 marathons and spent time as a volunteer firefighter.

Using mixed-effects commands

Below we summarize general capabilities of the mixed-effects commands. We let *mecmd* stand for any mixed-effects command, such as `mixed`, `melogit`, or `meprobit`, except `menl`. `menl` models the mean function nonlinearly and thus has a different syntax; see [ME] **menl**.

1. Fit a two-level random-intercept model with *levelvar* defining the second level:

   . *mecmd depvar* [*indepvars*] ... || *levelvar*: , ...

2. Fit a two-level random-coefficients model containing the random-effects covariates *revars* at the level *levelvar*:

   . *mecmd depvar* [*indepvars*] ... || *levelvar*: *revars*, ...

   This model assumes an independent covariance structure between the random effects; that is, all covariances are assumed to be 0. There is no statistical justification, however, for imposing any particular covariance structure between random effects at the onset of the analysis. In practice, models with an unstructured random-effects covariance matrix, which allows for distinct variances and covariances between all random-effects covariates (*revars*) at the same level, must be explored first; see *Other covariance structures* and example 3 in [ME] **melogit** for details.

   Stata's commands use the default independent covariance structure for computational feasibility. Numerical methods for fitting mixed-effects models are computationally intensive—computation time increases significantly as the number of parameters increases; see *Computation time and the Laplacian approximation* for details. The unstructured covariance is the most general and contains many parameters, which may result in an unreasonable computation time even for relatively simple random-effects models. Whenever feasible, however, you should start your statistical analysis by fitting mixed-effects models with an unstructured covariance between random effects, as we show next.

3. Specify the unstructured covariance between the random effects in the above:

   . *mecmd depvar* [*indepvars*] ... || *levelvar*: *revars*, `covariance(unstructured)` ...

4. Fit a three-level nested model with *levelvar1* defining the third level and *levelvar2* defining the second level:

   . *mecmd depvar* [*indepvars*] ... || *levelvar1*: || *levelvar2*:, ...

5. Fit the above three-level nested model as a two-level model with exchangeable covariance structure at the second level (`mixed` only):

   . *mecmd depvar* [*indepvars*] ... || *levelvar1*: R.*levelvar2*, `cov(exchangeable)` ...

   See example 13 in [ME] **mixed** for details about this equivalent specification. This specification may be useful for a more efficient fitting of random-effects models with a mixture of crossed and nested effects.

6. Fit higher-level nested models:

   . *mecmd depvar* [*indepvars*] ... || *levelvar1*: || *levelvar2*: || *levelvar3*: || ...

7. Fit a two-way crossed-effects model with the `_all:` notation for each random-effects equation:

   . *mecmd depvar* [*indepvars*] ... || `_all:` R.*factor1* || `_all:` R.*factor2* ...

   When you use the `_all:` notation for each random-effects equation, the total dimension of the random-effects design equals $r_1 + r_2$, where r_1 and r_2 are the numbers of levels in *factor1* and *factor2*, respectively. This specification may be infeasible for some mixed-effects models; see item 8 below for a more efficient specification of this model.

8. Fit a two-way crossed-effects model with the `_all:` notation for the first random-effects equation only:

   . *mecmd depvar* [*indepvars*] ... || `_all:` R.*factor1* || *factor2*:, ...

   Compared with the specification in item 7, this specification requires only $r_1 + 1$ parameters and is thus more efficient; see *Crossed-effects models* for details.

9. Fit a two-way full-factorial random-effects model:

```
. mecmd depvar [indepvars] ... || _all: R.factor1 || factor2: || factor1: ...
```

10. Fit a two-level mixed-effects model with a blocked-diagonal covariance structure between *revars1* and *revars2*:

```
. mecmd depvar [indepvars] ... || levelvar: revars1, noconstant ///
                              || levelvar: revars2, noconstant ...
```

11. Fit a linear mixed-effects model where the correlation between the errors follows an autoregressive process of order 1:

```
. mixed depvar [indepvars] ... || levelvar:, residuals(ar 1, t(time)) ...
```

    More residual error structures are available; see [ME] **mixed** for details.

12. Fit a two-level linear mixed-effects model accounting for sampling weights *expr1* at the first level and for sampling weights *expr2* at the level of *levelvar*:

```
. mixed depvar [indepvars] [pweight=expr1] ... || levelvar:, pweight(expr2) ...
```

    Mixed-effects commands—with the exception of `mixed`—allow constraints on both fixed-effects and random-effects parameters. We provide several examples below of imposing constraints on variance components.

13. Fit a mixed-effects model with the variance of the random intercept on *levelvar* constrained to be 16:

```
. constraint 1 _b[var(_cons[levelvar]):_cons]=16
. mecmd depvar [indepvars] ... || levelvar:, constraints(1) ...
```

14. Fit a mixed-effects model with the variance of the random intercept on *levelvar* and the variance of the random slope on *revar* to be equal:

```
. constraint 1 _b[var(revar[levelvar]):_cons] = _b[var(_cons[levelvar]):_cons]
. mecmd depvar [indepvars] ... || levelvar: revar, constraints(1) ...
```

    Note that the constraints above are equivalent to imposing an identity covariance structure for the random-effects equation:

```
. mecmd depvar [indepvars] ... || levelvar: revar, cov(identity) ...
```

15. Assuming four random slopes *revars*, fit a mixed-effects model with the variance components at the level of *levelvar* constrained to have a banded structure:

```
. mat p = (1,.,.,. \ 2,1,.,. \ 3,2,1,. \ 4,3,2,1)
. mecmd depvar [indepvars] ... || levelvar: revars, noconstant ///
    covariance(pattern(p)) ...
```

16. Assuming four random slopes *revars*, fit a mixed-effects model with the variance components at the level of *levelvar* constrained to the specified numbers, and with all the covariances constrained to be 0:

```
. mat f = diag((1,2,3,4))
. mecmd depvar [indepvars] ... || levelvar: revars, noconstant ///
    covariance(fixed(f)) ...
```

The variance components in models in items 15 and 16 can also be constrained by using the `constraints()` option, but using `covariance(pattern())` or `covariance(fixed())` is more convenient.

Mixed-effects models

Linear mixed-effects models

Linear mixed-effects (LME) models for continuous responses are a generalization of linear regression allowing for the inclusion of random deviations (effects) other than those associated with the overall error term. In matrix notation,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon} \tag{1}$$

where $\mathbf{y}$ is the $n \times 1$ vector of responses, $\mathbf{X}$ is an $n \times p$ design/covariate matrix for the fixed effects $\boldsymbol{\beta}$, and $\mathbf{Z}$ is the $n \times q$ design/covariate matrix for the random effects $\mathbf{u}$. The $n \times 1$ vector of errors $\boldsymbol{\epsilon}$ is assumed to be multivariate normal with mean 0 and variance matrix $\sigma^2_\epsilon\mathbf{R}$.

The fixed portion of (1), $\mathbf{X}\boldsymbol{\beta}$, is analogous to the linear predictor from a standard OLS regression model with $\boldsymbol{\beta}$ being the regression coefficients to be estimated. For the random portion of (1), $\mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}$, we assume that $\mathbf{u}$ has variance–covariance matrix $\mathbf{G}$ and that $\mathbf{u}$ is orthogonal to $\boldsymbol{\epsilon}$ so that

$$\text{Var}\begin{bmatrix}\mathbf{u}\\ \boldsymbol{\epsilon}\end{bmatrix} = \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ \mathbf{0} & \sigma^2_\epsilon\mathbf{R}\end{bmatrix}$$

The random effects $\mathbf{u}$ are not directly estimated (although they may be predicted) but instead are characterized by the elements of $\mathbf{G}$, known as variance components, that are estimated along with the error covariance parameters that include the overall error variance σ^2_ϵ and the parameters that are contained within $\mathbf{R}$.

The general forms of the design matrices $\mathbf{X}$ and $\mathbf{Z}$ allow estimation for a broad class of linear models: blocked designs, split-plot designs, growth curves, multilevel or hierarchical designs, etc. They also allow a flexible method of modeling within-cluster correlation. Subjects within the same cluster can be correlated as a result of a shared random intercept, or through a shared random slope on age (for example), or both. The general specification of $\mathbf{G}$ also provides additional flexibility: the random intercept and random slope could themselves be modeled as independent, or correlated, or independent with equal variances, and so forth. The general structure of $\mathbf{R}$ also allows for within-cluster errors to be heteroskedastic and correlated and allows flexibility in exactly how these characteristics can be modeled.

In clustered-data situations, it is convenient not to consider all n observations at once but instead to organize the mixed model as a series of M independent groups (or clusters)

$$\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j + \boldsymbol{\epsilon}_j \tag{2}$$

for $j = 1, \ldots, M$, with cluster j consisting of n_j observations. The response $\mathbf{y}_j$ comprises the rows of $\mathbf{y}$ corresponding with the jth cluster, with $\mathbf{X}_j$ and $\boldsymbol{\epsilon}_j$ defined analogously. The random effects $\mathbf{u}_j$ can now be thought of as M realizations of a $q \times 1$ vector that is normally distributed with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The matrix $\mathbf{Z}_j$ is the $n_j \times q$ design matrix for the jth cluster random effects. Relating this to (1),

$$\mathbf{Z} = \begin{bmatrix}\mathbf{Z}_1 & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{Z}_2 & \cdots & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_M\end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix}\mathbf{u}_1\\ \vdots\\ \mathbf{u}_M\end{bmatrix}; \quad \mathbf{G} = \mathbf{I}_M \otimes \boldsymbol{\Sigma}; \quad \mathbf{R} = \mathbf{I}_M \otimes \boldsymbol{\Lambda}$$

where $\boldsymbol{\Lambda}$ denotes the variance matrix of the level-1 errors and $\otimes$ is the Kronecker product.

The mixed-model formulation (2) is from Laird and Ware (1982) and offers two key advantages. First, it makes specifications of random-effects terms easier. If the clusters are schools, you can simply specify a random effect at the school level, as opposed to thinking of what a school-level random effect would mean when all the data are considered as a whole (if it helps, think Kronecker products). Second, representing a mixed-model with (2) generalizes easily to more than one set of random effects. For example, if classes are nested within schools, then (2) can be generalized to allow random effects at both the school and the class-within-school levels.

By our convention on counting and ordering model levels, (2) is a two-level model, with extensions to three, four, or any number of levels. The observation y_{ij} is for individual i within cluster j, and the individuals compose the first level, whereas the clusters compose the second level of the model. In a hypothetical three-level model with classes nested within schools, the observations within classes (the students, presumably) would constitute the first level, the classes would constitute the second level, and the schools would constitute the third level. This differs from certain citations in the classical ANOVA literature and texts such as Pinheiro and Bates (2000) but is the standard in the vast literature on hierarchical models, for example, Skrondal and Rabe-Hesketh (2004).

In Stata, you can use `mixed` to fit linear mixed-effects models; see [ME] **mixed** for a detailed discussion and examples. Various predictions, statistics, and diagnostic measures are available after fitting an LME model with `mixed`. For the most part, calculation centers around obtaining estimates of random effects; see [ME] **mixed postestimation** for a detailed discussion and examples.

Generalized linear mixed-effects models

Generalized linear mixed-effects (GLME) models, also known as generalized linear mixed models (GLMMs), are extensions of generalized linear models allowing for the inclusion of random deviations (effects). In matrix notation,

$$g\{E(\mathbf{y}|\mathbf{X},\mathbf{u})\} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim F \tag{3}$$

where $\mathbf{y}$ is the $n \times 1$ vector of responses from the distributional family F, $\mathbf{X}$ is an $n \times p$ design/covariate matrix for the fixed effects $\boldsymbol{\beta}$, and $\mathbf{Z}$ is an $n \times q$ design/covariate matrix for the random effects $\mathbf{u}$. The $\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}$ part is called the linear predictor and is often denoted as $\boldsymbol{\eta}$. $g(\cdot)$ is called the link function and is assumed to be invertible such that

$$E(\mathbf{y}|\mathbf{u}) = g^{-1}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}) = H(\boldsymbol{\eta}) = \boldsymbol{\mu}$$

For notational convenience here and throughout this manual entry, we suppress the dependence of $\mathbf{y}$ on $\mathbf{X}$. Substituting various definitions for $g(\cdot)$ and F results in a wide array of models. For instance, if $g(\cdot)$ is the logit function and $\mathbf{y}$ is distributed as Bernoulli, we have

$$\text{logit}\{E(\mathbf{y}|\mathbf{u})\} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim \text{Bernoulli}$$

or mixed-effects logistic regression. If $g(\cdot)$ is the natural log function and $\mathbf{y}$ is distributed as Poisson, we have

$$\ln\{E(\mathbf{y}|\mathbf{u})\} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim \text{Poisson}$$

or mixed-effects Poisson regression.

For the random portion of (3), $\mathbf{Z}\mathbf{u}$, we assume that $\mathbf{u}$ has variance–covariance matrix $\mathbf{G}$ such that

$$\text{Var}(\mathbf{u}) = \mathbf{G}$$

The random effects $\mathbf{u}$ are not directly estimated (although they may be predicted) but instead are characterized by the elements of $\mathbf{G}$, known as variance components.

Analogously to (2), in clustered-data situations, we can write

$$E(\mathbf{y}_j|\mathbf{u}_j) = g^{-1}(\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j) \qquad \mathbf{y}_j \sim F$$

with all the elements defined as before. In terms of the whole dataset, we now have

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_M \end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_M \end{bmatrix}; \quad \mathbf{G} = \mathbf{I}_M \otimes \boldsymbol{\Sigma}$$

In Stata, you can use `meglm` to fit mixed-effects models for nonlinear responses. Some combinations of families and links are so common that we implemented them as separate commands in terms of `meglm`.

Command	`meglm` equivalent
`melogit`	`family(bernoulli) link(logit)`
`meprobit`	`family(bernoulli) link(probit)`
`mecloglog`	`family(bernoulli) link(cloglog)`
`meologit`	`family(ordinal) link(logit)`
`meoprobit`	`family(ordinal) link(probit)`
`mepoisson`	`family(poisson) link(log)`
`menbreg`	`family(nbinomial) link(log)`

When no family–link combination is specified, `meglm` defaults to a Gaussian family with an identity link. Thus `meglm` can be used to fit linear mixed-effects models; however, for those models we recommend using the more specialized `mixed`, which, in addition to `meglm` capabilities, allows for modeling of the structure of the within-cluster errors; see [ME] **mixed** for details.

Various predictions, statistics, and diagnostic measures are available after fitting a GLME model with `meglm` and other `me` commands. For the most part, calculation centers around obtaining estimates of random effects; see [ME] **meglm postestimation** for a detailed discussion and examples.

Survival mixed-effects models

Parametric survival mixed-effects models use a trivariate response variable (t_0, t, d), where each response corresponds to a period under observation $(t_0, t]$ and results in either failure ($d = 1$) or right-censoring ($d = 0$) at time t. See [ST] **streg** for background information on parametric survival models. Two often-used models for adjusting survivor functions for the effects of covariates are the accelerated failure-time (AFT) model and the multiplicative or proportional hazards (PH) model.

In the AFT parameterization, the natural logarithm of the survival time, $\log t$, is expressed as a linear function of the covariates. When we incorporate random effects, this yields the model

$$\log(\mathbf{t}_j) = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j + \mathbf{v}_j$$

where $\log(\cdot)$ is an elementwise function, and $\mathbf{v}_j$ is a vector of observation-level errors. The distributional form of the error term determines the regression model.

In the PH model, the covariates have a multiplicative effect on the hazard function

$$h(\mathbf{t}_j) = h_0(\mathbf{t}_j)\exp(\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j)$$

where all the functions are elementwise, and $h_0(\cdot)$ is a baseline hazard function. The functional form of $h_0(\cdot)$ determines the regression model.

In Stata, you can use `mestreg` to fit multilevel mixed-effects parametric survival models for the following distributions and parameterizations.

Distribution	Parameterization
`exponential`	PH, AFT
`loglogistic`	AFT
`weibull`	PH, AFT
`lognormal`	AFT
`gamma`	AFT

`mestreg` is suitable only for data that have been set using the `stset` command. By using `stset` on your data, you define the variables `_t0`, `_t`, and `_d`, which serve as the trivariate response. See [ME] **mestreg** for more details about the command. Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects survival model with `mestreg`; see [ME] **mestreg postestimation** for a detailed discussion and examples.

Nonlinear mixed-effects models

NLME models are models containing both fixed effects and random effects where some of, or all, the fixed and random effects enter the model nonlinearly. They can be viewed as a generalization of LME models, in which the conditional mean of the outcome given the random effects is a nonlinear function of the coefficients and random effects. Alternatively, they can be considered as an extension of nonlinear regression models for independent data (see [R] **nl**), in which coefficients may incorporate random effects, allowing them to vary across different levels of hierarchy and thus inducing correlation within observations at the same level.

Using the notation from *Linear mixed-effects models* for LME models for clustered data, we can write an NLME model as

$$\mathbf{y}_j = \boldsymbol{\mu}\left(\mathbf{A}_j, \boldsymbol{\beta}, \mathbf{u}_j\right) + \boldsymbol{\epsilon}_j$$

where $\boldsymbol{\mu}(\cdot)$ is a real-valued vector function and $\mathbf{A}_j$ is an $n_j \times l$ matrix of covariates for the jth cluster, which includes both within-subject and between-subject covariates. Do not be surprised to see the $\mathbf{A}_j$ matrix here instead of the more familiar fixed-effects and random-effects design matrices $\mathbf{X}_j$ and $\mathbf{Z}_j$ from previous sections. Because both covariates and parameters can enter the model nonlinearly in NLME, we cannot express the regression function as a function containing the linear term $\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j$ as we can for LME and GLME models. The distributional assumptions on $\mathbf{u}_j$'s and $\boldsymbol{\epsilon}_j$'s are the same as for the LME models.

Parameters of NLME models often have scientifically meaningful interpretations, and research questions are formed based on them. To allow parameters to reflect phenomena of interest, NLME models are often formulated by using a multistage formulation; see *Alternative mixed-effects model specification* below for examples.

We can formulate our previous NLME model as a two-stage hierarchical model:

$$\text{Stage 1: Individual-level model } y_{ij} = m\left(\mathbf{x}_{ij}^{w},\, \boldsymbol{\phi}_j\right) + \epsilon_{ij}, \quad i = 1, \ldots, n_j$$
$$\text{Stage 2: Group-level model } \boldsymbol{\phi}_j = \boldsymbol{d}\left(\mathbf{x}_j^{b},\, \boldsymbol{\beta},\, \mathbf{u}_j\right), \quad j = 1, \ldots, M$$

In stage 1, we model the response by using a function $m(\cdot)$, which describes within-subject behavior. This function depends on subject-specific parameters $\boldsymbol{\phi}_j$'s, which have a natural physical interpretation, and a vector of within-subject covariates $\mathbf{x}_{ij}^{w}$. In stage 2, we use a known vector-valued function $\boldsymbol{d}(\cdot)$ to model between-subject behavior, that is, to model $\boldsymbol{\phi}_j$'s and to explain how they vary across subjects. The $\boldsymbol{d}(\cdot)$ function incorporates random effects and, optionally, a vector of between-subject covariates $\mathbf{x}_j^{b}$. The general idea is to specify a common functional form for each subject in stage 1 and then allow some parameters to vary randomly across subjects in stage 2.

You can use the `menl` command to fit NLME models to continuous outcomes; see [ME] **menl**. `menl` supports both the single-equation and multistage model formulations. It supports different covariance structures for random effects and can model heteroskedasticity and correlations within lowest-level groups. Various predictions, statistics, and diagnostic measures are available after fitting an NLME model; see [ME] **menl postestimation**.

For an introductory example, see *Nonlinear models*.

Alternative mixed-effects model specification

In this section, we present a hierarchical or multistage formulation of mixed-effects models where each level is described by its own set of equations. This formulation is common for NLME models; see *Nonlinear mixed-effects models*.

Consider a random-intercept model that we write here in general terms:

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + u_j + \epsilon_{ij} \tag{4}$$

This single-equation specification contains both level-1 and level-2 effects. In the hierarchical form, we specify a separate equation for each level.

$$\begin{aligned} y_{ij} &= \gamma_{0j} + \beta_1 x_{ij} + \epsilon_{ij} \\ \gamma_{0j} &= \beta_{00} + u_{0j} \end{aligned} \tag{5}$$

The equation for the intercept γ_{0j} consists of the overall mean intercept β_{00} and a cluster-specific random intercept u_{0j}. To fit this model by using, for example, `mixed`, we must translate the multiple-equation notation into a single-equation form. We substitute the second equation into the first one and rearrange terms.

$$\begin{aligned} y_{ij} &= \beta_{00} + u_{0j} + \beta_1 x_{ij} + \epsilon_{ij} \\ &= \beta_{00} + \beta_1 x_{ij} + u_{0j} + \epsilon_{ij} \end{aligned} \tag{6}$$

Note that model (6) is the same as model (4) with $\beta_{00} \equiv \beta_0$ and $u_{0j} \equiv u_j$. Thus the syntax for our generic random-intercept model is

```
. mixed y x || id:
```

where `id` is the variable designating the clusters.

We can extend model (5) to include a random slope. We do so by specifying an additional equation for the slope on x_{ij}.

$$\begin{aligned} y_{ij} &= \gamma_{0j} + \gamma_{1j}x_{ij} + \epsilon_{ij} \\ \gamma_{0j} &= \beta_{00} + u_{0j} \\ \gamma_{1j} &= \beta_{10} + u_{1j} \end{aligned} \tag{7}$$

The additional equation for the slope γ_{1j} consists of the overall mean slope β_{10} and a cluster-specific random slope u_{1j}. We substitute the last two equations into the first one to obtain a reduced-form model.

$$\begin{aligned} y_{ij} &= (\beta_{00} + u_{0j}) + (\beta_{10} + u_{1j})x_{ij} + \epsilon_{ij} \\ &= \beta_{00} + \beta_{10}x_{ij} + u_{0j} + u_{1j}x_{ij} + \epsilon_{ij} \end{aligned}$$

The syntax for this model becomes

```
. mixed y x || id: x, covariance(unstructured)
```

where we specified an unstructured covariance structure for the level-2 u terms.

Here we further extend the random-slope random-intercept model (7) by adding a level-2 covariate z_j into the level-2 equations.

$$\begin{aligned} y_{ij} &= \gamma_{0j} + \gamma_{1j}x_{ij} + \epsilon_{ij} \\ \gamma_{0j} &= \beta_{00} + \beta_{01}z_j + u_{0j} \\ \gamma_{1j} &= \beta_{10} + \beta_{11}z_j + u_{1j} \end{aligned}$$

We substitute as before to obtain a single-equation form:

$$\begin{aligned} y_{ij} &= (\beta_{00} + \beta_{01}z_j + u_{0j}) + (\beta_{10} + \beta_{11}z_j + u_{1j})x_{ij} + \epsilon_{ij} \\ &= \beta_{00} + \beta_{01}z_j + \beta_{10}x_{ij} + \beta_{11}z_jx_{ij} + u_{0j} + u_{1j}x_{ij} + \epsilon_{ij} \end{aligned}$$

Now the fixed-effects portion of the equation contains a constant and variables x, z, and their interaction. Assuming both x and z are continuous variables, we can use the following Stata syntax to fit this model:

```
. mixed y x z c.x#c.z  || id: x, covariance(unstructured)
```

Although the `menl` command is not as suitable for fitting LME models as `mixed`, it can accommodate a multistage formulation. For example, (5) can be fit in `menl` as

```
. menl y = {gamma0:}+{b1}*x, define(gamma0: {b00}+{U0[id]})
```

and (7) as

```
. menl y = {gamma0:}+{gamma1:}*x, define(gamma0: {b00}+{U0[id]}) ///
                                  define(gamma1: {b10}+{U1[id]})
```

In the above `menl`'s specifications, `gamma0` and `gamma1` can be specified more efficiently by using linear combinations; see [ME] **menl** for details.

We refer you to Raudenbush and Bryk (2002) and Rabe-Hesketh and Skrondal (2022) for a more thorough discussion and further examples of multistage mixed-model formulations, including three-level models.

Likelihood calculation

The key to fitting mixed models lies in estimating the variance components, and for that there exist many methods. Most of the early literature in LME models dealt with estimating variance components in ANOVA models. For simple models with balanced data, estimating variance components amounts to solving a system of equations obtained by setting expected mean-squares expressions equal to their observed counterparts. Much of the work in extending the ANOVA method to unbalanced data for general ANOVA designs is attributed to Henderson (1953).

The ANOVA method, however, has its shortcomings. Among these is a lack of uniqueness in that alternative, unbiased estimates of variance components could be derived using other quadratic forms of the data in place of observed mean squares (Searle, Casella, and McCulloch 1992, 38–39). As a result, ANOVA methods gave way to more modern methods, such as minimum norm quadratic unbiased estimation (MINQUE) and minimum variance quadratic unbiased estimation (MIVQUE); see Rao (1973) for MINQUE and LaMotte (1973) for MIVQUE. Both methods involve finding optimal quadratic forms of the data that are unbiased for the variance components.

Stata uses maximum likelihood (ML) to fit LME and GLME models. The ML estimates are based on the usual application of likelihood theory, given the distributional assumptions of the model. In addition, for linear mixed-effects models, `mixed` offers the method of restricted maximum likelihood (REML). The basic idea behind REML (Thompson 1962) is that you can form a set of linear contrasts of the response that do not depend on the fixed effects $\boldsymbol{\beta}$ but instead depend only on the variance components to be estimated. You then apply ML methods by using the distribution of the linear contrasts to form the likelihood; see the *Methods and formulas* section of [ME] **mixed** for a detailed discussion of ML and REML methods in the context of linear mixed-effects models.

Log-likelihood calculations for fitting any mixed-effects model require integrating out the random effects. For LME models, this integral has a closed-form solution; for GLME and NLME models, it does not. In dealing with this difficulty, early estimation methods avoided the integration altogether. Two such popular methods are the closely related penalized quasilikelihood (PQL) and marginal quasilikelihood (MQL) (Breslow and Clayton 1993). Both PQL and MQL use a combination of iterative reweighted least squares (see [R] **glm**) and standard estimation techniques for fitting LME models. Efficient computational methods for fitting LME models have existed for some time (Bates and Pinheiro 1998; Littell et al. 2006), and PQL and MQL inherit this computational efficiency. However, both of these methods suffer from two key disadvantages. First, they have been shown to be biased, and this bias can be severe when clusters are small or intracluster correlation is high (Rodríguez and Goldman 1995; Lin and Breslow 1996). Second, because they are "quasilikelihood" methods and not true likelihood methods, their use prohibits comparing nested models via likelihood-ratio (LR) tests, blocking the main avenue of inference involving variance components.

The advent of modern computers has brought with it the development of more computationally intensive methods, such as bias-corrected PQL (Lin and Breslow 1996), Bayesian Markov-Chain Monte Carlo, and simulated maximum likelihood, just to name a few; see Ng et al. (2006) for a discussion of these alternate strategies (and more) for mixed-effects models for binary outcomes.

One widely used modern method is to directly estimate the integral required to calculate the log likelihood by Gauss–Hermite quadrature or some variation thereof. Because the log likelihood itself is estimated, this method has the advantage of permitting LR tests for comparing nested models. Also, if done correctly, quadrature approximations can be quite accurate, thus minimizing bias. Stata commands for fitting GLME models such as `meglm` support three types of Gauss–Hermite quadratures: mean–variance adaptive Gauss–Hermite quadrature, mode-curvature adaptive Gauss–Hermite quadrature, and nonadaptive Gauss–Hermite quadrature; see *Methods and formulas* of [ME] **meglm** for a detailed discussion of

these quadrature methods. A fourth method, the Laplacian approximation, that does not involve numerical integration is also offered; see *Computation time and the Laplacian approximation* below and *Methods and formulas* of [ME] **meglm** for a detailed discussion of the Laplacian approximation method.

In the context of NLME models, the use of an adaptive quadrature to fit these models can be often computationally infeasible. A popular alternative method used to fit NLME models is the linearization method of Lindstrom and Bates (1990), also known as the conditional first-order linearization method. It is based on a first-order Taylor-series approximation of the mean function and essentially linearizes the mean function with respect to fixed and random effects. The linearization method is computationally efficient because it avoids the intractable integration, but the approximation cannot be made arbitrarily accurate. Despite its potential limiting accuracy, the linearization method has proven the most popular in practice (Fitzmaurice et al. 2009, sec 5.4.8). The linearization method of Lindstrom and Bates (1990), with extensions from Pinheiro and Bates (1995), is the method of estimation in `menl`.

Computation time and the Laplacian approximation

Like many programs that fit generalized linear mixed models, `me` commands can be computationally intensive. This is particularly true for large datasets with many lowest-level clusters, models with many random coefficients, models with many estimable parameters (both fixed effects and variance components), or any combination thereof.

Computation time will also depend on hardware and other external factors but in general is (roughly) a function of $p^2\{M + M(N_Q)^{q_t}\}$, where p is the number of estimable parameters, M is the number of lowest-level (smallest) clusters, N_Q is the number of quadrature points, and q_t is the total dimension of the random effects, that is, the total number of random intercepts and coefficients at all levels.

For a given model and a given dataset, the only prevailing factor influencing computation time is $(N_Q)^{q_t}$. However, because this is a power function, this factor can get prohibitively large. For example, using five quadrature points for a model with one random intercept and three random coefficients, we get $(N_Q)^{q_t} = 5^4 = 625$. Even a modest increase to seven quadrature points would increase this factor by almost fourfold ($7^4 = 2{,}401$), which, depending on M and p, could drastically slow down estimation. When fitting mixed-effects models, you should always assess whether the approximation is adequate by refitting the model with a larger number of quadrature points. If the results are essentially the same, the lower number of quadrature points can be used.

However, we do not deny a tradeoff between speed and accuracy, and in that spirit we give you the option to choose a (possibly) less accurate solution in the interest of getting quicker results. Toward this end is the limiting case of $N_Q = 1$, otherwise known as the Laplacian approximation; see *Methods and formulas* of [ME] **meglm**. The computational benefit is evident—1 raised to any power equals 1—and the Laplacian approximation has been shown to perform well in certain situations (Liu and Pierce 1994; Tierney and Kadane 1986). When using Laplacian approximation, keep the following in mind:

1. Fixed-effects parameters and their standard errors are well approximated by the Laplacian method. Therefore, if your interest lies primarily here, then the Laplacian approximation may be a viable alternative.
2. Estimates of variance components exhibit bias, particularly the variances.
3. The model log likelihood and comparison LR test are in fair agreement with statistics obtained via quadrature methods.

Although this is by no means the rule, we find the above observations to be fairly typical based on our own experience. Pinheiro and Chao (2006) also make observations similar to points 1 and 2 on the basis of their simulation studies: bias due to Laplace (when present) tends to exhibit itself more in the estimated variance components than in the estimates of the fixed effects as well as at the lower levels in higher-level models.

Item 3 is of particular interest, because it demonstrates that the Laplacian approximation can produce a decent estimate of the model log likelihood. Consequently, you can use the Laplacian approximation during the model building phase of your analysis, during which you are comparing competing models by using LR tests. Once you settle on a parsimonious model that fits well, you can then increase the number of quadrature points and obtain more accurate parameter estimates for further study.

Of course, sometimes the Laplacian approximation will perform either better or worse than observed here. This behavior depends primarily on cluster size and intracluster correlation, but the relative influence of these factors is unclear. The idea behind the Laplacian approximation is to approximate the posterior density of the random effects given the response with a normal distribution; see *Methods and formulas* of [ME] **meglm**. Asymptotic theory dictates that this approximation improves with larger clusters. Of course, the key question, as always, is "How large is large enough?" Also, there are data situations where the Laplacian approximation performs well even with small clusters. Therefore, it is difficult to make a definitive call as to when you can expect the Laplacian approximation to yield accurate results across all aspects of the model.

Furthermore, the Pinheiro and Chao (2006) algorithm for the random-effects mode and curvature estimates, available with option `intmethod(pclaplace)`, can speed up computations dramatically for hierarchical models with four or more levels, especially when random slopes are included.

In conclusion, consider our above advice as a rule of thumb based on empirical evidence.

Diagnosing convergence problems

Given the flexibility of mixed-effects models, you will find that some models fail to converge when used with your data. The default gradient-based method used by mixed-effects commands, except `menl`, is the Newton–Raphson algorithm, requiring the calculation of a gradient vector and Hessian (second-derivative) matrix; see [R] **ml**.

A failure to converge can take any one of three forms:

1. repeated nonconcave or backed-up iterations without convergence;
2. a Hessian (second-derivative) calculation that has become asymmetric, unstable, or has missing values; or
3. the message "standard error calculation has failed" when computing standard errors.

All three situations essentially amount to the same thing: the Hessian calculation has become unstable, most likely because of a ridge in the likelihood function, a subsurface of the likelihood in which all points give the same value of the likelihood and for which there is no unique solution.

Such behavior is usually the result of one of the following two situations:

A. A model that is not identified given the data, for example, fitting the three-level nested random intercept model

$$y_{jk} = \mathbf{x}_{jk}\boldsymbol{\beta} + u_k^{(3)} + u_{jk}^{(2)} + \epsilon_{jk}$$

without any replicated measurements at the (j, k) level, that is, with only one i per (j, k) combination. This model is unidentified for such data because the random intercepts $u_{jk}^{(2)}$ are confounded with the overall errors ϵ_{jk}.

B. A model that contains a variance component whose estimate is really close to 0. When this occurs, a ridge is formed by an interval of values near 0, which produce the same likelihood and look equally good to the optimizer.

For LME models, one useful way to diagnose problems of nonconvergence is to rely on the expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977), normally used by `mixed` only as a means of refining starting values; see *Diagnosing convergence problems* of [ME] **mixed** for details.

If your data and model are nearly unidentified, as opposed to fully unidentified, you may be able to obtain convergence with standard errors by changing some of the settings of the gradient-based optimization. Adding the `difficult` option can be particularly helpful if you are seeing many "nonconcave" messages; you may also consider changing the `technique()` or using the `nonrtolerance` option; see [R] **Maximize**.

Regardless of how the convergence problem revealed itself, you may try to obtain better starting values; see *Obtaining better starting values* in [ME] **meglm** for details.

Achieving convergence and diagnosing convergence problems can be even more challenging with NLME models. As with other mixed-effects models, complicated variance–covariance structures for random effects and errors can often lead to overparameterized models that fail to converge. In addition, highly nonlinear mean specifications can lead to multiple solutions and thus to potential convergence to a local maximum. `menl` uses the linearization estimation method that alternates between the penalized least-squares estimation of the fixed-effects parameters and the Newton–Raphson estimation of the random-effects parameters of the approximating LME model, which was the result of the linearization of the original NLME model. This alternating method does not provide a joint Hessian matrix for all parameters, so there is no check for the tolerance of the scaled gradient, and thus the convergence cannot be established in its strict sense. The convergence is declared based on the stopping rules described in *Methods and formulas* of [ME] **menl**. Exploring different initial values to investigate convergence is particularly important with NLME models; see *Obtaining initial values* in [ME] **menl**.

Distribution theory for likelihood-ratio test

When determining the asymptotic distribution of an LR test comparing two nested mixed-effects models, issues concerning boundary problems imposed by estimating strictly positive quantities (that is, variances) can complicate the situation. For example, when performing LR tests involving linear mixed-effects models (whether comparing with linear regression within `mixed` or comparing two separate linear mixed-effects models with `lrtest`), you may thus sometimes see a test labeled as `chibar` rather than the usual `chi2`, or you may see a `chi2` test with a note attached stating that the test is conservative or possibly conservative depending on the hypothesis being tested.

At the heart of the issue is the number of variances being restricted to 0 in the reduced model. If there are none, the usual asymptotic theory holds, and the distribution of the test statistic is χ^2 with degrees of freedom equal to the difference in the number of estimated parameters between both models.

When there is only one variance being set to 0 in the reduced model, the asymptotic distribution of the LR test statistic is a 50:50 mixture of a χ^2_p and a χ^2_{p+1} distribution, where p is the number of other restricted parameters in the reduced model that are unaffected by boundary conditions. Stata labels such test statistics as `chibar` and adjusts the significance levels accordingly. See Self and Liang (1987) for the appropriate theory or Gutierrez, Carter, and Drukker (2001) for a Stata-specific discussion.

When more than one variance parameter is being set to 0 in the reduced model, however, the situation becomes more complicated. For example, consider a comparison test versus linear regression for a mixed model with two random coefficients and unstructured covariance matrix

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_0^2 & \sigma_{01} \\ \sigma_{01} & \sigma_1^2 \end{bmatrix}$$

Because the random component of the mixed model comprises three parameters $(\sigma_0^2, \sigma_{01}, \sigma_1^2)$, on the surface it would seem that the LR comparison test would be distributed as χ^2_3. However, two complications need to be considered. First, the variances σ_0^2 and σ_1^2 are restricted to be positive, and second, constraints such as $\sigma_1^2 = 0$ implicitly restrict the covariance σ_{01} to be 0 as well. From a technical standpoint, it is unclear how many parameters must be restricted to reduce the model to linear regression.

Because of these complications, appropriate and sufficiently general distribution theory for the more-than-one-variance case has yet to be developed. Theory (for example, Stram and Lee [1994]) and empirical studies (for example, McLachlan and Basford [1988]) have demonstrated that, whatever the distribution of the LR test statistic, its tail probabilities are bounded above by those of the χ^2 distribution with degrees of freedom equal to the full number of restricted parameters (three in the above example).

The `mixed` and `me` commands use this reference distribution, the χ^2 with full degrees of freedom, to produce a conservative test and place a note in the output labeling the test as such. Because the displayed significance level is an upper bound, rejection of the null hypothesis based on the reported level would imply rejection on the basis of the actual level.

Examples

Two-level models

▷ Example 1: Growth-curve model

Consider a longitudinal dataset, used by both Ruppert, Wand, and Carroll (2003) and Diggle et al. (2002), consisting of `weight` measurements of 48 pigs on 9 successive `weeks`. Pigs are identified by the variable `id`. Each pig experiences a linear trend in growth, but overall weight measurements vary from pig to pig. Because we are not really interested in these particular 48 pigs per se, we instead treat them as a random sample from a larger population and model the between-pig variability as a random effect, or in the terminology of (2), as a random-intercept term at the pig level. We thus wish to fit the model

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_j + \epsilon_{ij}$$

for $i = 1, \ldots, 9$ weeks and $j = 1, \ldots, 48$ pigs. The fixed portion of the model, $\beta_0 + \beta_1 \texttt{week}_{ij}$, simply states that we want one overall regression line representing the population average. The random effect u_j serves to shift this regression line up or down according to each pig. Because the random effects occur at the pig level (`id`), we fit the model by typing

```
. use https://www.stata-press.com/data/r19/pig
(Longitudinal analysis of pig weights)
. mixed weight week || id:
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -1014.9268
Iteration 1:  Log likelihood = -1014.9268
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =   25337.49
Log likelihood = -1014.9268                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0390124	159.18	0.000	6.133433	6.286359
_cons	19.35561	.5974059	32.40	0.000	18.18472	20.52651

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(_cons)	14.81751	3.124225	9.801716	22.40002
var(Residual)	4.383264	.3163348	3.805112	5.04926

```
LR test vs. linear model: chibar2(01) = 472.65        Prob >= chibar2 = 0.0000
```

We explain the output in detail in example 1 of [ME] **mixed**. Here we only highlight the most important points.

1. The first estimation table reports the fixed effects. We estimate $\beta_0 = 19.36$ and $\beta_1 = 6.21$.
2. The second estimation table shows the estimated variance components. The first section of the table is labeled `id: Identity`, meaning that these are random effects at the `id` (pig) level and that their variance–covariance matrix is a multiple of the identity matrix; that is, $\boldsymbol{\Sigma} = \sigma_u^2\mathbf{I}$. The estimate of $\hat{\sigma}_u^2$ is 14.82 with standard error 3.12.
3. The row labeled `var(Residual)` displays the estimated standard deviation of the overall error term; that is, $\hat{\sigma}_\epsilon^2 = 4.38$. This is the estimated variance of the level-one errors, which are loosely labeled as `Residual` in the output.
4. An LR test comparing the model with one-level ordinary linear regression is provided and is highly significant for these data.

We can predict the random intercept u_j and list the predicted random intercept for the first 10 pigs by typing

```
. predict r_int, reffects
. egen byte tag = tag(id)
. list id r_int if id<=10 & tag
```

	id	r_int
1.	1	-1.683105
10.	2	.8987018
19.	3	-1.952043
28.	4	-1.79068
37.	5	-3.189159
46.	6	-3.780823
55.	7	-2.382344
64.	8	-1.952043
73.	9	-6.739143
82.	10	1.16764

In example 3 of [ME] **mixed**, we show how to fit a random-slope model for these data, and in example 1 of [ME] **mixed postestimation**, we show how to plot the estimated regression lines for each of the pigs. ◁

▷ Example 2: Split-plot design

Here we replicate the example of a split-plot design from Kuehl (2000, 477). The researchers investigate the effects of nitrogen in four different chemical forms and the effects of thatch accumulation on the quality of golf turf. The experimental plots were arranged in a randomized complete block design with two replications. After two years of nitrogen treatment, the second treatment factor, years of thatch accumulation, was added to the experiment. Each of the eight experimental plots was split into three subplots. Within each plot, the subplots were randomly assigned to accumulate thatch for a period of 2, 5, and 8 years.

```
. use https://www.stata-press.com/data/r19/clippings, clear
(Turfgrass experiment)
. describe
Contains data from https://www.stata-press.com/data/r19/clippings.dta
 Observations:            24                  Turfgrass experiment
    Variables:             4                  21 Feb 2024 14:57
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
chlorophyll     float   %9.0g                 Chlorophyll content (mg/g) of
                                                grass clippings
thatch          byte    %9.0g                 Years of thatch accumulation
block           byte    %9.0g                 Replication
nitrogen        byte    %17.0g     nitrolab   Nitrogen fertilizer
-------------------------------------------------------------------------------
Sorted by:
```

Nitrogen treatment is stored in the variable `nitrogen`, and the chemicals used are urea, ammonium sulphate, isobutylidene diurea (IBDU), and sulphur-coated urea (urea SC). The length of thatch accumulation is stored in the variable `thatch`. The response is the chlorophyll content of grass clippings, recorded in mg/g and stored in the variable `chlorophyll`. The `block` variable identifies the replication group.

There are two sources of variation in this example corresponding to the whole-plot errors and the subplot errors. The subplot errors are the level-one errors. The whole-plot errors represents variation in the chlorophyll content across nitrogen treatments and replications. We create the variable `wpunit` to represent the whole-plot units that correspond to the levels of the nitrogen treatment and block interaction.

```
. egen wpunit = group(nitrogen block)
. mixed chlorophyll ibn.nitrogen##ibn.thatch ibn.block, noomitted noconstant ||
> wpunit:, reml
note: 8.thatch omitted because of collinearity.
note: 1.nitrogen#8.thatch omitted because of collinearity.
note: 2.nitrogen#8.thatch omitted because of collinearity.
note: 3.nitrogen#8.thatch omitted because of collinearity.
note: 4.nitrogen#2.thatch omitted because of collinearity.
note: 4.nitrogen#5.thatch omitted because of collinearity.
note: 4.nitrogen#8.thatch omitted because of collinearity.
note: 2.block omitted because of collinearity.
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log restricted-likelihood = -13.212401
Iteration 1:  Log restricted-likelihood = -13.203147
Iteration 2:  Log restricted-likelihood = -13.203125
Iteration 3:  Log restricted-likelihood = -13.203125
```

```
Computing standard errors ...
Mixed-effects REML regression                   Number of obs     =         24
Group variable: wpunit                          Number of groups  =          8
                                                Obs per group:
                                                              min =          3
                                                              avg =        3.0
                                                              max =          3
                                                Wald chi2(13)     =    2438.36
Log restricted-likelihood = -13.203125          Prob > chi2       =     0.0000
```

chlorophyll	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
nitrogen						
Urea	5.245833	.3986014	13.16	0.000	4.464589	6.027078
Ammonium s..	5.945833	.3986014	14.92	0.000	5.164589	6.727078
IBDU	7.945834	.3986014	19.93	0.000	7.164589	8.727078
Urea (SC)	8.595833	.3986014	21.56	0.000	7.814589	9.377078
thatch						
2	-1.1	.4632314	-2.37	0.018	-2.007917	-.1920828
5	.1500006	.4632314	0.32	0.746	-.7579163	1.057917
nitrogen# thatch						
Urea#2	-.1500005	.6551081	-0.23	0.819	-1.433989	1.133988
Urea#5	.0999994	.6551081	0.15	0.879	-1.183989	1.383988
Ammonium s.. # 2	.8999996	.6551081	1.37	0.169	-.3839887	2.183988
Ammonium s.. # 5	-.1000006	.6551081	-0.15	0.879	-1.383989	1.183988
IBDU#2	-.2000005	.6551081	-0.31	0.760	-1.483989	1.083988
IBDU#5	-1.950001	.6551081	-2.98	0.003	-3.233989	-.6660124
block						
1	-.2916666	.2643563	-1.10	0.270	-.8097955	.2264622

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
wpunit: Identity				
var(_cons)	.0682407	.1195933	.0021994	2.117345
var(Residual)	.2145833	.1072917	.080537	.5717376

```
LR test vs. linear model: chibar2(01) = 0.53          Prob >= chibar2 = 0.2324
```

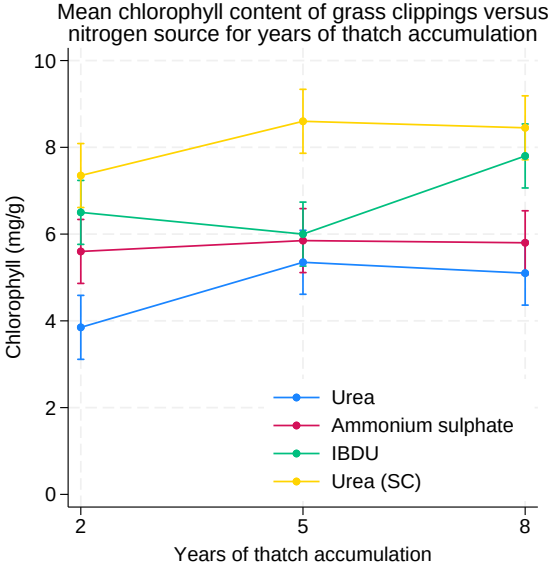
We can calculate the cell means for source of nitrogen and years of thatch accumulation by using `margins`.

```
. margins thatch#nitrogen
Predictive margins                                      Number of obs = 24
Expression: Linear prediction, fixed portion, predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
thatch# nitrogen						
2#Urea	3.85	.3760479	10.24	0.000	3.11296	4.58704
2 # Ammonium s..	5.6	.3760479	14.89	0.000	4.86296	6.33704
2#IBDU	6.5	.3760479	17.29	0.000	5.76296	7.23704
2#Urea (SC)	7.35	.3760479	19.55	0.000	6.61296	8.087041
5#Urea	5.35	.3760479	14.23	0.000	4.61296	6.087041
5 # Ammonium s..	5.85	.3760479	15.56	0.000	5.11296	6.58704
5#IBDU	6	.3760479	15.96	0.000	5.26296	6.73704
5#Urea (SC)	8.6	.3760479	22.87	0.000	7.86296	9.337041
8#Urea	5.1	.3760479	13.56	0.000	4.36296	5.837041
8 # Ammonium s..	5.8	.3760479	15.42	0.000	5.06296	6.53704
8#IBDU	7.8	.3760479	20.74	0.000	7.06296	8.537041
8#Urea (SC)	8.45	.3760479	22.47	0.000	7.712959	9.18704

It is easier to see the effect of the treatments if we plot the impact of the four nitrogen and the three thatch treatments. We can use `marginsplot` to plot the means of chlorophyll content versus years of thatch accumulation by nitrogen source.

```
. marginsplot, ytitle(Chlorophyll (mg/g)) title("")
> subtitle("Mean chlorophyll content of grass clippings versus"
> "nitrogen source for years of thatch accumulation") xsize(3) ysize(3.2)
> legend(cols(1) position(5) ring(0) region(lwidth(none)))
> ylabel(0(2)10, angle(0))
Variables that uniquely identify margins: thatch nitrogen
```



We can see an increase in the mean chlorophyll content over the years of thatch accumulation for all but one nitrogen source.

The marginal means can be obtained by using `margins` on one variable at a time.

```
. margins thatch
Predictive margins                                      Number of obs = 24
Expression: Linear prediction, fixed portion, predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
thatch						
2	5.825	.188024	30.98	0.000	5.45648	6.19352
5	6.45	.188024	34.30	0.000	6.08148	6.81852
8	6.7875	.188024	36.10	0.000	6.41898	7.15602

```
. margins nitrogen
Predictive margins                                      Number of obs = 24
Expression: Linear prediction, fixed portion, predict()
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
nitrogen						
Urea	4.766667	.2643563	18.03	0.000	4.248538	5.284796
Ammonium s..	5.75	.2643563	21.75	0.000	5.231871	6.268129
IBDU	6.766667	.2643563	25.60	0.000	6.248538	7.284796
Urea (SC)	8.133333	.2643563	30.77	0.000	7.615205	8.651462

Marchenko (2006) shows more examples of fitting other experimental designs using linear mixed-effects models.

◁

▷ Example 3: Binomial counts

We use the data taken from Agresti (2013, 219) on graduate school applications to the 23 departments within the College of Liberal Arts and Sciences at the University of Florida during the 1997–1998 academic year. The dataset contains the department ID (`department`), the number of applications (`napplied`), and the number of students admitted (`nadmitted`) cross-classified by gender (`female`).

```
. use https://www.stata-press.com/data/r19/admissions, clear
(Graduate school admissions data)
. describe
Contains data from https://www.stata-press.com/data/r19/admissions.dta
 Observations:            46                  Graduate school admissions data
    Variables:             4                  25 Feb 2024 09:28
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
department	byte	%8.0g	dept	Department ID
nadmitted	byte	%8.0g		Number of admissions
napplied	int	%9.0g		Number of applications
female	byte	%8.0g		1 if female; 0 if male

```
Sorted by:
```

We wish to investigate whether admission decisions are independent of gender. Given department and gender, the probability of admission follows a binomial model, that is, $\Pr(Y_{ij} = y_{ij}) = \text{Binomial}(n_{ij}, \pi_{ij})$, where $i = \{0, 1\}$ and $j = 1, \ldots, 23$. We fit a mixed-effects binomial logistic model with a random intercept at the department level.

```
. melogit nadmitted female || department:, binomial(napplied) or
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -302.47786
Iteration 1:  Log likelihood = -300.00004
Iteration 2:  Log likelihood = -299.99934
Iteration 3:  Log likelihood = -299.99934
Refining starting values:
Grid node 0:  Log likelihood = -145.08843
Fitting full model:
Iteration 0:  Log likelihood = -145.08843
Iteration 1:  Log likelihood =  -140.8514
Iteration 2:  Log likelihood = -140.61709
Iteration 3:  Log likelihood = -140.61628
Iteration 4:  Log likelihood = -140.61628
Mixed-effects logistic regression               Number of obs     =         46
Binomial variable: napplied
Group variable: department                      Number of groups  =         23
                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(1)      =       2.14
Log likelihood = -140.61628                     Prob > chi2       =     0.1435
```

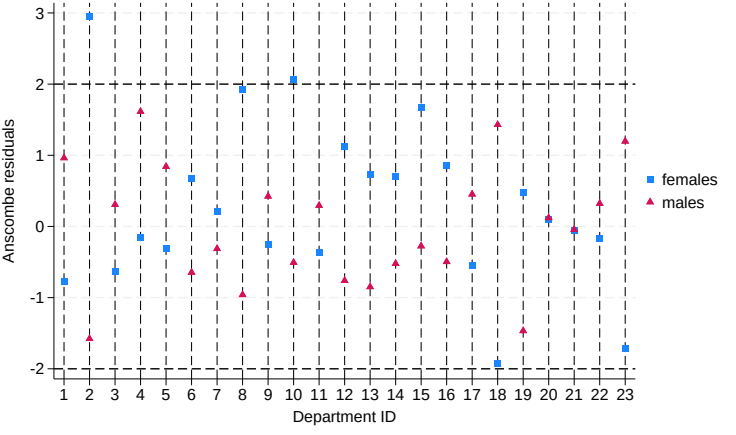
nadmitted	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
female	1.176898	.1310535	1.46	0.144	.9461357	1.463944
_cons	.7907009	.2057191	-0.90	0.367	.4748457	1.316655
department						
var(_cons)	1.345383	.460702			.6876497	2.632234

```
Note: Estimates are transformed only in the first equation to odds ratios.
Note: _cons estimates baseline odds (conditional on zero random effects).
LR test vs. logistic model: chibar2(01) = 318.77      Prob >= chibar2 = 0.0000
```

The odds of being admitted are higher for females than males but without statistical significance. The estimate of $\hat{\sigma}_u^2$ is 1.35 with the standard error of 0.46. An LR test comparing the model with the one-level binomial regression model favors the random-intercept model, indicating that there is a significant variation in the number of admissions between departments.

We can further assess the model fit by performing a residual analysis. For example, here we predict and plot Anscombe residuals.

```
. predict anscres, anscombe
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
. twoway (scatter anscres department if female, msymbol(S))
> (scatter anscres department if !female, msymbol(T)),
> yline(-2 2) xline(1/23, lwidth(vvthin) lpattern(dash))
> xlabel(1/23) legend(label(1 "females") label(2 "males"))
```



Anscombe residuals are constructed to be approximately normally distributed, thus residuals that are above two in absolute value are usually considered outliers. In the graph above, the residual for female admissions in department 2 is a clear outlier, suggesting a poor fit for that particular observation; see [ME] **meglm postestimation** for more information about Anscombe residuals and other model diagnostics tools.

◁

Covariance structures

▷ Example 4: Growth-curve model with correlated random effects

Here we extend the model from example 1 of [ME] **me** to allow for a random slope on `week` and an unstructured covariance structure between the random intercept and the random slope on `week`.

```
. use https://www.stata-press.com/data/r19/pig, clear
(Longitudinal analysis of pig weights)
. mixed weight week || id: week, covariance(unstructured)
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -868.96185
Iteration 1:  Log likelihood = -868.96185
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =    4649.17
Log likelihood = -868.96185                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0910745	68.18	0.000	6.031393	6.388399
_cons	19.35561	.3996387	48.43	0.000	18.57234	20.13889

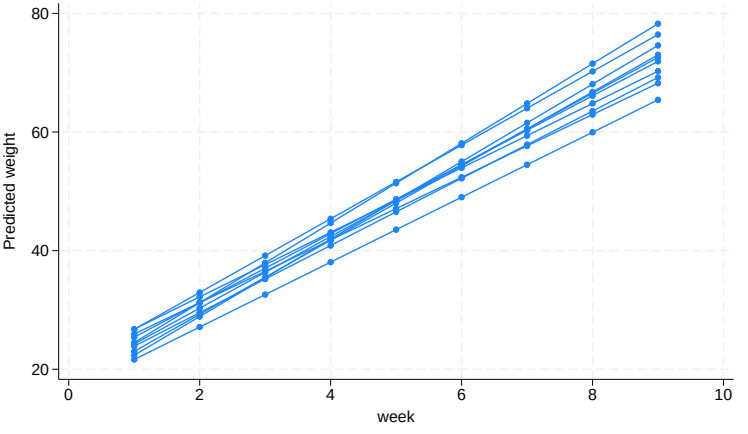
Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Unstructured				
var(week)	.3715251	.0812958	.2419532	.570486
var(_cons)	6.823363	1.566194	4.351297	10.69986
cov(week,_cons)	-.0984378	.2545767	-.5973991	.4005234
var(Residual)	1.596829	.123198	1.372735	1.857505

```
LR test vs. linear model: chi2(3) = 764.58                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

The `unstructured` covariance structure allows for correlation between the random effects. Other covariance structures supported by `mixed`, besides the default `independent`, include `identity` and `exchangeable`; see [ME] **mixed** for details. You can also specify multiple random-effects equations at the same level, in which case the covariance types can be combined to form more complex blocked-diagonal covariance structures; see example 5 below.

We can predict the fitted values and plot the estimated regression line for each of the pigs. The fitted values are based on both the fixed and the random effects.

```
. predict wgt_hat, fitted
. twoway connected wgt_hat week if id<=10, connect(L) ytitle("Predicted weight")
```



Example 5: Blocked-diagonal covariance structures

In this example, we fit a logistic mixed-effects model with a blocked-diagonal covariance structure of random effects.

We use the data from the 1989 Bangladesh fertility survey (Huq and Cleland 1990), which polled 1,934 Bangladeshi women on their use of contraception. The women sampled were from 60 districts, identified by the variable `district`. Each district contained either urban or rural areas (variable `urban`) or both. The variable `c_use` is the binary response, with a value of 1 indicating contraceptive use. Other covariates include mean-centered `age` and a factor variable for the number of `children`. Below we fit a standard logistic regression model amended to have random coefficients for the `children` factor variable and an overall district random intercept.

```
. use https://www.stata-press.com/data/r19/bangladesh, clear
(Bangladesh Fertility Survey, 1989)
```

```
. melogit c_use i.urban age i.children
> || district: i.children, cov(exchangeable)
> || district:, or nolog baselevel nofvlabel
```

```
Mixed-effects logistic regression               Number of obs     =      1,934
Group variable: district                        Number of groups  =         60

                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118

Integration method: mvaghermite                 Integration pts.  =          7

                                                Wald chi2(5)      =     100.01
Log likelihood = -1206.2397                     Prob > chi2       =     0.0000
 ( 1)  [/]var(1.children[district]) - [/]var(3.children[district]) = 0
 ( 2)  [/]cov(1.children[district],2.children[district]) -
       [/]cov(2.children[district],3.children[district]) = 0
 ( 3)  [/]cov(1.children[district],3.children[district]) -
       [/]cov(2.children[district],3.children[district]) = 0
 ( 4)  [/]var(2.children[district]) - [/]var(3.children[district]) = 0
```

c_use	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
urban						
0	1	(constrained)				
1	2.105163	.2546604	6.15	0.000	1.660796	2.668426
age	.9735765	.0077461	-3.37	0.001	.9585122	.9888775
children						
0	1	(constrained)				
1	2.992596	.502149	6.53	0.000	2.153867	4.157931
2	3.879345	.7094125	7.41	0.000	2.710815	5.551584
3	3.774627	.7055812	7.11	0.000	2.616744	5.444863
_cons	.1859471	.0274813	-11.38	0.000	.1391841	.2484214
district						
var(1.children)	.0841518	.0880698			.0108201	.654479
var(2.children)	.0841518	.0880698			.0108201	.654479
var(3.children)	.0841518	.0880698			.0108201	.654479
var(_cons)	.1870273	.0787274			.0819596	.426786
district						
cov(1.children,2.children)	.0616875	.0844681	0.73	0.465	-.1038669	.2272419
cov(1.children,3.children)	.0616875	.0844681	0.73	0.465	-.1038669	.2272419
cov(2.children,3.children)	.0616875	.0844681	0.73	0.465	-.1038669	.2272419

```
Note: Estimates are transformed only in the first equation to odds ratios.
Note: _cons estimates baseline odds (conditional on zero random effects).
LR test vs. logistic model: chi2(3) = 44.57                 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

The fixed effects can be interpreted just as you would the output from `logit`. Urban women have roughly double the odds of using contraception as compared with their rural counterparts. Having any number of children will increase the odds from three- to fourfold when compared with the base category of no children. Contraceptive use also decreases with age.

Because we specified `cov(exchangeable)`, the estimated variances for the `children` factor levels are constrained to be the same, and the estimated covariances for the `children` factor levels are constrained to be the same. More complex covariance structures with constraints can be specified using `covariance(pattern())` and `covariance(fixed())`; see example 6 below.

◁

▷ Example 6: Meta analysis

In this example, we present a mixed-effects model for meta analysis of clinical trials. The term "meta-analysis" refers to a statistical analysis that involves summary data from similar but independent studies. The model can be fit directly with the `meta` suite of commands; however, in this example, we will fit it with `meglm` to illustrate the use of constraints.

Turner et al. (2000) performed a study of nine clinical trials examining the effect of taking diuretics during pregnancy on the risk of pre-eclampsia. The summary data consist of the log odds-ratio (variable `lnor`) estimated from each study, and the corresponding estimated variance (variable `var`). The square root of the variance is stored in the variable `std` and the trial identifier is stored in the variable `trial`.

```
. use https://www.stata-press.com/data/r19/diuretics
(Meta analysis of clinical trials studying diuretics and pre-eclampsia)
. list
```

	trial	lnor	var	std
1.	1	.04	.16	.4
2.	2	-.92	.12	.3464102
3.	3	-1.12	.18	.4242641
4.	4	-1.47	.3	.5477226
5.	5	-1.39	.11	.3316625
6.	6	-.3	.01	.1
7.	7	-.26	.12	.3464102
8.	8	1.09	.69	.8306624
9.	9	.14	.07	.2645751

In a random-effects modeling of summary data, the observed log odds-ratios are treated as a continuous outcome and assumed to be normally distributed, and the true treatment effect varies randomly among the trials. The random-effects model can be written as

$$y_i \sim N(\theta + \nu_i, \sigma_i^2)$$
$$\nu_i \sim N(0, \tau^2)$$

where y_i is the observed treatment effect corresponding to the ith study, $\theta + \nu_i$ is the true treatment effect, σ_i^2 is the variance of the observed treatment effect, and τ is the between-trial variance component. Our aim is to estimate θ, the global mean.

Notice that the responses y_i do not provide enough information to estimate this model, because we cannot estimate the group-level variance component from a dataset that contains one observation per group. However, we already have estimates for the σ_i's, therefore we can constrain each σ_i to be equal to its estimated value, which will allow us to estimate θ and τ. We use `meglm` to estimate this model because the `mixed` command does not support constraints.

In `meglm`, one way to constrain a group of individual variances to specific values is by using the fixed covariance structure (an alternative way is to define each constraint individually with the `constraint` command and specify them in the `constraints()` option). The `covariance(fixed())` option requires a Stata matrix defining the constraints, thus we first create matrix `f` with the values of σ_i, stored in variable `var`, on the main diagonal. We will use this matrix to constrain the variances.

```
. mkmat var, mat(f)
. matrix f = diag(f)
```

In the random-effects equation part, we need to specify nine random slopes, one for each trial. Because random-effects equations support factor variables (see **[U] 11.4.3 Factor variables**), we can use the `ibn.trial` notation. Because the model is computationally demanding, we use Laplacian approximation instead of the default mean-variance adaptive quadrature; see *Computation time and the Laplacian approximation* above for details.

```
. meglm lnor || _all: ibn.trial, nocons cov(fixed(f)) intm(laplace) nocnsreport
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -10.643432
Iteration 1:  Log likelihood = -10.643432
Refining starting values:
Grid node 0:  Log likelihood = -10.205455
Fitting full model:
Iteration 0:  Log likelihood = -10.205455
Iteration 1:  Log likelihood = -9.4851561  (backed up)
Iteration 2:  Log likelihood = -9.4587068
Iteration 3:  Log likelihood = -9.4552982
Iteration 4:  Log likelihood = -9.4552759
Iteration 5:  Log likelihood = -9.4552759
Mixed-effects GLM                               Number of obs     =          9
Family: Gaussian
Link:   Identity
Group variable: _all                            Number of groups  =          1
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
Integration method: laplace
                                                Wald chi2(0)      =          .
Log likelihood = -9.4552759                     Prob > chi2       =          .
```

lnor	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_cons	-.5166151	.2059448	-2.51	0.012	-.9202594	-.1129707
_all						
var(1.trial)	.16	(constrained)				
var(2.trial)	.12	(constrained)				
var(3.trial)	.18	(constrained)				
var(4.trial)	.3	(constrained)				
var(5.trial)	.11	(constrained)				
var(6.trial)	.01	(constrained)				
var(7.trial)	.12	(constrained)				
var(8.trial)	.69	(constrained)				
var(9.trial)	.07	(constrained)				
var(e.lnor)	.2377469	.1950926			.0476023	1.187413

We estimate $\widehat{\theta} = -0.52$, which agrees with the estimate reported by Turner et al. (2000).

We can fit the above model in a more efficient way. We can consider the trials as nine independent random variables, each with variance unity, and each being multiplied by a different standard error. To accomplish this, we treat `trial` as a random-effects level, use the standard deviations of the log odds-ratios as a random covariate at the `trial` level, and constrain the variance component of `trial` to unity.

```
. constraint 1 _b[/var(std[trial])] = 1
. meglm lnor || trial: std, nocons constraints(1)
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -10.643432
Iteration 1:  Log likelihood = -10.643432
Refining starting values:
Grid node 0:  Log likelihood = -10.205455
Fitting full model:
Iteration 0:  Log likelihood = -10.205455
Iteration 1:  Log likelihood = -9.4851164  (backed up)
Iteration 2:  Log likelihood =   -9.45869
Iteration 3:  Log likelihood = -9.4552794
Iteration 4:  Log likelihood = -9.4552759
Iteration 5:  Log likelihood = -9.4552759
Mixed-effects GLM                               Number of obs     =          9
Family: Gaussian
Link:   Identity
Group variable: trial                           Number of groups  =          9
                                                Obs per group:
                                                              min =          1
                                                              avg =        1.0
                                                              max =          1
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(0)      =          .
Log likelihood = -9.4552759                     Prob > chi2       =          .
 ( 1)  [/]var(std[trial]) = 1
```

lnor	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_cons	-.5166151	.2059448	-2.51	0.012	-.9202594	-.1129708
trial						
var(std)	1	(constrained)				
var(e.lnor)	.2377469	.1950926			.0476023	1.187413

The results are the same, but this model took a fraction of the time compared with the less efficient specification.

◁

Three-level models

The methods we have discussed so far extend from two-level models to models with three or more levels with nested random effects. By "nested", we mean that the random effects shared within lower-level subgroups are unique to the upper-level groups. For example, assuming that classroom effects would be nested within schools would be natural, because classrooms are unique to schools. Below we illustrate a three-level mixed-effects ordered probit model.

▷ Example 7: Three-level ordinal response model

In this example, we fit a three-level ordered probit model. The data are from the Television, School, and Family Smoking Prevention and Cessation Project (Flay et al. 1988; Rabe-Hesketh and Skrondal 2022, chap. 11), where schools were randomly assigned into one of four groups defined by two treatment variables. Students within each school are nested in classes, and classes are nested in schools. The

dependent variable is the tobacco and health knowledge (THK) scale score collapsed into four ordered categories. We regress the outcome on the treatment variables and their interaction and control for the pretreatment score.

```
. use https://www.stata-press.com/data/r19/tvsfpors, clear
(Television, School, and Family Project)
. meoprobit thk prethk cc##tv || school: || class:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood = -2127.8111
Iteration 2:  Log likelihood = -2127.7612
Iteration 3:  Log likelihood = -2127.7612
Refining starting values:
Grid node 0:  Log likelihood = -2195.6424
Fitting full model:
Iteration 0:  Log likelihood = -2195.6424  (not concave)
Iteration 1:  Log likelihood = -2167.9576  (not concave)
Iteration 2:  Log likelihood = -2140.2644  (not concave)
Iteration 3:  Log likelihood = -2128.6948  (not concave)
Iteration 4:  Log likelihood = -2119.9225
Iteration 5:  Log likelihood = -2117.0947
Iteration 6:  Log likelihood = -2116.7004
Iteration 7:  Log likelihood = -2116.6981
Iteration 8:  Log likelihood = -2116.6981
Mixed-effects oprobit regression                Number of obs     =      1,600
        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
         school |         28         18       57.1        137
          class |        135          1       11.9         28
------------------------------------------------------------------------------
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(4)      =     124.20
Log likelihood = -2116.6981                     Prob > chi2       =     0.0000
------------------------------------------------------------------------------
         thk | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
      prethk |    .238841   .0231446    10.32   0.000     .1934784    .2842036
        1.cc |   .5254813   .1285816     4.09   0.000     .2734659    .7774967
        1.tv |   .1455573   .1255827     1.16   0.246    -.1005803    .3916949
             |
       cc#tv |
        1 1 |  -.2426203   .1811999    -1.34   0.181    -.5977656    .1125251
-------------+----------------------------------------------------------------
       /cut1 |   -.074617   .1029791                    -.2764523    .1272184
       /cut2 |   .6863046   .1034813                     .4834849    .8891242
       /cut3 |   1.413686   .1064889                     1.204972    1.622401
-------------+----------------------------------------------------------------
school       |
   var(_cons) |   .0186456   .0160226                     .0034604    .1004695
-------------+----------------------------------------------------------------
school>class |
   var(_cons) |   .0519974   .0224014                     .0223496    .1209745
------------------------------------------------------------------------------
LR test vs. oprobit model: chi2(2) = 22.13                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Notes:

1. Our model now has two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `school` level (level three), and the second is a random intercept at the `class` level (level two). The order in which these are specified (from left to right) is significant—`meoprobit` assumes that `class` is nested within `school`.
2. The information on groups is now displayed as a table, with one row for each grouping. You can suppress this table with the `nogroup` or the `noheader` option, which will also suppress the rest of the header.
3. The variance-component estimates are now organized and labeled according to level. The variance component for `class` is labeled `school>class` to emphasize that classes are nested within schools.

◁

The above extends to models with more than two levels of nesting in the obvious manner, by adding more random-effects equations, each separated by `||`. The order of nesting goes from left to right as the groups go from biggest (highest level) to smallest (lowest level).

Crossed-effects models

Not all mixed-effects models contain nested levels of random effects.

▷ Example 8: Crossed random effects

Returning to our longitudinal analysis of pig weights, suppose that we wish to fit

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_i + v_j + \epsilon_{ij} \tag{8}$$

for the $i = 1, \ldots, 9$ weeks and $j = 1, \ldots, 48$ pigs and

$$u_i \sim N(0, \sigma_u^2); \quad v_j \sim N(0, \sigma_v^2); \quad \epsilon_{ij} \sim N(0, \sigma_\epsilon^2)$$

all independently. That is, we assume an overall population-average growth curve $\beta_0 + \beta_1 \texttt{week}$ and a random pig-specific shift. In other words, the effect due to `week`, u_i, is systematic to that week and common to all pigs. The rationale behind (8) could be that, assuming that the pigs were measured contemporaneously, we might be concerned that week-specific random factors such as weather and feeding patterns had significant systematic effects on all pigs.

Model (8) is an example of a two-way crossed-effects model, with the pig effects v_j being crossed with the week effects u_i. One way to fit such models is to consider all the data as one big cluster, and treat u_i and v_j as a series of $9 + 48 = 57$ random coefficients on indicator variables for `week` and `pig`. The random effects $\mathbf{u}$ and the variance components $\mathbf{G}$ are now represented as

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_9 \\ v_1 \\ \vdots \\ v_{48} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{G}); \quad \mathbf{G} = \begin{bmatrix} \sigma_u^2 \mathbf{I}_9 & \mathbf{0} \\ \mathbf{0} & \sigma_v^2 \mathbf{I}_{48} \end{bmatrix}$$

Because $\mathbf{G}$ is block diagonal, it can be represented as repeated-level equations. All we need is an ID variable to identify all the observations as one big group and a way to tell mixed-effects commands to treat `week` and `pig` as crossed-effects factor variables (or equivalently, as two sets of overparameterized indicator variables identifying weeks and pigs, respectively). The mixed-effects commands support the special group designation `_all` for the former and the `R.`*varname* notation for the latter.

```
. use https://www.stata-press.com/data/r19/pig
(Longitudinal analysis of pig weights)
. mixed weight week || _all: R.id || _all: R.week
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood =  -1013.824
Iteration 1:  Log likelihood =  -1013.824
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: _all                            Number of groups  =          1
                                                Obs per group:
                                                              min =        432
                                                              avg =      432.0
                                                              max =        432
                                                Wald chi2(1)      =   13258.28
Log likelihood =  -1013.824                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0539313	115.14	0.000	6.104192	6.315599
_cons	19.35561	.6333982	30.56	0.000	18.11418	20.59705

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
_all: Identity				
var(R.id)	14.83623	3.126142	9.816733	22.42231
_all: Identity				
var(R.week)	.0849874	.0868856	.0114588	.6303302
var(Residual)	4.297328	.3134404	3.724888	4.957741

```
LR test vs. linear model: chi2(2) = 474.85                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We estimate $\widehat{\sigma}_u^2 = 0.08$ and $\widehat{\sigma}_v^2 = 14.84$.

The `R.`*varname* notation is equivalent to giving a list of overparameterized (none dropped) indicator variables for use in a random-effects specification. When you use `R.`*varname*, mixed-effects commands handle the calculations internally rather than creating the indicators in the data. Because the set of indicators is overparameterized, `R.`*varname* implies `noconstant`.

Note that the column dimension of our random-effects design is 57. Computation time and memory requirements grow (roughly) quadratically with the dimension of the random effects. As a result, fitting such crossed-effects models is feasible only when the total column dimension is small to moderate. For this reason, mixed-effects commands use the Laplacian approximation as the default estimation method for crossed-effects models; see *Computation time and the Laplacian approximation* above for more details.

It is often possible to rewrite a mixed-effects model in a way that is more computationally efficient. For example, we can treat pigs as nested within the `_all` group, yielding the equivalent and more efficient (total column dimension 10) way to fit (8):

```
. mixed weight week || _all: R.week || id:
```

The results of both estimations are identical, but the latter specification, organized at the cluster (pig) level with random-effects dimension 1 (a random intercept) is much more computationally efficient. Whereas with the first form we are limited in how many pigs we can analyze, there is no such limitation with the second form.

All the mixed-effects commands—except `mixed`—automatically attempt to recast the less efficient model specification into a more efficient one. However, this automatic conversion may not be sufficient for some complicated mixed-effects specifications, especially if both crossed and nested effects are involved. Therefore, we strongly encourage you to always specify the more efficient syntax; see Rabe-Hesketh and Skrondal (2022) and Marchenko (2006) for additional techniques to make calculations more efficient in more complex mixed-effects models.

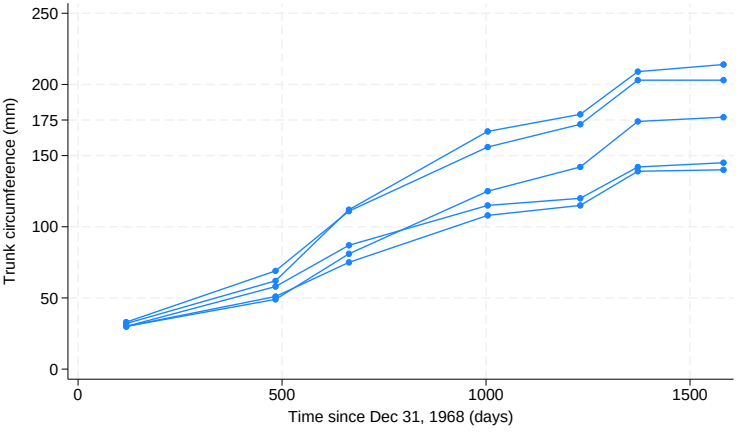
◁

Nonlinear models

NLME models are popular in population pharmacokinetics, bioassays, studies of biological and agricultural growth processes, and other applications, where the mean function is a nonlinear function of fixed and random effects. *Remarks and examples* of [ME] **menl** provide many examples of fitting different NLME models by using `menl`, including a pharmacokinetics model in example 15. Here we consider simple data from Draper and Smith (1998) that contain trunk circumference (in mm) of five different orange trees measured over seven different time points.

Let's plot our data first.

```
. use https://www.stata-press.com/data/r19/orange
(Growth of orange trees (Draper and Smith, 1998))
. twoway scatter circumf age, connect(L) ylabel(#6 175)
```



Consider the following nonlinear growth model for these data,

$$\texttt{circumf}_{ij} = \frac{\beta_1}{1 + \exp\left\{-\left(\texttt{age}_{ij} - \beta_2\right)/\beta_3\right\}} + \epsilon_{ij}$$

where ϵ_{ij}'s are i.i.d. N(0, σ^2_ϵ). In this model, β_1 can be interpreted as the average asymptotic trunk circumference of trees as $\texttt{age}_{ij} \to \infty$. We can crudely estimate it as the average of the trunk circumference values at the last observed time point, which for these data is roughly 175 mm. β_2 is the age at which a tree attains half of the average asymptotic trunk circumference β_1; that is, if we set $\texttt{age}_{ij} = \beta_2$, then $E(\texttt{circumf}_{ij}) = 0.5\beta_1$. β_3 is a scale parameter that represents the number of days it takes for a tree to grow from 50% to about 73% of the average asymptotic trunk circumference. That is, if we set $\texttt{age} = t_{0.73} = \beta_2 + \beta_3$, then $E(\texttt{circumf}_{ij}) = \beta_1/\{1 + \exp(-1)\} = 0.73\beta_1$ and then $\beta_3 = t_{0.73} - \beta_2$.

The above model can be easily fit by using, for example, `nl`; see [R] **nl**. However, if we study the graph more carefully, we will notice that there is an increasing variability in the trunk circumferences of trees as they approach their limiting age. So it may be more reasonable to allow β_1 to vary between trees,

$$\texttt{circumf}_{ij} = \frac{\beta_1 + u_{1j}}{1 + \exp\left\{-\left(\texttt{age}_{ij} - \beta_2\right)/\beta_3\right\}} + \epsilon_{ij} \tag{9}$$

where u_{1j}'s are i.i.d. $N(0, \sigma^2_{u_1})$. We use `menl` to fit this model.

The specification of NLME models in `menl` is fairly straightforward. Following the dependent variable and the equality sign (`=`), we specify the expression for the mean function as a usual Stata expression but with parameters and random effects enclosed in curly braces (`{}`).

```
. menl circumf = ({b1}+{U1[tree]})/(1+exp(-(age-{b2})/{b3}))
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -131.58458
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
Linearization log likelihood = -131.58458
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b1	191.049	16.15403	11.83	0.000	159.3877	222.7103
/b2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/b3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Identity				
var(U1)	991.1514	639.4637	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

In the above specification, we used `{U1[tree]}` to include random intercepts at the `tree` level in our model. `U1` is the name or label associated with these random intercepts.

The output of `menl` is similar to that of `mixed`—the header information is displayed first, fixed-effects parameter estimates are displayed in the first or the fixed-effects parameter table, and the estimates of variance components are displayed in the second or the random-effects parameter table.

The header information is similar to that of `mixed`, but unlike `mixed`, `menl` in general does not report a model χ^2 statistic in the header because a test of the joint significance of all fixed-effects parameters (except the constant term) may not be relevant in a nonlinear model. `menl` also reports the so-called linearization log likelihood. `menl` uses the linearization method of Lindstrom and Bates (1990), with extensions from Pinheiro and Bates (1995), for estimation. This method is based on the approximation of the NLME model by an LME model, in which a first-order Taylor-series approximation is used to linearize the nonlinear mean function with respect to fixed and random effects; see *Introduction* and *Methods and formulas* in [ME] **menl** for details. The linearization log likelihood is the log likelihood of this approximating LME model. We can use this log likelihood for model comparison of different NLME models and to form likelihood-ratio tests, but note that this is not the log likelihood of the corresponding NLME model. Depending on the accuracy of the approximation, the linearization log likelihood may be close to the true NLME log likelihood.

As part of Stata's standard estimation output, `menl` reports z tests against zeros for the estimated fixed-effects parameters. Testing a parameter against zero may or may not be of interest, or may not even be appropriate, in a nonlinear model. In our example, `{b3}` is the denominator of a fraction, so the test of `{b3}` against zero may not be feasible in this model. Instead, we may be interested in testing `{b3}` against, for example, 300, which would correspond to testing whether the average trunk circumference

of orange trees increases from 50% to 73% of its asymptotic value in 300 days. We can perform this test by using, for instance, the `test` command; see [R] **test**. As a side note, setting $\beta_3 = 0$ in (9) results in a simple random-intercept model, in a limiting sense.

From the random-effects table, the variability in limiting growth β_1 between trees, labeled as `var(U1)`, is statistically significant in this model with an estimate of 991 (mm^2) and a 95% CI of [280, 3510].

We can rewrite (9) as a two-stage model,

$$\texttt{circumf}_{ij} = \frac{\phi_{1j}}{1 + \exp\left\{-\left(\texttt{age}_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij} \tag{10}$$

where the stage 2 specification is

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\end{bmatrix} = \begin{bmatrix}\beta_1 + u_{1j}\\ \beta_2\\ \beta_3\end{bmatrix} \tag{11}$$

The model defined by (10) and (11) is the same as that defined by (9) but with a different parameterization.

In `menl`, we can accommodate this two-stage formulation with the `define()` option. For example, we can fit the two-stage model defined by (10) and (11) as follows:

```
. menl circumf = {phi1:}/(1+exp(-(age-{b2})/{b3})), define(phi1: {b1}+{U1[tree]})
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -131.58458
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
Linearization log likelihood = -131.58458
        phi1: {b1}+{U1[tree]}
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b1	191.049	16.15403	11.83	0.000	159.3877	222.7103
/b2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/b3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Identity				
var(U1)	991.1514	639.4637	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

The results are identical to the previous model. Here we defined a substitutable expression `phi1` in the `define()` option as a function of `{b1}` and `{U1[tree]}` and included it in our main expression as `{phi1:}`. Including a colon (`:`) in `{phi1:}` is important to notify `menl` that it is a substitutable expression rather than a simple scalar parameter `{phi1}`.

In general, we can accommodate multistage formulations by using the `define()` option repeatedly.

More conveniently, we can use a linear-combination specification (see *Linear combinations* in [ME] **menl**) within the `define()` option to define the linear combination `{b1}+{U1[tree]}`.

```
. menl circumf = {phi1:}/(1+exp(-(age-{b2})/{b3})), define(phi1: U1[tree], xb)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -131.58458
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
Linearization log likelihood = -131.58458
        phi1: U1[tree], xb
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	191.049	16.15403	11.83	0.000	159.3877	222.7103
/b2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/b3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Identity				
var(U1)	991.1514	639.4637	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

The `{phi1: U1[tree], xb}` specification used in the `define()` option, but without curly braces, creates a linear combination named `phi1` that contains a constant `{phi1:_cons}` and random intercepts `{U1}` at the `tree` level. In the linear-combination specification, the constant is included automatically unless you specify the `noconstant` option such as `{phi1: U1[tree], xb noconstant}`. Also, you do not specify curly braces around random effects within the linear-combination specification. If we had covariates, say, `x1` and `x2`, that we also wanted to include in the linear combination, we would have used `{phi1: x1 x2 U1[tree]}`. Notice that we did not specify the `xb` option in the previous linear combination. When a linear combination contains more than one term, this option is implied. When a linear combination contains only one term, such as in `{phi1: U1[tree], xb}`, the `xb` option must be specified to request that `menl` treat the specification as a linear combination instead of a scalar parameter; see *Random-effects substitutable expressions* in [ME] **menl** for details.

Instead of using `define()`, we could have similarly specified the linear combination directly in the main expression:

```
. menl circumf = {phi1: U1[tree], xb}/(1+exp(-(age-{b2})/{b3}))
  (output omitted)
```

However, by using the `define()` option, we simplified the look of the main equation.

We can extend the stage 2 specification (11) to allow, for example, β_2 to vary across trees by including random intercepts at the tree level for ϕ_{2j},

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 + u_{1j} \\ \beta_2 + u_{2j} \\ \beta_3 \end{bmatrix}$$

We can then fit the corresponding model by using `menl` as follows:

```
. menl circumf = {phi1:}/(1+exp(-(age-{phi2:})/{b3})),
> define(phi1: U1[tree], xb) define(phi2: U2[tree], xb)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -131.60539
Iteration 2:  Linearization log likelihood =  -131.5827
Iteration 3:  Linearization log likelihood =  -131.5805
Iteration 4:  Linearization log likelihood = -131.58027
Iteration 5:  Linearization log likelihood = -131.58026

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7

Linearization log likelihood = -131.58026
        phi1: U1[tree], xb
        phi2: U2[tree], xb
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	190.5939	16.211	11.76	0.000	158.8209	222.3669
phi2						
_cons	719.6027	35.77597	20.11	0.000	649.4831	789.7223
/b3	342.0794	26.42036	12.95	0.000	290.2965	393.8624

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Independent				
var(U1)	1012.15	666.2808	278.557	3677.698
var(U2)	503.2308	2401.323	.043651	5801505
var(Residual)	59.27073	18.21298	32.45482	108.2434

The large standard error for the estimate of the variance component `var(U2)` suggests that our model is overparameterized—a common problem when fitting NLME models. We could verify this, for instance, by computing information criteria ([R] **estimates stats**) or by performing a likelihood-ratio test ([R] **lrtest**).

By default, menl assumes an independent covariance structure for the random effects such as U1 and U2 in our example. We can specify, for example, an unstructured model by using the covariance() option. We demonstrate this only for illustration, given that our simpler model that assumed independence between U1 and U2 was already overparameterized.

```
. menl circumf = {phi1:}/(1+exp(-(age-{phi2:})/{b3})),
> define(phi1: U1[tree], xb) define(phi2: U2[tree], xb)
> covariance(U1 U2, unstructured)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -130.90452
Iteration 2:  Linearization log likelihood = -130.90205
Iteration 3:  Linearization log likelihood = -130.90177
Iteration 4:  Linearization log likelihood = -130.90177

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7

Linearization log likelihood = -130.90177
        phi1: U1[tree], xb
        phi2: U2[tree], xb
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	189.8349	17.20035	11.04	0.000	156.1228	223.5469
phi2						
_cons	709.5333	37.24229	19.05	0.000	636.5397	782.5268
/b3	340.4731	25.52176	13.34	0.000	290.4514	390.4948

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Unstructured				
var(U1)	1180.097	775.0821	325.7263	4275.459
var(U2)	1469.879	2777.13	36.2289	59635.9
cov(U1,U2)	1015.504	1124.568	-1188.609	3219.617
var(Residual)	56.07332	16.20294	31.82682	98.79144

In menl, we need to list the names of the random effects in the covariance() option for which we want to specify a covariance structure other than the independent one used by default.

In our example, parameters ϕ_{1j} and ϕ_{2j} were modeled as linear functions of random effects and parameters β_1 and β_2. The relationship does not have to be linear; see example 15 in [ME] **menl**.

This example has a small number of trees or clusters, so REML estimation would have been more appropriate. We could have obtained REML estimates in our examples by specifying the reml option with menl.

See [ME] **menl** for more examples of and details about the menl command.

Acknowledgments

We are indebted to Sophia Rabe-Hesketh of the University of California, Berkeley, and coauthor of the Stata Press book *Multilevel and Longitudinal Modeling Using Stata*; Anders Skrondal of the University of Oslo and the Norwegian Institute of Public Health, and coauthor of the Stata Press book *Multilevel and Longitudinal Modeling Using Stata*; and Andrew Pickles of King's College London for their extensive body of work in Stata, both previous and ongoing, in this area.

References

Agresti, A. 2013. *Categorical Data Analysis*. 3rd ed. Hoboken, NJ: Wiley.

Bates, D. M., and J. C. Pinheiro. 1998. "Computational methods for multilevel modelling". In *Technical Memorandum BL0112140-980226-01TM*. Murray Hill, NJ: Bell Labs, Lucent Technologies.

Breslow, N. E., and D. G. Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association* 88: 9–25. https://doi.org/10.2307/2290687.

Davidian, M., and D. M. Giltinan. 1995. *Nonlinear Models for Repeated Measurement Data*. Boca Raton, FL: Chapman and Hall/CRC. https://doi.org/10.1201/9780203745502.

———. 2003. Nonlinear models for repeated measurement data: An overview and update. *Journal of Agricultural, Biological, and Environmental Statistics* 8: 387–419. https://doi.org/10.1198/1085711032697.

De Boeck, P., and M. Wilson, eds. 2004. *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*. New York: Springer. https://doi.org/10.1007/978-1-4757-3990-9.

Demidenko, E. 2013. *Mixed Models: Theory and Applications with R*. 2nd ed. Hoboken, NJ: Wiley.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B ser., 39: 1–38. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Diggle, P. J., P. J. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.

Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley. https://doi.org/10.1002/9781118625590.

Fitzmaurice, G. M., M. Davidian, G. Verbeke, and G. Molenberghs, eds. 2009. *Longitudinal Data Analysis*. Boca Raton, FL: Chapman and Hall/CRC. https://doi.org/10.1201/9781420011579.

Flay, B. R., B. R. Brannon, C. A. Johnson, W. B. Hansen, A. L. Ulene, D. A. Whitney-Saltiel, L. R. Gleason, S. Sussman, M. D. Gavin, K. M. Glowacz, D. F. Sobol, and D. C. Spiegel. 1988. The television, school, and family smoking cessation and prevention project: I. Theoretical basis and program development. *Preventive Medicine* 17: 585–607. https://doi.org/10.1016/0091-7435(88)90053-9.

Gelman, A., and J. Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511790942.

Gutierrez, R. G., S. L. Carter, and D. M. Drukker. 2001. sg160: On boundary-value likelihood-ratio tests. *Stata Technical Bulletin* 60: 15–18. Reprinted in *Stata Technical Bulletin Reprints*, vol. 10, pp. 269–273. College Station, TX: Stata Press.

Hall, B. H., Z. Griliches, and J. A. Hausman. 1986. Patents and R and D: Is there a lag? *International Economic Review* 27: 265–283. https://doi.org/10.2307/2526504.

Hedeker, D., and R. D. Gibbons. 2006. *Longitudinal Data Analysis*. Hoboken, NJ: Wiley.

Henderson, C. R. 1953. Estimation of variance and covariance components. *Biometrics* 9: 226–252. https://doi.org/10.2307/3001853.

Huq, N. M., and J. Cleland. 1990. *Bangladesh Fertility Survey 1989 (Main Report)*. National Institute of Population Research and Training.

Kuehl, R. O. 2000. *Design of Experiments: Statistical Principles of Research Design and Analysis*. 2nd ed. Belmont, CA: Duxbury.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

LaMotte, L. R. 1973. Quadratic estimation of variance components. *Biometrics* 29: 311–330. https://doi.org/10.2307/2529395.

Lesaffre, E., and B. Spiessens. 2001. On the effect of the number of quadrature points in a logistic random-effects model: An example. *Journal of the Royal Statistical Society*, C ser., 50: 325–335. https://doi.org/10.1111/1467-9876.00237.

Lin, X., and N. E. Breslow. 1996. Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association* 91: 1007–1016. https://doi.org/10.2307/2291720.

Lindstrom, M. J., and D. M. Bates. 1990. Nonlinear mixed effects models for repeated measures data. *Biometrics* 46: 673–687. https://doi.org/10.2307/2532087.

Littell, R. C., G. A. Milliken, W. W. Stroup, R. D. Wolfinger, and O. Schabenberger. 2006. *SAS System for Mixed Models*. 2nd ed. Cary, NC: SAS Institute.

Liu, Q., and D. A. Pierce. 1994. A note on Gauss–Hermite quadrature. *Biometrika* 81: 624–629. https://doi.org/10.2307/2337136.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models*. 2nd ed. Hoboken, NJ: Wiley.

McLachlan, G. J., and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. New York: Dekker.

Ng, E. S.-W., J. R. Carpenter, H. Goldstein, and J. Rasbash. 2006. Estimation in generalised linear mixed models with binary outcomes by simulated maximum likelihood. *Statistical Modelling* 6: 23–42. https://doi.org/10.1191/1471082X06st106oa.

Pinheiro, J. C., and D. M. Bates. 1995. Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics* 4: 12–35. https://doi.org/10.2307/1390625.

———. 2000. *Mixed-Effects Models in S and S-PLUS*. New York: Springer. https://doi.org/10.1007/b98882.

Pinheiro, J. C., and E. C. Chao. 2006. Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models. *Journal of Computational and Graphical Statistics* 15: 58–81. https://doi.org/10.1198/106186006X96962.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Rao, C. R. 1973. *Linear Statistical Inference and Its Applications*. 2nd ed. New York: Wiley. https://doi.org/10.1002/9780470316436.

Raudenbush, S. W., and A. S. Bryk. 2002. *Hierarchical Linear Models: Applications and Data Analysis Methods*. 2nd ed. Thousand Oaks, CA: Sage.

Rodríguez, G., and N. Goldman. 1995. An assessment of estimation procedures for multilevel models with binary responses. *Journal of the Royal Statistical Society*, A ser., 158: 73–89. https://doi.org/10.2307/2983404.

Ruppert, D., M. P. Wand, and R. J. Carroll. 2003. *Semiparametric Regression*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511755453.

Searle, S. R., G. Casella, and C. E. McCulloch. 1992. *Variance Components*. New York: Wiley. https://doi.org/10.1002/9780470316856.

Self, S. G., and K.-Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association* 82: 605–610. https://doi.org/10.2307/2289471.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman and Hall/CRC.

Stram, D. O., and J. W. Lee. 1994. Variance components testing in the longitudinal mixed effects model. *Biometrics* 50: 1171–1177. https://doi.org/10.2307/2533455.

Thompson, W. A., Jr. 1962. The problem of negative estimates of variance components. *Annals of Mathematical Statistics* 33: 273–289. https://doi.org/10.1214/aoms/1177704731.

Tierney, L., and J. B. Kadane. 1986. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association* 81: 82–86. https://doi.org/10.1080/01621459.1986.10478240.

Turner, R. M., R. Z. Omar, M. Yang, H. Goldstein, and S. G. Thompson. 2000. A multilevel model framework for meta-analysis of clinical trials with binary outcomes. *Statistics in Medicine* 19: 3417–3432. https://doi.org/10.1002/1097-0258(20001230)19:24<3417::aid-sim614>3.0.co;2-l.

Tutz, G., and W. Hennevogl. 1996. Random effects in ordinal regression models. *Computational Statistics and Data Analysis* 22: 537–557. https://doi.org/10.1016/0167-9473(96)00004-7.

Vella, F., and M. Verbeek. 1998. Whose wages do unions raise? A dynamic model of unionism and wage rate determination for young men. *Journal of Applied Econometrics* 13: 163–183. https://doi.org/10.1002/(SICI)1099-1255(199803/04)13:2<163::AID-JAE460>3.0.CO;2-Y.

Verbeke, G., and G. Molenberghs. 2000. *Linear Mixed Models for Longitudinal Data*. New York: Springer. https://doi.org/10.1007/978-1-4419-0300-6.

Vonesh, E. F., and V. M. Chinchilli. 1997. *Linear and Nonlinear Models for the Analysis of Repeated Measurements*. New York: Dekker. https://doi.org/10.1201/9781482293272.

Also see

[ME] **Glossary**

estat df — Calculate degrees of freedom for fixed effects

Description Menu for estat Syntax Options
Remarks and examples Stored results Also see

Description

`estat df` is for use after estimation with `mixed`.

`estat df` calculates and displays the degrees of freedom (DF) for each fixed effect using the specified methods. This allows for a comparison of different DF methods. `estat df` can also be used to continue with postestimation using a different DF method without rerunning the model.

Menu for estat

Statistics > Postestimation

Syntax

`estat df` [`,` `method(`*df_methods*`)` `post`[`(`*df_method*`)`] `eim` `oim`]

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`method(`*df_methods*`)` specifies a list of methods to compute DF. The supported methods are `residual`, `repeated`, `anova`, `satterthwaite`, and `kroger`; more than one method may be specified. Methods `satterthwaite` and `kroger` are only available with REML estimation. If option `dfmethod()` was not specified in the most recently fit `mixed` model, then option `method()` is required. See *Small-sample inference for fixed effects* under *Remarks and examples* in [ME] **mixed** for more details.

`post` causes `estat df` to behave like a Stata estimation command. When `post` is specified, `estat df` will post the DF for each fixed effect as well as everything related to the DF computation to `e()` for the method specified in `method()`. Thus, after posting, you could continue to use this DF for other postestimation commands. For example, you could use `test, small` to perform Wald F tests on linear combination of the fixed effects.

`post` may also be specified using the syntax `post(`*df_method*`)`. You must use this syntax if you specify multiple *df_methods* in option `method()`. With this syntax, `estat df` computes the DF using the method specified in `post()` and stores the results in `e()`. Only one computation method may be specified using the syntax `post()`.

The *df_method* specified in `post()` must be one of the DF methods specified in option `method()`. If only one method is specified in option `method()`, then one can simply use `post` to make this DF method active for postestimation and for `mixed` replay.

`eim` specifies that the expected information matrix be used in the DF computation. It can be used only when `method()` contains `kroger` or `satterthwaite`. `eim` is the default.

`oim` specifies that the observed information matrix be used in the DF computation. It can be used only when `method()` contains `kroger` or `satterthwaite`.

Remarks and examples

▷ Example 1: Changing the degrees of freedom method

To illustrate the use of `estat df`, we refit the dental veneer data from example 16 of [ME] **mixed** using the Kenward–Roger method (option `dfmethod(kroger)`) to compute the DF for fixed effects.

```
. use https://www.stata-press.com/data/r19/veneer
(Dental veneer data)
. mixed gcf followup base_gcf cda age || patient: followup,
> covariance(unstructured) || tooth:, reml nolog dfmethod(kroger)
Mixed-effects REML regression                   Number of obs =    110
        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum     Average    Maximum
----------------+-------------------------------------------------------------
        patient |         12          2         9.2         12
          tooth |         55          2         2.0          2
------------------------------------------------------------------------------
DF method: Kenward-Roger                        DF:       min =  10.41
                                                          avg =  28.96
                                                          max =  50.71
                                                F(4, 27.96)   =   1.47
Log restricted-likelihood = -420.92761          Prob > F      = 0.2370
------------------------------------------------------------------------------
         gcf | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    followup |   .3009815   1.938641     0.16   0.879    -3.96767    4.569633
    base_gcf |  -.0183127   .1466261    -0.12   0.901    -.3132419    .2766164
         cda |   -.329303   .5533506    -0.60   0.554    -1.440355    .7817493
         age |  -.5773932   .2350491    -2.46   0.033    -1.098324   -.056462
       _cons |   45.73862   13.21824     3.46   0.002     18.53866    72.93858
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects parameters |   Estimate   Std. err.      [95% conf. interval]
----------------------------+-------------------------------------------------
patient: Unstructured       |
                var(followup) |   41.88772   18.79997      17.38009    100.9535
                   var(_cons) |   524.9851   253.0205      204.1287    1350.175
          cov(followup,_cons) |  -140.4229   66.57623     -270.9099   -9.935904
----------------------------+-------------------------------------------------
tooth: Identity             |
                   var(_cons) |   47.45738   16.63034       23.8792     94.3165
----------------------------+-------------------------------------------------
                var(Residual) |   48.86704   10.50523      32.06479    74.47382
------------------------------------------------------------------------------
LR test vs. linear model: chi2(4) = 91.12                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Rather than specifying option `dftable(pvalue)` or `dftable(ci)` at estimation, we can display the covariate-specific DFs during postestimation by typing

```
. estat df
Degrees of freedom
```

	Kenward-Roger
gcf	
followup	10.96355
base_gcf	47.2708
cda	50.70932
age	10.41127
_cons	25.43377

`estat df` can also compare different DF methods using the `method()` option. For example, we can compare the Kenward–Roger method with the Satterthwaite method by typing

```
. estat df, method(kroger satterthwaite)
Degrees of freedom
```

	Kenward-Roger	Satterthwaite
gcf		
followup	10.96355	10.96355
base_gcf	47.2708	47.2708
cda	50.70932	50.70932
age	10.41127	10.41127
_cons	25.43377	25.43377

The two methods produce the same estimates of DFs for single-hypothesis tests, but the results differ for multiple-hypotheses tests; see example 4 of [ME] **mixed postestimation** for details.

Suppose that we decide to proceed with the Satterthwaite method in subsequent analysis. Rather than retyping our `mixed` command with the `dfmethod(satterthwaite)` option, we can post the Satterthwaite DFs using the `post` option of `estat df`.

```
. estat df, method(satterthwaite) post
Degrees of freedom
```

	Satterthwaite
gcf	
followup	10.96355
base_gcf	47.2708
cda	50.70932
age	10.41127
_cons	25.43377

The returned values associated with `dfmethod(kroger)` from the `mixed` command will be replaced with those of `dfmethod(satterthwaite)`.

◁

Stored results

`estat df` stores the following in `r()`:

Macros
- `r(dfmethods)` — DF methods

Matrices
- `r(df)` — parameter-specific DFs for each method specified in `method()`
- `r(V_df)` — variance–covariance matrix of the estimators when `kroger` method is specified

If `post()` is specified, `estat df` also stores the following in `e()`:

Scalars
- `e(F)` — overall F test statistic for the method specified in `post()`
- `e(ddf_m)` — model DDF for the method specified in `post()`
- `e(df_max)` — maximum DF for the method specified in `post()`
- `e(df_avg)` — average DF for the method specified in `post()`
- `e(df_min)` — minimum DF for the method specified in `post()`

Macros
- `e(dfmethod)` — DF method specified in `post()`
- `e(dftitle)` — title for DF method

Matrices
- `e(df)` — parameter-specific DFs for the method specified in `post()`
- `e(V_df)` — variance–covariance matrix of the estimators when `kroger` method is posted

Also see

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

estat group — Summarize the composition of the nested groups

Description

`estat group` reports the number of groups and minimum, average, and maximum group sizes for each level of the model. Model levels are identified by the corresponding group variable in the data. Because groups are treated as nested, the information in this summary may differ from what you would get if you used the `tabulate` command on each group variable individually.

Menu for estat

Statistics > Postestimation

Syntax

```
estat group
```

Remarks and examples

See example 3 in [ME] **mixed postestimation** and example 4 in [ME] **menl postestimation**.

Also see

[ME] **mecloglog** — Multilevel mixed-effects complementary log–log regression

[ME] **meglm** — Multilevel mixed-effects generalized linear models

[ME] **meintreg** — Multilevel mixed-effects interval regression

[ME] **melogit** — Multilevel mixed-effects logistic regression

[ME] **menbreg** — Multilevel mixed-effects negative binomial regression

[ME] **menl** — Nonlinear mixed-effects regression

[ME] **meologit** — Multilevel mixed-effects ordered logistic regression

[ME] **meoprobit** — Multilevel mixed-effects ordered probit regression

[ME] **mepoisson** — Multilevel mixed-effects Poisson regression

[ME] **meprobit** — Multilevel mixed-effects probit regression

[ME] **mestreg** — Multilevel mixed-effects parametric survival models

[ME] **metobit** — Multilevel mixed-effects tobit regression

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

estat icc — Estimate intraclass correlations

Description | Menu for estat | Syntax | Option
Remarks and examples | Stored results | Methods and formulas | Also see

Description

`estat icc` is for use after estimation with `mixed`, `meintreg`, `metobit`, `melogit`, `meprobit`, `meologit`, `meoprobit`, and `mecloglog`. `estat icc` is also for use after estimation with `meglm` in cases when the fitted model is a linear, logit, probit, ordered logit, ordered probit, or complementary log–log mixed-effects model.

`estat icc` displays the intraclass correlation for pairs of responses at each nested level of the model. Intraclass correlations are available for random-intercept models or for random-coefficients models conditional on random-effects covariates being equal to 0. They are not available for crossed-effects models or with residual error structures other than independent structures.

Menu for estat

Statistics > Postestimation

Syntax

```
estat icc [ , level(#) ]
```

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Option

`level(#)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see **[U] 20.8 Specifying the width of confidence intervals**.

Remarks and examples

See, for instance, example 2 in [ME] **mixed postestimation** and examples 1 and 4 in [ME] **melogit postestimation**.

Stored results

`estat icc` stores the following in `r()`:

Scalars

`r(icc#)`	level-# intraclass correlation
`r(se#)`	standard errors of level-# intraclass correlation
`r(level)`	confidence level of confidence intervals

Macros

`r(label#)`	label for level #

Matrices

`r(ci#)`	vector of confidence intervals (lower and upper) for level-# intraclass correlation

For a G-level nested model, # can be any integer between 2 and G.

Methods and formulas

Intraclass correlations

Consider a simple, two-level random-intercept model, stated in terms of a latent linear response, where only $y_{ij} = I(y^*_{ij} > 0)$ is observed for the latent variable,

$$y^*_{ij} = \beta + u^{(2)}_j + \epsilon^{(1)}_{ij}$$

with $i = 1, \ldots, n_j$ and level-2 groups $j = 1, \ldots, M$. Here β is an unknown fixed intercept, $u^{(2)}_j$ is a level-2 random intercept, and $\epsilon^{(1)}_{ij}$ is a level-1 error term. In a mixed-effects linear, probit, and ordered probit regression, errors are assumed to be normally distributed with mean 0 and variance γ. In a mixed-effects logistic and ordered logistic regression, errors are assumed to be logistic with mean 0 and variance γ. Random intercepts are assumed to be normally distributed with mean 0 and variance σ^2_2 and to be independent of error terms.

The intraclass correlation for this model is

$$\rho = \text{Corr}(y^*_{ij}, y^*_{i'j}) = \frac{\sigma^2_2}{\gamma + \sigma^2_2}$$

where $\gamma = \sigma^2_1$ for a mixed-effects linear regression, $\gamma = 1$ for a mixed-effects probit and ordered probit regression, $\gamma = \pi^2/3$ for a mixed-effects logistic and ordered logistic regression, and $\gamma = \pi^2/6$ for a mixed-effects complementary log–log regression. The intraclass correlation corresponds to the correlation between the latent responses i and i' from the same group j.

Now consider a three-level nested random-intercept model,

$$y^*_{ijk} = \beta + u^{(2)}_{jk} + u^{(3)}_k + \epsilon^{(1)}_{ijk}$$

for measurements $i = 1, \ldots, n_{jk}$ and level-2 groups $j = 1, \ldots, M_{1k}$ nested within level-3 groups $k = 1, \ldots, M_2$. Here $u^{(2)}_{jk}$ is a level-2 random intercept, $u^{(3)}_k$ is a level-3 random intercept, and $\epsilon^{(1)}_{ijk}$ is a level-1 error term. The random intercepts are assumed to be normally distributed with mean 0 and variances σ^2_2 and σ^2_3, respectively, and to be mutually independent. The error terms are also independent of the random intercepts.

We can consider two types of intraclass correlations for this model. We will refer to them as level-2 and level-3 intraclass correlations. The level-3 intraclass correlation is

$$\rho^{(3)} = \text{Corr}(y^*_{ijk}, y^*_{i'j'k}) = \frac{\sigma^2_3}{\gamma + \sigma^2_2 + \sigma^2_3}$$

This is the correlation between latent responses i and i' from the same level-3 group k and from different level-2 groups j and j'.

The level-2 intraclass correlation is

$$\rho^{(2)} = \text{Corr}(y^*_{ijk}, y^*_{i'jk}) = \frac{\sigma^2_2 + \sigma^2_3}{\gamma + \sigma^2_2 + \sigma^2_3}$$

This is the correlation between latent responses i and i' from the same level-3 group k and level-2 group j. (Note that level-1 intraclass correlation is undefined.)

More generally, for a G-level nested random-intercept model, the g-level intraclass correlation is defined as

$$\rho^{(g)} = \frac{\sum_{l=g}^{G} \sigma_l^2}{\gamma + \sum_{l=2}^{G} \sigma_l^2}$$

The above formulas also apply in the presence of fixed-effects covariates $\mathbf{X}$ in a random-effects model. In this case, intraclass correlations are conditional on fixed-effects covariates and are referred to as residual intraclass correlations. `estat icc` also uses the same formulas to compute intraclass correlations for random-coefficients models, assuming 0 baseline values for the random-effects covariates, and labels them as conditional intraclass correlations.

Intraclass correlations will always fall in [0,1] because variance components are nonnegative. To accommodate the range of an intraclass correlation, we use the logit transformation to obtain confidence intervals. We use the delta method to estimate the standard errors of the intraclass correlations.

Let $\hat{\rho}^{(g)}$ be a point estimate of the intraclass correlation and $\widehat{\mathrm{SE}}(\hat{\rho}^{(g)})$ be its standard error. The $(1 - \alpha) \times 100\%$ confidence interval for $\mathrm{logit}(\rho^{(g)})$ is

$$\mathrm{logit}(\hat{\rho}^{(g)}) \pm z_{\alpha/2} \frac{\widehat{\mathrm{SE}}(\hat{\rho}^{(g)})}{\hat{\rho}^{(g)}(1 - \hat{\rho}^{(g)})}$$

where $z_{\alpha/2}$ is the $1 - \alpha/2$ quantile of the standard normal distribution and $\mathrm{logit}(x) = \ln\{x/(1 - x)\}$. Let k_u be the upper endpoint of this interval, and let k_l be the lower. The $(1 - \alpha) \times 100\%$ confidence interval for $\rho^{(g)}$ is then given by

$$\left(\frac{1}{1 + e^{-k_l}}, \frac{1}{1 + e^{-k_u}} \right)$$

Also see

[ME] **mecloglog** — Multilevel mixed-effects complementary log–log regression

[ME] **meglm** — Multilevel mixed-effects generalized linear models

[ME] **meintreg** — Multilevel mixed-effects interval regression

[ME] **melogit** — Multilevel mixed-effects logistic regression

[ME] **meologit** — Multilevel mixed-effects ordered logistic regression

[ME] **meoprobit** — Multilevel mixed-effects ordered probit regression

[ME] **meprobit** — Multilevel mixed-effects probit regression

[ME] **metobit** — Multilevel mixed-effects tobit regression

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

estat recovariance — Display estimated random-effects covariance matrices

Description Menu for estat Syntax Options
Remarks and examples Stored results Also see

Description

`estat recovariance` is for use after estimation with `menl` and `mixed`.

`estat recovariance` displays the estimated variance–covariance matrix of the random effects for each level in the model.

Menu for estat

Statistics > Postestimation

Syntax

`estat recovariance` [, `relevel(`*levelvar*`) correlation` *matlist_options*]

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`relevel(`*levelvar*`)` specifies the level in the model for which the random-effects covariance matrix is to be displayed. By default, the covariance matrices for all levels in the model are displayed. *levelvar* is the name of the model level and is either the name of the variable describing the grouping at that level or is `_all`, a special designation for a group comprising all the estimation data. The `_all` designation is not supported with `menl`.

`correlation` displays the covariance matrix as a correlation matrix.

matlist_options are style and formatting options that control how the matrix (or matrices) is displayed; see [P] **matlist** for a list of options that are available.

Remarks and examples

For `menl`, the rows and columns of the matrix are labeled with full random-effects names as they are defined in the model.

For other commands, the rows and columns of the matrix are labeled as `_cons` for the random intercepts; for random coefficients, the label is the name of the associated variable in the data.

See example 1 in [ME] **mixed postestimation**.

Stored results

`estat recovariance` stores the following in `r()`:

Scalars

`r(relevels)`	number of levels

Matrices

`r(Cov#)`	level-# random-effects covariance matrix
`r(Corr#)`	level-# random-effects correlation matrix (if option `correlation` was specified)

For a G-level nested model, # can be any integer between 2 and G.

Also see

[ME] **menl** — Nonlinear mixed-effects regression

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

estat sd — Display variance components as standard deviations and correlations

Description Menu for estat Syntax Options
Remarks and examples Stored results Also see

Description

`estat sd` displays the random-effects and within-group error parameter estimates as standard deviations and correlations.

Menu for estat

Statistics > Postestimation

Syntax

`estat sd [ , variance verbose post coeflegend ]`

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`variance` specifies that `estat sd` display the random-effects and within-group error parameter estimates as variances and covariances. If the `post` option is specified, the estimated variances and covariances and their respective standard errors are posted to `e()`. `variance` is allowed only after `mixed` and `menl`.

`verbose` specifies that the full estimation table be displayed. By default, only the random-effects and within-group error parameters are displayed. This option is implied when `post` is specified.

`post` causes `estat sd` to behave like a Stata estimation (e-class) command. `estat sd` posts the vector of calculated standard deviation and correlation parameters along with the corresponding variance–covariance matrix to `e()`, so that you can treat the estimated parameters just as you would results from any other estimation command. For example, you could use `test` to perform simultaneous tests of hypotheses on the parameters, or you could use `lincom` to create linear combinations.

The following option is not shown in the dialog box:

`coeflegend` specifies that the legend of the coefficients and how to specify them in an expression be displayed rather than displaying the statistics for the coefficients. This option is allowed only if `post` is also specified.

Remarks and examples

See example 1 in [ME] **mixed postestimation** and example 16 in [ME] **menl**.

Stored results

`estat sd` stores the following in `r()`:

Matrices

`r(b)`	coefficient vector
`r(V)`	variance–covariance matrix of the estimators
`r(table)`	table of results

If `post` is specified, `estat sd` stores the following in `e()`:

Macros

`e(cmd)`	`estat sd`
`e(properties)`	`b V`

Matrices

`e(b)`	coefficient vector
`e(V)`	variance–covariance matrix of the estimators

Also see

[ME] **mecloglog** — Multilevel mixed-effects complementary log–log regression

[ME] **meglm** — Multilevel mixed-effects generalized linear models

[ME] **meintreg** — Multilevel mixed-effects interval regression

[ME] **melogit** — Multilevel mixed-effects logistic regression

[ME] **menbreg** — Multilevel mixed-effects negative binomial regression

[ME] **menl** — Nonlinear mixed-effects regression

[ME] **meologit** — Multilevel mixed-effects ordered logistic regression

[ME] **meoprobit** — Multilevel mixed-effects ordered probit regression

[ME] **mepoisson** — Multilevel mixed-effects Poisson regression

[ME] **meprobit** — Multilevel mixed-effects probit regression

[ME] **mestreg** — Multilevel mixed-effects parametric survival models

[ME] **metobit** — Multilevel mixed-effects tobit regression

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

estat wcorrelation — Display within-cluster correlations and standard deviations

Description | Menu for estat | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | Reference
Also see

Description

`estat wcorrelation` is for use after estimation with `menl` and `mixed`.

`estat wcorrelation` displays the overall correlation matrix for a given cluster calculated on the basis of the design of the random effects and their assumed covariance and the correlation structure of the errors. This allows for a comparison of different multilevel models in terms of the ultimate within-cluster correlation matrix that each model implies.

Menu for estat

Statistics > Postestimation

Syntax

`estat wcorrelation` [, *options*]

options	Description
`at(`*at_spec*`)`	specify the cluster for which you want the correlation matrix; default is the first two-level cluster encountered in the data
`all`	display correlation matrix for all the data
`covariance`	display the covariance matrix instead of the correlation matrix
`list`	list the data corresponding to the correlation matrix
`nosort`	list the rows and columns of the correlation matrix in the order they were originally present in the data
`iterate(`#`)`	maximum number of iterations to compute random effects; default is `iterate(50)`; only for use after `menl`
`tolerance(`#`)`	convergence tolerance when computing random effects; default is `tolerance(1e-6)`; only for use after `menl`
`nrtolerance(`#`)`	scaled gradient tolerance when computing random effects; default is `nrtolerance(1e-5)`; only for use after `menl`
`nonrtolerance`	ignore the `nrtolerance()` option; only for use after `menl`
`format(`*%fmt*`)`	set the display format; default is `format(%6.3f)`
matlist_options	style and formatting options that control how matrices are displayed

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

Options

`at(`*at_spec*`)` specifies the cluster of observations for which you want the within-cluster correlation matrix. *at_spec* is

relevel_var = *value* [, *relevel_var* = *value* ...]

For example, if you specify

```
. estat wcorrelation, at(school = 33)
```

you get the within-cluster correlation matrix for those observations in school 33. If you specify

```
. estat wcorrelation, at(school = 33 classroom = 4)
```

you get the correlation matrix for classroom 4 in school 33.

If `at()` is not specified, then you get the correlations for the first level-two cluster encountered in the data. This is usually what you want.

`all` specifies that you want the correlation matrix for all the data. This is not recommended unless you have a relatively small dataset or you enjoy seeing large $n \times n$ matrices. However, this can prove useful in some cases.

`covariance` specifies that the within-cluster covariance matrix be displayed instead of the default correlations and standard deviations.

`list` lists the model data for those observations depicted in the displayed correlation matrix. With linear mixed-effects models, this option is also useful if you have many random-effects design variables and you wish to see the represented values of these design variables.

`nosort` lists the rows and columns of the correlation matrix in the order that they were originally present in the data. Normally, `estat wcorrelation` will first sort the data according to level variables, by-group variables, and time variables to produce correlation matrices whose rows and columns follow a natural ordering. `nosort` suppresses this.

`iterate(#)` specifies the maximum number of iterations when computing estimates of the random effects. The default is `iterate(50)`. This option is only for use after `menl`.

`tolerance(#)` specifies a convergence tolerance when computing estimates of the random effects. The default is `tolerance(1e-6)`. This option is only for use after `menl`.

`nrtolerance(#)` and `nonrtolerance` control the tolerance for the scaled gradient when computing estimates of the random effects. These options are only for use after `menl`.

`nrtolerance(#)` specifies the tolerance for the scaled gradient. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

`nonrtolerance` specifies that the default `nrtolerance()` criterion be turned off.

`format(`*%fmt*`)` sets the display format for the standard deviation vector and correlation matrix. The default is `format(%6.3f)`.

matlist_options are style and formatting options that control how the matrix (or matrices) is displayed; see [P] **matlist** for a list of options that are available.

Remarks and examples

▷ Example 1: Displaying within-cluster correlations for different clusters

Here we fit a model where different clusters have different within-cluster correlations, and we show how to display them for different clusters. We use the Asian children weight data from example 6 of [ME] **mixed**.

```
. use https://www.stata-press.com/data/r19/childweight
(Weight data on Asian children)
. mixed weight age || id: age, covariance(unstructured)
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -344.37065
Iteration 1:  Log likelihood = -342.83814
Iteration 2:  Log likelihood = -342.71861
Iteration 3:  Log likelihood = -342.71777
Iteration 4:  Log likelihood = -342.71777
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        198
Group variable: id                              Number of groups  =         68
                                                Obs per group:
                                                              min =          1
                                                              avg =        2.9
                                                              max =          5
                                                Wald chi2(1)      =     755.27
Log likelihood = -342.71777                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	3.459671	.1258878	27.48	0.000	3.212936	3.706407
_cons	5.110496	.149478	34.19	0.000	4.817524	5.403468

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Unstructured				
var(age)	.2023928	.12429	.0607393	.6744041
var(_cons)	.0970259	.1108024	.0103473	.9098067
cov(age,_cons)	.1401334	.0566912	.0290206	.2512461
var(Residual)	1.357922	.1650507	1.070075	1.723199

```
LR test vs. linear model: chi2(3) = 27.38                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We use estat wcorrelation to display the within-cluster correlations for the first cluster.

```
. estat wcorrelation, list
Standard deviations and correlations for id = 45:
Standard deviations:
```

obs	1	2	3	4	5
sd	1.224	1.314	1.448	1.506	1.771

Correlations:

obs	1	2	3	4	5
1	1.000				
2	0.141	1.000			
3	0.181	0.274	1.000		
4	0.193	0.293	0.376	1.000	
5	0.230	0.348	0.447	0.477	1.000

Data:

	id	weight	age
1.	45	5.171	.136893
2.	45	10.86	.657084
3.	45	13.15	1.21834
4.	45	13.2	1.42916
5.	45	15.88	2.27242

We specified the list option to display the data associated with the cluster. The next cluster in the dataset has ID 258. To display the within-cluster correlations for this cluster, we specify the at() option.

```
. estat wcorrelation, at(id=258) list
Standard deviations and correlations for id = 258:
Standard deviations:
```

obs	1	2	3	4
sd	1.231	1.320	1.424	1.782

Correlations:

obs	1	2	3	4
1	1.000			
2	0.152	1.000		
3	0.186	0.270	1.000	
4	0.244	0.356	0.435	1.000

Data:

	id	weight	age
1.	258	5.3	.19165
2.	258	9.74	.687201
3.	258	9.98	1.12799
4.	258	11.34	2.30527

The within-cluster correlations for this model depend on age. The values for age in the two clusters are different, as are the corresponding within-cluster correlations.

◁

See example 1 of [ME] **mixed postestimation** for a model fit where each cluster had the same model-implied within-cluster correlations.

Stored results

`estat wcorrelation` stores the following in `r()`:

Matrices

`r(sd)`	standard deviations
`r(Corr)`	within-cluster correlation matrix
`r(Cov)`	within-cluster variance–covariance matrix
`r(G)`	variance–covariance matrix of random effects
`r(Z)`	model-based design matrix
`r(R)`	variance–covariance matrix of level-one errors
`r(path)`	path identifying cluster for which correlation is reported

Results `r(G)`, `r(Z)`, and `r(R)` are available only after `mixed`. Result `r(path)` is available only after `menl`.

Methods and formulas

Methods and formulas are presented under the following headings:

Linear mixed-effects model
Nonlinear mixed-effects model

Linear mixed-effects model

A two-level linear mixed model of the form

$$\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j + \boldsymbol{\epsilon}_j$$

implies the marginal model

$$\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta} + \boldsymbol{\epsilon}^*_j$$

where $\boldsymbol{\epsilon}^*_j \sim N(\mathbf{0}, \mathbf{V}_j)$, $\mathbf{V}_j = \mathbf{Z}_j\mathbf{G}\mathbf{Z}_j' + \mathbf{R}$. In a marginal model, the random part is described in terms of the marginal or total errors $\boldsymbol{\epsilon}^*_j$, and $\mathbf{V}_j$ is the covariance structure of these errors.

`estat wcorrelation` calculates the marginal covariance matrix $\widehat{\mathbf{V}}_j$ for cluster j and by default displays the results in terms of standard deviations and correlations. This allows for a comparison of different multilevel models in terms of the ultimate within-cluster correlation matrix that each model implies.

Calculation of the marginal covariance matrix extends naturally to higher-level models; see, for example, chapter 4.8 in West, Welch, and Gałecki (2022).

Nonlinear mixed-effects model

For nonlinear mixed-effects models, there is no closed-form expression for the marginal covariance matrix $\mathrm{Cov}(\mathbf{y}_j)$. This is because it is expressed in terms of a q-dimensional integral (q is the number of random effects in the model), which, in general, is analytically intractable. Under the linear mixed-effects approximation, the marginal covariance matrix is estimated by $\widehat{\mathbf{V}}_j = \widehat{\mathbf{Z}}_j\widehat{\boldsymbol{\Sigma}}\widehat{\mathbf{Z}}_j' + \hat{\sigma}^2\widehat{\boldsymbol{\Lambda}}_j$, where $\widehat{\mathbf{Z}}_j$, $\widehat{\boldsymbol{\Sigma}}$, and $\widehat{\boldsymbol{\Lambda}}_j$ are defined in *Methods and formulas* of [ME] **menl**.

`estat wcorrelation` calculates the estimated marginal covariance matrix $\widehat{\mathbf{V}}_j$ for cluster j and by default displays the results in terms of standard deviations and correlations.

Under the linear mixed-effects approximation, estimation of the marginal covariance matrix extends naturally to higher-level models; see, for example, chapter 4.8 in West, Welch, and Gałecki (2022).

Reference

West, B. T., K. B. Welch, and A. T. Gałecki. 2022. *Linear Mixed Models: A Practical Guide Using Statistical Software*. 3rd ed. Boca Raton, FL: CRC Press.

Also see

[ME] **menl** — Nonlinear mixed-effects regression

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

mecloglog — Multilevel mixed-effects complementary log–log regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | Reference | Also see

Description

`mecloglog` fits mixed-effects models for binary or binomial responses. The conditional distribution of the response given the random effects is assumed to be Bernoulli, with probability of success determined by the inverse complementary log–log function.

Quick start

Two-level complementary log–log model of `y` on `x` with random intercepts by `lev2`

```
mecloglog y x || lev2:
```

Add binary variable `a` and random coefficients for `a`

```
mecloglog y x a || lev2: a
```

Same as above, but allow the random effects to be correlated

```
mecloglog y x a || lev2: a, covariance(unstructured)
```

Three-level random-intercept model of `y` on `x` with `lev2` nested within `lev3`

```
mecloglog y x || lev3: || lev2:
```

Crossed-effects model of `y` on `x` with two-way crossed random effects by factors `a` and `b`

```
mecloglog y x || _all:R.a || b:
```

Menu

Statistics > Multilevel mixed-effects models > Complementary log–log regression

Syntax

```
mecloglog depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
noconstant	suppress constant term from the fixed-effects equation
offset(*varname*)	include *varname* in model with coefficient constrained to 1
asis	retain perfect predictor variables

re_options	Description
Model	
covariance(*vartype*)	variance–covariance structure of the random effects
noconstant	suppress constant term from the random-effects equation
fweight(*varname*)	frequency weights at higher levels
iweight(*varname*)	importance weights at higher levels
pweight(*varname*)	sampling weights at higher levels

options	Description
Model	
`binomial(`*varname* | *#*`)`	set binomial trials if data are in binomial form
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`*#*`)`	set confidence level; default is `level(95)`
`eform`	report exponentiated coefficients
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`*#*`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid[(`*gridspec*`)]`	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

bayes, by, collect, and svy are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: mecloglog**.

vce() and weights are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

startvalues(), startgrid, noestimate, dnumerical, collinear, and coeflegend do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

noconstant suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

offset(*varname*) specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

asis forces retention of perfect predictor variables and their associated, perfectly predicted observations and may produce instabilities in maximization; see [R] **probit**.

covariance(*vartype*) specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: independent, exchangeable, identity, unstructured, fixed(*matname*), or pattern(*matname*).

covariance(independent) covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is covariance(independent) unless a crossed random-effects model is fit, in which case the default is covariance(identity).

covariance(exchangeable) structure specifies one common variance for all random effects and one common pairwise covariance.

covariance(identity) is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

covariance(unstructured) allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

covariance(fixed(*matname*)) and covariance(pattern(*matname*)) covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of

matname are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`binomial(`*varname* | #`)` specifies that the data are in binomial form; that is, *depvar* records the number of successes from a series of binomial trials. This number of trials is given either as *varname*, which allows this number to vary over the observations, or as the constant #. If `binomial()` is not specified (the default), *depvar* is treated as Bernoulli, with any nonzero, nonmissing values indicating positive responses.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(`#`)`; see [R] **Estimation options**.

`eform` reports exponentiated coefficients and corresponding standard errors and confidence intervals. This option may be specified either at estimation or upon replay.

`nocnsreport`; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite performs mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for mecloglog are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with mecloglog but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

Mixed-effects complementary log–log (cloglog) regression is cloglog regression containing both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

`mecloglog` allows for many levels of random effects. However, for simplicity, we here consider the two-level model, where for a series of M independent clusters, and conditional on a set of fixed effects $\mathbf{x}_{ij}$ and a set of random effects $\mathbf{u}_j$,

$$\Pr(y_{ij} = 1|\mathbf{x}_{ij}, \mathbf{u}_j) = H(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j) \tag{1}$$

for $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The responses are the binary-valued y_{ij}, and we follow the standard Stata convention of treating $y_{ij} = 1$ if $depvar_{ij} \neq 0$ and treating $y_{ij} = 0$ otherwise. The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard cloglog regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. For notational convenience here and throughout this manual entry, we suppress the dependence of y_{ij} on $\mathbf{x}_{ij}$.

The $1 \times q$ vector $\mathbf{z}_{ij}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$, so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

Finally, because this is cloglog regression, $H(\cdot)$ is the inverse of the complementary log–log function that maps the linear predictor to the probability of a success ($y_{ij} = 1$) with $H(v) = 1 - \exp\{-\exp(v)\}$.

Model (1) may also be stated in terms of a latent linear response, where only $y_{ij} = I(y^*_{ij} > 0)$ is observed for the latent

$$y^*_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij}$$

The errors ϵ_{ij} are independent and identically extreme-value (Gumbel) distributed with the mean equal to Euler's constant and variance $\sigma^2_\epsilon = \pi^2/6$, independently of $\mathbf{u}_j$. This nonsymmetric error distribution is an alternative to the symmetric error distribution underlying logistic and probit analysis and is usually used when the positive (or negative) outcome is rare.

Below we present two short examples of mixed-effects cloglog regression; refer to [ME] **me** and [ME] **meglm** for examples of other random-effects models. A two-level cloglog model can also be fit using `xtcloglog` with the `re` option; see [XT] **xtcloglog**. In the absence of random effects, mixed-effects cloglog regression reduces to standard cloglog regression; see [R] **cloglog**.

▷ Example 1: Two-level random-intercept model

In example 1 of [XT] **xtcloglog**, we analyze unionization of women in the United States over the period 1970–1988. The women are identified by the variable `idcode`. Here we refit that model with `mecloglog`. Because the original example used 12 integration points by default, we request 12 integration points as well.

```
. use https://www.stata-press.com/data/r19/union
(NLS Women 14-24 in 1968)
```

```
. mecloglog union age grade not_smsa south##c.year || idcode:, intpoints(12)
Fitting fixed-effects model:

Iteration 0:  Log likelihood = -14237.139
Iteration 1:  Log likelihood = -13546.159
Iteration 2:  Log likelihood = -13540.611
Iteration 3:  Log likelihood = -13540.607
Iteration 4:  Log likelihood = -13540.607

Refining starting values:

Grid node 0:  Log likelihood = -11104.448

Fitting full model:

Iteration 0:  Log likelihood = -11104.448
Iteration 1:  Log likelihood = -10617.891
Iteration 2:  Log likelihood = -10537.919
Iteration 3:  Log likelihood = -10535.946
Iteration 4:  Log likelihood = -10535.941
Iteration 5:  Log likelihood = -10535.941

Mixed-effects cloglog regression                Number of obs     =     26,200
Group variable: idcode                          Number of groups  =      4,434
                                                Obs per group:
                                                              min =          1
                                                              avg =        5.9
                                                              max =         12
Integration method: mvaghermite                 Integration pts.  =         12
                                                Wald chi2(6)      =     248.12
Log likelihood = -10535.941                     Prob > chi2       =     0.0000
```

union	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	.0128542	.0119441	1.08	0.282	-.0105559	.0362642
grade	.0699965	.0138551	5.05	0.000	.0428409	.097152
not_smsa	-.1982009	.0649258	-3.05	0.002	-.3254531	-.0709488
1.south	-2.049901	.4892644	-4.19	0.000	-3.008842	-1.090961
year	-.0006158	.0123999	-0.05	0.960	-.0249191	.0236875
south#c.year						
1	.0164457	.0060685	2.71	0.007	.0045516	.0283399
_cons	-3.277375	.6610552	-4.96	0.000	-4.57302	-1.981731
idcode						
var(_cons)	3.489803	.1630921			3.184351	3.824555

```
LR test vs. cloglog model: chibar2(01) = 6009.33      Prob >= chibar2 = 0.0000
```

The estimates are practically the same. `xtcloglog` reports the estimated variance component as a standard deviation, $\hat{\sigma}_{\mathbf{u}} = 1.86$. `mecloglog` reports $\hat{\sigma}^2_{\mathbf{u}} = 3.49$, the square root of which is 1.87. We find that age and education each have a positive effect on union membership, although the former is not statistically significant. Women who live outside of metropolitan areas are less likely to unionize.

The estimated variance of the random intercept at the individual level, $\widehat{\boldsymbol{\sigma}}^2$, is 3.49 with standard error 0.16. The reported likelihood-ratio test shows that there is enough variability between women to favor a mixed-effects cloglog regression over an ordinary cloglog regression; see *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

◁

Example 2: Three-level random-intercept model

Rabe-Hesketh, Toulopoulou, and Murray (2001) analyzed data from a study that measured the cognitive ability of patients with schizophrenia compared with their relatives and control subjects. Cognitive ability was measured as the successful completion of the "Tower of London", a computerized task, measured at three levels of difficulty. For all but one of the 226 subjects, there were three measurements (one for each difficulty level). Because patients' relatives were also tested, a family identifier, `family`, was also recorded.

We fit a cloglog model with response `dtlm`, the indicator of cognitive function, and with covariates `difficulty` and a set of indicator variables for `group`, with the controls (`group==1`) being the base category. We also allow for random effects due to families and due to subjects within families. The first is a random intercept (constant only) at the `family` level, and the second is a random intercept at the `subject` level. The order in which these are specified (from left to right) is significant—`mecloglog` assumes that `subject` is nested within `family`. The equations are separated by `||`.

```
. use https://www.stata-press.com/data/r19/towerlondon
(Tower of London data)
. mecloglog dtlm difficulty i.group || family: || subject:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -337.21921
Iteration 1:  Log likelihood = -313.79023
Iteration 2:  Log likelihood = -313.56906
Iteration 3:  Log likelihood = -313.56888
Iteration 4:  Log likelihood = -313.56888
Refining starting values:
Grid node 0:  Log likelihood = -314.57061
Fitting full model:
Iteration 0:  Log likelihood = -314.57061  (not concave)
Iteration 1:  Log likelihood = -308.82101
Iteration 2:  Log likelihood = -305.71841
Iteration 3:  Log likelihood = -305.26804
Iteration 4:  Log likelihood = -305.26516
Iteration 5:  Log likelihood = -305.26516
Mixed-effects cloglog regression               Number of obs     =        677
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
family	118	2	5.7	27
subject	226	2	3.0	3

```
Integration method: mvaghermite                Integration pts.  =          7
                                               Wald chi2(3)      =      83.32
Log likelihood = -305.26516                    Prob > chi2       =     0.0000
```

dtlm	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
difficulty	-1.342844	.1501508	-8.94	0.000	-1.637135	-1.048554
group						
2	-.1331007	.269389	-0.49	0.621	-.6610935	.3948922
3	-.7714314	.3097099	-2.49	0.013	-1.378452	-.164411
_cons	-1.6718	.2290325	-7.30	0.000	-2.120695	-1.222905
family						
var(_cons)	.2353453	.2924064			.0206122	2.687117
family>subject						
var(_cons)	.7737687	.4260653			.2629714	2.276742

```
LR test vs. cloglog model: chi2(2) = 16.61                 Prob > chi2 = 0.0002
Note: LR test is conservative and provided only for reference.
```

After adjusting for the random-effects structure, the probability of successful completion of the Tower of London decreases dramatically as the level of difficulty increases. Also, schizophrenics (`group==3`) tended not to perform as well as the control subjects.

◁

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by ||.

Stored results

mecloglog stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_dv)	number of dependent variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(N_clust)	number of clusters
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	mecloglog
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	cloglog
e(title)	title in estimation output
e(link)	cloglog
e(family)	bernoulli or binomial
e(clustvar)	name of cluster variable
e(offset)	offset
e(binomial)	binomial number of trials
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method

e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved
Matrices	
e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
Functions	
e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices	
r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

mecloglog is a convenience command for meglm with a cloglog link and a bernoulli or binomial family; see [ME] **meglm**.

Model (1) assumes Bernoulli data, a special case of the binomial. Because binomial data are also supported by mecloglog (option binomial()), the methods presented below are in terms of the more general binomial mixed-effects model.

For a two-level binomial model, consider the response y_{ij} as the number of successes from a series of r_{ij} Bernoulli trials (replications). For cluster j, $j = 1, \ldots, M$, the conditional distribution of $\mathbf{y}_j = (y_{j1}, \ldots, y_{jn_j})'$, given a set of cluster-level random effects $\mathbf{u}_j$, is

$$
\begin{aligned}
f(\mathbf{y}_j|\mathbf{u}_j) &= \prod_{i=1}^{n_j} \left[\binom{r_{ij}}{y_{ij}} \left\{H(\boldsymbol{\eta}_{ij})\right\}^{y_{ij}} \left\{1 - H(\boldsymbol{\eta}_{ij})\right\}^{r_{ij}-y_{ij}} \right] \\
&= \exp\left(\sum_{i=1}^{n_j} \left[y_{ij} \log\left\{H(\boldsymbol{\eta}_{ij})\right\} - (r_{ij} - y_{ij}) \exp(\boldsymbol{\eta}_{ij}) + \log \binom{r_{ij}}{y_{ij}} \right] \right)
\end{aligned}
$$

for $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \text{offset}_{ij}$ and $H(v) = 1 - \exp\{-\exp(v)\}$.

Defining $\mathbf{r}_j = (r_{j1}, \ldots, r_{jn_j})'$ and

$$c\left(\mathbf{y}_j, \mathbf{r}_j\right) = \sum_{i=1}^{n_j} \log \binom{r_{ij}}{y_{ij}}$$

where $c(\mathbf{y}_j, \mathbf{r}_j)$ does not depend on the model parameters, we can express the above compactly in matrix notation,

$$f(\mathbf{y}_j|\mathbf{u}_j) = \exp\left[\mathbf{y}_j' \log\left\{H(\boldsymbol{\eta}_j)\right\} - (\mathbf{r}_j - \mathbf{y}_j)' \exp(\boldsymbol{\eta}_j) + c\left(\mathbf{y}_j, \mathbf{r}_j\right)\right]$$

where $\boldsymbol{\eta}_j$ is formed by stacking the row vectors $\boldsymbol{\eta}_{ij}$. We extend the definitions of the functions $H(\cdot)$, $\log(\cdot)$, and $\exp(\cdot)$ to be vector functions where necessary.

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j, \mathbf{u}_j)$,

$$\begin{aligned}\mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2} \left|\boldsymbol{\Sigma}\right|^{-1/2} \int f(\mathbf{y}_j|\mathbf{u}_j) \exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right) d\mathbf{u}_j \\ &= \exp\left\{c\left(\mathbf{y}_j, \mathbf{r}_j\right)\right\} (2\pi)^{-q/2} \left|\boldsymbol{\Sigma}\right|^{-1/2} \int \exp\left\{h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right)\right\} d\mathbf{u}_j\end{aligned} \tag{2}$$

where

$$h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right) = \mathbf{y}_j' \log\left\{H(\boldsymbol{\eta}_j)\right\} - (\mathbf{r}_j - \mathbf{y}_j)' \exp(\boldsymbol{\eta}_j) - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j, \mathbf{r}_j, \mathbf{X}_j, \mathbf{Z}_j)$.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`mecloglog` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

Reference

Rabe-Hesketh, S., T. Toulopoulou, and R. M. Murray. 2001. Multilevel modeling of cognitive function in schizophrenic patients and their first degree relatives. *Multivariate Behavioral Research* 36: 279–298. https://doi.org/10.1207/S15327906MBR3602_07.

Also see

[ME] **mecloglog postestimation** — Postestimation tools for mecloglog

[ME] **melogit** — Multilevel mixed-effects logistic regression

[ME] **meprobit** — Multilevel mixed-effects probit regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: mecloglog** — Bayesian multilevel complementary log–log regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtcloglog** — Random-effects and population-averaged cloglog models

[U] 20 Estimation and postestimation commands

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation command is of special interest after `mecloglog`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as mean responses; linear predictions; density and distribution functions; standard errors; and Pearson, deviance, and Anscombe residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function
`pearson`	Pearson residuals
`deviance`	deviance residuals
`anscombe`	Anscombe residuals

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

`pearson`, `deviance`, `anscombe` may not be combined with `marginal`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`mu`, the default, calculates the predicted mean, that is, the probability of a positive outcome.

`eta`, `xb`, `stdp`, `density`, `distribution`, `pearson`, `deviance`, `anscombe`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, and `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for mean responses and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`pearson`	not allowed with `margins`
`deviance`	not allowed with `margins`
`anscombe`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.

Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects complementary log–log model with `mecloglog`. Here we show a short example of predicted probabilities and predicted random effects; refer to [ME] **meglm postestimation** for additional examples.

▷ Example 1: Obtaining predicted probabilities and random effects

In example 2 of [ME] **mecloglog**, we analyzed the cognitive ability (`dtlm`) of patients with schizophrenia compared with their relatives and control subjects, by using a three-level complementary log–log model with random effects at the family and subject levels. Cognitive ability was measured as the successful completion of the "Tower of London", a computerized task, measured at three levels of difficulty.

```
. use https://www.stata-press.com/data/r19/towerlondon
(Tower of London data)
. mecloglog dtlm difficulty i.group || family: || subject:
Fitting fixed-effects model:
  (output omitted)
Mixed-effects cloglog regression                Number of obs     =        677
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
family	118	2	5.7	27
subject	226	2	3.0	3

```
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(3)      =      83.32
Log likelihood = -305.26516                     Prob > chi2       =     0.0000
```

dtlm	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
difficulty	-1.342844	.1501508	-8.94	0.000	-1.637135	-1.048554
group						
2	-.1331007	.269389	-0.49	0.621	-.6610935	.3948922
3	-.7714314	.3097099	-2.49	0.013	-1.378452	-.164411
_cons	-1.6718	.2290325	-7.30	0.000	-2.120695	-1.222905
family						
var(_cons)	.2353453	.2924064			.0206122	2.687117
family> subject						
var(_cons)	.7737687	.4260653			.2629714	2.276742

```
LR test vs. cloglog model: chi2(2) = 16.61                 Prob > chi2 = 0.0002
Note: LR test is conservative and provided only for reference.
```

We obtain predicted probabilities based on the contribution of both fixed effects and random effects by typing

```
. predict pr
(option mu assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

As the note says, the predicted values are based on the posterior means of random effects. You can use the `modes` option to obtain predictions based on the posterior modes of random effects.

We obtain predictions of the posterior means themselves by typing

```
. predict re*, reffects
(calculating posterior means of random effects)
(using 7 quadrature points)
```

Because we have one random effect at the family level and another random effect at the subject level, Stata saved the predicted posterior means in the variables `re1` and `re2`, respectively. If you are not sure which prediction corresponds to which level, you can use the `describe` command to show the variable labels.

Here we list the data for family 16:

```
. list family subject dtlm pr re1 re2 if family==16, sepby(subject)
```

	family	subject	dtlm	pr	re1	re2
208.	16	5	1	.486453	.4184933	.2760492
209.	16	5	0	.1597047	.4184933	.2760492
210.	16	5	0	.0444156	.4184933	.2760492
211.	16	34	1	.9659582	.4184933	1.261488
212.	16	34	1	.5862808	.4184933	1.261488
213.	16	34	1	.205816	.4184933	1.261488
214.	16	35	0	.5571261	.4184933	-.1616545
215.	16	35	1	.1915688	.4184933	-.1616545
216.	16	35	0	.0540124	.4184933	-.1616545

We can see that the predicted random effects (`re1`) at the family level are the same for all members of the family. Similarly, the predicted random effects (`re2`) at the individual level are constant within each individual.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

Also see

[ME] **mecloglog** — Multilevel mixed-effects complementary log–log regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

meglm — Multilevel mixed-effects generalized linear models

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
References | Also see

Description

`meglm` fits multilevel mixed-effects generalized linear models. `meglm` allows a variety of distributions for the response conditional on normally distributed random effects.

Quick start

Without weights

Random-effects probit regression of `y` on `x1` with random intercepts by `lev2`

```
meglm y x1 || lev2:, family(binomial) link(probit)
```

Same as above, but fit a logit model and report odds ratios

```
meglm y x1 || lev2:, family(binomial) or
```

Two-level gamma model of `y` with fixed and random coefficients on `x1`

```
meglm y x1 || lev2: x1, family(gamma)
```

Nested three-level random-intercept Poisson model reporting incidence-rate ratios

```
meglm y x1 || lev3: || lev2:, family(poisson) irr
```

Two-level linear regression of `y` on `x1` and `x2` with random intercepts by `lev2`, random coefficients on `x2`, and robust standard errors

```
meglm y x1 x2 || lev2: x2, vce(robust)
```

With weights

Two-level linear regression of `y` on `x` with random intercepts by `psu` for two-stage sampling with PSU-level and observation-level sampling weights `wvar2` and `wvar1`, respectively

```
meglm y x [pweight=wvar1] || psu:, pweight(wvar2)
```

Add secondary sampling stage with units identified by `ssu` having weights `wvar2` and PSU-level weights `wvar3` for a three-level random-intercept model

```
meglm y x [pw=wvar1] || psu:, pw(wvar3) || ssu:, pw(wvar2)
```

Same as above, but `svyset` data first

```
svyset psu, weight(wvar3) || ssu, weight(wvar2) || _n, weight(wvar1)
svy: meglm y x || psu: || ssu:
```

Menu

Statistics > Multilevel mixed-effects models > Generalized linear models (GLM)

Syntax

```
meglm depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress the constant term from the fixed-effects equation
`exposure(`$varname_e$`)`	include $\ln(varname_e)$ in model with coefficient constrained to 1
`offset(`$varname_o$`)`	include $varname_o$ in model with coefficient constrained to 1
`asis`	retain perfect predictor variables

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`family(`*family*`)`	distribution of *depvar*; default is `family(gaussian)`
`link(`*link*`)`	link function; default varies per family
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`eform`	report exponentiated fixed-effects coefficients
`irr`	report fixed-effects coefficients as incidence-rate ratios
`or`	report fixed-effects coefficients as odds ratios
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

family	Description
gaussian	Gaussian (normal); the default
bernoulli	Bernoulli
binomial [# | *varname*]	binomial; default number of binomial trials is 1
gamma	gamma
nbinomial [mean | constant]	negative binomial; default dispersion is mean
ordinal	ordinal
poisson	Poisson

link	Description
identity	identity
log	log
logit	logit
probit	probit
cloglog	complementary log–log

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
pcaghermite	Pinheiro–Chao mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models
pclaplace	Pinheiro–Chao Laplacian approximation

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

bayes, by, collect, and svy are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: meglm**.

vce() and weights are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

startvalues(), startgrid, noestimate, dnumerical, collinear, and coeflegend do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

noconstant suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

exposure(*varname*$_e$) specifies a variable that reflects the amount of exposure over which the *depvar* events were observed for each observation; ln(*varname*$_e$) is included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`offset(`*varname*$_o$`)` specifies that *varname*$_o$ be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`asis` forces retention of perfect predictor variables and their associated, perfectly predicted observations and may produce instabilities in maximization; see [R] **probit**.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if *matname*$[i, j]$ = *matname*$[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`family(`*family*`)` specifies the distribution of *depvar*; `family(gaussian)` is the default.

`link(`*link*`)` specifies the link function; the default is the canonical link for the `family()` specified except for the gamma and negative binomial families.

If you specify both `family()` and `link()`, not all combinations make sense. You may choose from the following combinations:

	identity	log	logit	probit	cloglog
Gaussian	D	x			
Bernoulli			D	x	x
binomial			D	x	x
gamma		D			
negative binomial		D			
ordinal			D	x	x
Poisson		D			

D denotes the default.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(#)`; see [R] **Estimation options**.

`eform` reports exponentiated fixed-effects coefficients and corresponding standard errors and confidence intervals. This option may be specified either at estimation or upon replay.

`irr` reports estimated fixed-effects coefficients transformed to incidence-rate ratios, that is, $\exp(\beta)$ rather than β. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified either at estimation or upon replay. This option is allowed for count models only.

`or` reports estimated fixed-effects coefficients transformed to odds ratios, that is, $\exp(\beta)$ rather than β. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. `or` may be specified at estimation or upon replay. This option is allowed for logistic models only.

`nocnsreport`; see [R] **Estimation options**.

`notable` suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite and pcaghermite perform mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace and pclaplace perform the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point. Techniques pcaghermite and pclaplace are available only with family(binomial) and family(bernoulli) combined with link(logit) and with family(poisson); these techniques obtain the random-effects mode and curvature using the efficient hierarchical decomposition algorithm described in Pinheiro and Chao (2006). For hierarchical models, this algorithm takes advantage of the design structure to minimize memory use and utilizes a series of orthogonal triangulations to compute the factored random-effects Hessian indirectly, avoiding the sparse full Hessian. Techniques mcaghermite and laplace use Cholesky factorization on the full Hessian. For four- and higher-level hierarchical designs, there can be dramatic computation-time differences.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for meglm are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with meglm but are not shown in the dialog box:

startvalues(*svmethod*) specifies how starting values are to be computed. Starting values specified in from() override the computed starting values.

startvalues(zero) specifies that starting values be set to 0.

startvalues(constantonly) builds on startvalues(zero) by fitting a constant-only model to obtain estimates of the intercept and auxiliary parameters, and it substitutes 1 for the variances of random effects.

startvalues(fixedonly[, iterate(#)]) builds on startvalues(constantonly) by fitting a full fixed-effects model to obtain estimates of coefficients along with intercept and auxiliary parameters, and it continues to use 1 for the variances of random effects. This is the default behavior. iterate(#) limits the number of iterations for fitting the fixed-effects model.

startvalues(iv[, iterate(#)]) builds on startvalues(fixedonly) by using instrumental-variable methods with generalized residuals to obtain variances of random effects. iterate(#) limits the number of iterations for fitting the instrumental-variable model.

startvalues(iterate(#)) limits the number of iterations for fitting the default model (fixed effects).

startgrid[(*gridspec*)] performs a grid search on variance components of random effects to improve starting values. No grid search is performed by default unless the starting values are found to be not feasible, in which case meglm runs startgrid() to perform a "minimal" search involving q^3 likelihood evaluations, where q is the number of random effects. Sometimes this resolves the problem. Usually, however, there is no problem and startgrid() is not run by default. There can be benefits from running startgrid() to get better starting values even when starting values are feasible.

startgrid() is a brute-force approach that tries various values for variances and covariances and chooses the ones that work best. You may already be using a default form of startgrid() without knowing it. If you see meglm displaying Grid node 1, Grid node 2, … following Grid node 0 in the iteration log, that is meglm doing a default search because the original starting values were not feasible. The default form tries 0.1, 1, and 10 for all variances of all random effects.

startgrid(*numlist*) specifies values to try for variances of random effects.

startgrid(*covspec*) specifies the particular variances and covariances in which grid searches are to be performed. *covspec* is *name*[*level*] for variances and *name1*[*level*]**name2*[*level*] for covariances. For example, the variance of the random intercept at level id is specified as _cons[id], and the variance of the random slope on variable week at the same level is specified as week[id]. The error variance for the linear mixed-effects model is specified as e.*depvar*, where *depvar* is the name of the dependent variable. The covariance between the random slope and the random intercept above is specified as _cons[id]*week[id].

startgrid(*numlist covspec*) combines the two syntaxes. You may also specify startgrid() multiple times so that you can search the different ranges for different variances and covariances.

noestimate specifies that the model is not to be fit. Instead, starting values are to be shown (as modified by the above options if modifications were made), and they are to be shown using the coeflegend style of output.

dnumerical specifies that during optimization, the gradient vector and Hessian matrix be computed using numerical techniques instead of analytical formulas. By default, analytical formulas for computing the gradient and Hessian are used for all integration methods except intmethod(laplace).

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

For a general introduction to me commands, see [ME] **me**. For additional examples of mixed-effects models for binary and binomial outcomes, see [ME] **melogit**, [ME] **meprobit**, and [ME] **mecloglog**. For additional examples of mixed-effects models for ordinal responses, see [ME] **meologit** and [ME] **meoprobit**. For additional examples of mixed-effects models for multinomial outcomes, see [SEM] **Example 41g**. For additional examples of mixed-effects models for count outcomes, see [ME] **mepoisson** and [ME] **menbreg**. For additional examples of mixed-effects parametric survival models, see [ME] **mestreg**. For additional examples of mixed-effects models for censored outcomes, see [ME] **metobit** and [ME] **meintreg**.

Remarks are presented under the following headings:

Introduction
Two-level models for continuous responses
Two-level models for nonlinear responses
Three-level models for nonlinear responses
Crossed-effects models
Obtaining better starting values
Survey data
Video example

Introduction

meglm fits multilevel mixed-effects generalized linear models of the form

$$g\{E(\mathbf{y}|\mathbf{X},\mathbf{u})\} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim F \tag{1}$$

where $\mathbf{y}$ is the $n \times 1$ vector of responses from the distributional family F, $\mathbf{X}$ is an $n \times p$ design/covariate matrix for the fixed effects $\boldsymbol{\beta}$, and $\mathbf{Z}$ is the $n \times q$ design/covariate matrix for the random effects $\mathbf{u}$. The $\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}$ part is called the linear predictor, and it is often denoted as $\boldsymbol{\eta}$. The linear predictor also contains the offset or exposure variable when offset() or exposure() is specified. $g(\cdot)$ is called the link function and is assumed to be invertible such that

$$E(\mathbf{y}|\mathbf{X},\mathbf{u}) = g^{-1}(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}) = H(\boldsymbol{\eta}) = \boldsymbol{\mu}$$

For notational convenience here and throughout this manual entry, we suppress the dependence of $\mathbf{y}$ on $\mathbf{X}$. Substituting various definitions for $g(\cdot)$ and F results in a wide array of models. For instance, if $\mathbf{y}$ is distributed as Gaussian (normal) and $g(\cdot)$ is the identity function, we have

$$E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim \text{normal}$$

or mixed-effects linear regression. If $g(\cdot)$ is the logit function and $\mathbf{y}$ is distributed as Bernoulli, we have

$$\text{logit}\{E(\mathbf{y})\} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim \text{Bernoulli}$$

or mixed-effects logistic regression. If $g(\cdot)$ is the natural log function and $\mathbf{y}$ is distributed as Poisson, we have

$$\ln\{E(\mathbf{y})\} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}, \qquad \mathbf{y} \sim \text{Poisson}$$

or mixed-effects Poisson regression. In fact, some combinations of families and links are so common that we implemented them as separate commands in terms of meglm.

Command	meglm equivalent
melogit	family(bernoulli) link(logit)
meprobit	family(bernoulli) link(probit)
mecloglog	family(bernoulli) link(cloglog)
meologit	family(ordinal) link(logit)
meoprobit	family(ordinal) link(probit)
mepoisson	family(poisson) link(log)
menbreg	family(nbinomial) link(log)

When no family–link combination is specified, meglm defaults to a Gaussian family with an identity link. Thus meglm can be used to fit linear mixed-effects models; however, for those models we recommend using the more specialized mixed, which, in addition to meglm capabilities, allows for modeling of the structure of the level-one errors; see [ME] **mixed** for details.

The random effects $\mathbf{u}$ are assumed to be distributed as multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated (although they may be predicted), but instead are characterized by the variance components, the elements of $\mathbf{G} = \mathrm{Var}(\mathbf{u})$.

The general forms of the design matrices $\mathbf{X}$ and $\mathbf{Z}$ allow estimation for a broad class of generalized mixed-effects models: blocked designs, split-plot designs, growth curves, multilevel or hierarchical designs, etc. They also allow a flexible method of modeling within-cluster correlation. Subjects within the same cluster can be correlated as a result of a shared random intercept, or through a shared random slope on a covariate, or both. The general specification of variance components also provides additional flexibility—the random intercept and random slope could themselves be modeled as independent, or correlated, or independent with equal variances, and so forth.

Comprehensive treatments of mixed models are provided by, for example, Searle, Casella, and McCulloch (1992); Verbeke and Molenberghs (2000); Raudenbush and Bryk (2002); Hedeker and Gibbons (2006); McCulloch, Searle, and Neuhaus (2008); and Rabe-Hesketh and Skrondal (2022).

The key to fitting mixed models lies in estimating the variance components, and for that there exist many methods; see, for example, Breslow and Clayton (1993); Lin and Breslow (1996); Bates and Pinheiro (1998); and Ng et al. (2006). meglm uses maximum likelihood (ML) to estimate model parameters. The ML estimates are based on the usual application of likelihood theory, given the distributional assumptions of the model.

Returning to (1): in clustered-data situations, it is convenient not to consider all n observations at once but instead to organize the mixed model as a series of M independent groups (or clusters)

$$g\{E(\mathbf{y}_j)\} = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j \tag{2}$$

for $j = 1, \ldots, M$, with cluster j consisting of n_j observations. The response $\mathbf{y}_j$ comprises the rows of $\mathbf{y}$ corresponding with the jth cluster, with $\mathbf{X}_j$ defined analogously. The random effects $\mathbf{u}_j$ can now be thought of as M realizations of a $q \times 1$ vector that is normally distributed with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The matrix $\mathbf{Z}_i$ is the $n_j \times q$ design matrix for the jth cluster random effects. Relating this to (1), note that

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_M \end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_M \end{bmatrix}; \quad \mathbf{G} = \mathbf{I}_M \otimes \mathbf{\Sigma}$$

where $\mathbf{I}_M$ is the $M \times M$ identity matrix and $\otimes$ is the Kronecker product.

The mixed-model formula (2) is from Laird and Ware (1982) and offers two key advantages. First, it makes specifications of random-effects terms easier. If the clusters are schools, you can simply specify a random effect at the school level, as opposed to thinking of what a school-level random effect would mean when all the data are considered as a whole (if it helps, think Kronecker products). Second, representing a mixed-model with (2) generalizes easily to more than one set of random effects. For example, if classes are nested within schools, then (2) can be generalized to allow random effects at both the school and the class-within-school levels.

Two-level models for continuous responses

We begin with a simple application of (2).

▷ Example 1: Two-level linear mixed model

Consider a longitudinal dataset, used by both Ruppert, Wand, and Carroll (2003) and Diggle et al. (2002), consisting of `weight` measurements of 48 pigs on 9 successive `weeks`. Pigs are identified by the variable `id`. Each pig experiences a linear trend in growth but overall weight measurements vary from pig to pig. Because we are not really interested in these particular 48 pigs per se, we instead treat them as a random sample from a larger population and model the between-pig variability as a random effect, or in the terminology of (2), as a random-intercept term at the pig level. We thus wish to fit the model

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_j + \epsilon_{ij}$$

for $i = 1, \ldots, 9$ weeks and $j = 1, \ldots, 48$ pigs. The fixed portion of the model, $\beta_0 + \beta_1 \texttt{week}_{ij}$, simply states that we want one overall regression line representing the population average. The random effect u_j serves to shift this regression line up or down according to each pig. Because the random effects occur at the pig level (`id`), we fit the model by typing

```
. use https://www.stata-press.com/data/r19/pig
(Longitudinal analysis of pig weights)
. meglm weight week || id:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -1251.2506
Iteration 1:  Log likelihood = -1251.2506
Refining starting values:
Grid node 0:  Log likelihood = -1150.6253
Fitting full model:
Iteration 0:  Log likelihood = -1150.6253  (not concave)
Iteration 1:  Log likelihood = -1036.1793
Iteration 2:  Log likelihood =  -1017.912
Iteration 3:  Log likelihood = -1014.9537
Iteration 4:  Log likelihood = -1014.9268
Iteration 5:  Log likelihood = -1014.9268
Mixed-effects GLM                               Number of obs     =        432
Family: Gaussian
Link:   Identity
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(1)      =   25337.48
Log likelihood = -1014.9268                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0390124	159.18	0.000	6.133433	6.286359
_cons	19.35561	.5974047	32.40	0.000	18.18472	20.52651
id						
var(_cons)	14.81745	3.124202			9.801687	22.39989
var(e.weight)	4.383264	.3163349			3.805112	5.049261

```
LR test vs. linear model: chibar2(01) = 472.65        Prob >= chibar2 = 0.0000
```

At this point, a guided tour of the model specification and output is in order:

1. By typing `weight week`, we specified the response, `weight`, and the fixed portion of the model in the same way that we would if we were using `regress` or any other estimation command. Our fixed effects are a coefficient on `week` and a constant term.
2. When we added `|| id:`, we specified random effects at the level identified by the group variable `id`, that is, the pig level (level two). Because we wanted only a random intercept, that is all we had to type.
3. The estimation log displays a set of iterations from optimizing the log likelihood. By default, these are Newton–Raphson iterations, but other methods are available by specifying the appropriate *maximize_options*; see [R] **Maximize**.

4. The header describes the model, presents a summary of the random-effects group, reports the integration method used to fit the model, and reports a Wald test against the null hypothesis that all the coefficients on the independent variables in the mean equation are 0. Here the null hypothesis is rejected at all conventional levels. You can suppress the group information with the `nogroup` or the `noheader` option, which will suppress the rest of the header as well.
5. The estimation table reports the fixed effects, followed by the random effects, followed by the overall error term.
   a. For the fixed-effects part, we estimate $\beta_0 = 19.36$ and $\beta_1 = 6.21$.
   b. The random-effects equation is labeled `id`, meaning that these are random effects at the `id` (pig) level. We have only one random effect at this level, the random intercept. The variance of the level-two errors, σ_u^2, is estimated as 14.82 with standard error 3.12.
   c. The row labeled `var(e.weight)` displays the estimated variance of the overall error term: $\hat{\sigma}_\epsilon^2 = 4.38$. This is the variance of the level-one errors.
6. Finally, a likelihood-ratio test comparing the model with ordinary linear regression is provided and is highly significant for these data. See *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

◁

See *Remarks and examples* in [ME] **mixed** for further analysis of these data including a random-slope model and a model with an unstructured covariance structure.

Two-level models for nonlinear responses

By specifying different family–link combinations, we can fit a variety of mixed-effects models for nonlinear responses. Here we replicate one of the models from example 2 of `melogit`.

▷ Example 2: Two-level logistic regression model

Ng et al. (2006) analyzed a subsample of data from the 1989 Bangladesh fertility survey (Huq and Cleland 1990), which polled 1,934 Bangladeshi women on their use of contraception. The women sampled were from 60 districts, identified by the variable `district`. Each district contained either urban or rural areas (variable `urban`) or both. The variable `c_use` is the binary response, with a value of 1 indicating contraceptive use. Other covariates include mean-centered `age` and a factor variable for the number of `children`.

We fit a standard logistic regression model, amended to have a random intercept for each district and a random slope on the urban factor variable. We fit the model by typing

```
. use https://www.stata-press.com/data/r19/bangladesh
(Bangladesh Fertility Survey, 1989)
. meglm c_use i.urban age i.children
> || district: i.urban, family(bernoulli) link(logit) nofvlabel
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -1229.5485
Iteration 1:  Log likelihood = -1228.5268
Iteration 2:  Log likelihood = -1228.5263
Iteration 3:  Log likelihood = -1228.5263
Refining starting values:
Grid node 0:  Log likelihood = -1215.8592
Fitting full model:
Iteration 0:  Log likelihood = -1215.8592  (not concave)
Iteration 1:  Log likelihood = -1209.6285
Iteration 2:  Log likelihood = -1205.7903
Iteration 3:  Log likelihood = -1205.1337
Iteration 4:  Log likelihood = -1205.0034
Iteration 5:  Log likelihood = -1205.0025
Iteration 6:  Log likelihood = -1205.0025
Mixed-effects GLM                               Number of obs     =      1,934
Family: Bernoulli
Link:   Logit
Group variable: district                        Number of groups  =         60
                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(5)      =      97.30
Log likelihood = -1205.0025                     Prob > chi2       =     0.0000
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
1.urban	.7143927	.1513595	4.72	0.000	.4177335	1.011052
age	-.0262261	.0079656	-3.29	0.001	-.0418384	-.0106138
children						
1	1.128973	.1599347	7.06	0.000	.815507	1.442439
2	1.363165	.1761804	7.74	0.000	1.017857	1.708472
3	1.352238	.1815608	7.45	0.000	.9963853	1.708091
_cons	-1.698137	.1505019	-11.28	0.000	-1.993115	-1.403159
district						
var(1.urban)	.2741013	.2131525			.059701	1.258463
var(_cons)	.2390807	.0857012			.1184191	.4826891

```
LR test vs. logistic model: chi2(2) = 47.05                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Because we did not specify a covariance structure for the random effects $(u_{1j}, u_{0j})'$, `meglm` used the default independent structure:

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_{1j} \\ u_{0j} \end{bmatrix} = \begin{bmatrix} \sigma^2_{u1} & 0 \\ 0 & \sigma^2_{u0} \end{bmatrix}$$

with $\hat{\sigma}^2_{u1} = 0.27$ and $\hat{\sigma}^2_{u0} = 0.24$. You can request a different covariance structure by specifying the `covariance()` option. See examples 1–3 in `melogit` for further analysis of these data, and see [ME] **me** and [ME] **mixed** for further examples of covariance structures.

◁

Three-level models for nonlinear responses

Two-level models extend naturally to models with three or more levels with nested random effects. Here we replicate the model from example 2 of [ME] **meologit**.

▷ Example 3: Three-level ordered logistic regression model

We use the data from the Television, School, and Family Smoking Prevention and Cessation Project (Flay et al. 1988; Rabe-Hesketh and Skrondal 2022, chap. 11), where schools were randomly assigned into one of four groups defined by two treatment variables. Students within each school are nested in classes, and classes are nested in schools. The dependent variable is the tobacco and health knowledge (THK) scale score collapsed into four ordered categories. We regress the outcome on the treatment variables, social resistance classroom curriculum and TV intervention, and their interaction and control for the pretreatment score.

```
. use https://www.stata-press.com/data/r19/tvsfpors
(Television, School, and Family Project)
. meglm thk prethk cc##tv || school: || class:, family(ordinal) link(logit)
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood =  -2125.509
Iteration 2:  Log likelihood = -2125.1034
Iteration 3:  Log likelihood = -2125.1032
Refining starting values:
Grid node 0:  Log likelihood = -2152.1514
Fitting full model:
Iteration 0:  Log likelihood = -2152.1514  (not concave)
Iteration 1:  Log likelihood = -2125.9213  (not concave)
Iteration 2:  Log likelihood = -2120.1861
Iteration 3:  Log likelihood = -2115.6177
Iteration 4:  Log likelihood = -2114.5896
Iteration 5:  Log likelihood = -2114.5881
Iteration 6:  Log likelihood = -2114.5881
Mixed-effects GLM                               Number of obs     =      1,600
Family: Ordinal
Link:   Logit
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
school	28	18	57.1	137
class	135	1	11.9	28

```
Integration method: mvaghermite                 Integration pts.  =         7
                                                Wald chi2(4)      =    124.39
Log likelihood = -2114.5881                     Prob > chi2       =    0.0000
```

thk	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
prethk	.4085273	.039616	10.31	0.000	.3308814	.4861731
1.cc	.8844369	.2099124	4.21	0.000	.4730161	1.295858
1.tv	.236448	.2049065	1.15	0.249	-.1651614	.6380575
cc#tv						
1 1	-.3717699	.2958887	-1.26	0.209	-.951701	.2081612
/cut1	-.0959459	.1688988			-.4269815	.2350896
/cut2	1.177478	.1704946			.8433151	1.511642
/cut3	2.383672	.1786736			2.033478	2.733865
school						
var(_cons)	.0448735	.0425387			.0069997	.2876749
school>class						
var(_cons)	.1482157	.0637521			.063792	.3443674

```
LR test vs. ologit model: chi2(2) = 21.03                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Notes:

1. Our model now has two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `school` level (level three), and the second is a random intercept at the `class` level (level two). The order in which these are specified (from left to right) is significant—`meglm` assumes that `class` is nested within `school`.
2. The information on groups is now displayed as a table, with one row for each grouping. You can suppress this table with the `nogroup` or the `noheader` option, which will suppress the rest of the header, as well.
3. The variance-component estimates are now organized and labeled according to level. The variance component for `class` is labeled `school>class` to emphasize that classes are nested within schools.

We refer you to example 2 of [ME] **meologit** and example 1 of [ME] **meologit postestimation** for a substantive interpretation of the results.

◁

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by `||`. The order of nesting goes from left to right as the groups go from biggest (highest level) to smallest (lowest level).

Crossed-effects models

Not all mixed models contain nested levels of random effects. In this section, we consider a crossed-effects model, that is, a mixed-effects model in which the levels of random effects are not nested; see [ME] **me** for more information on crossed-effects models.

▷ Example 4: Crossed-effects logistic regression model

We use the salamander cross-breeding data from Karim and Zeger (1992) as analyzed in Rabe-Hesketh and Skrondal (2022, sec. 16.8). The salamanders come from two populations—whiteside and roughbutt—and are labeled whiteside males (`wsm`), whiteside females (`wsf`), roughbutt males (`rbm`), and roughbutt females (`rbf`). Male identifiers are recorded in the variable `male`, and female identifiers are recorded in the variable `female`. The salamanders were divided into groups such that each group contained 60 male–female pairs, with each salamander having three potential partners from the same population and three potential partners from the other population. The outcome (`y`) is coded 1 if there was a successful mating and is coded 0 otherwise; see the references for a detailed description of the mating experiment.

We fit a crossed-effects logistic regression for successful mating, where each male has the same value of his random intercept across all females, and each female has the same value of her random intercept across all males.

To fit a crossed-effects model in Stata, we use the `_all:` `R.`*varname* syntax. We treat the entire dataset as one super cluster, denoted `_all`, and we nest each gender within the super cluster by using the `R.`*varname* notation. `R.male` requests a random intercept for each level of `male` and imposes an identity covariance structure on the random effects; that is, the variances of the random intercepts are restricted to be equal for all male salamanders. `R.female` accomplishes the same for the female salamanders. In Stata, we type

```
. use https://www.stata-press.com/data/r19/salamander
. meglm y wsm##wsf || _all: R.male || _all: R.female, family(bernoulli)
> link(logit) or
note: crossed random-effects model specified; option intmethod(laplace)
      implied.
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -223.13998
Iteration 1:  Log likelihood = -222.78752
Iteration 2:  Log likelihood = -222.78735
Iteration 3:  Log likelihood = -222.78735
Refining starting values:
Grid node 0:  Log likelihood = -211.58149
Fitting full model:
Iteration 0:  Log likelihood = -211.58149
Iteration 1:  Log likelihood = -209.33737
Iteration 2:  Log likelihood = -209.29378  (not concave)
Iteration 3:  Log likelihood = -209.29291
Iteration 4:  Log likelihood = -209.27663
Iteration 5:  Log likelihood = -209.27659
Iteration 6:  Log likelihood = -209.27659
Mixed-effects GLM                               Number of obs     =        360
Family: Bernoulli
Link:   Logit
Group variable: _all                            Number of groups  =          1
                                                Obs per group:
                                                              min =        360
                                                              avg =      360.0
                                                              max =        360
Integration method: laplace
                                                Wald chi2(3)      =      42.59
Log likelihood = -209.27659                     Prob > chi2       =     0.0000
```

y	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
1.wsm	.4955657	.2293702	-1.52	0.129	.2000446	1.227653
1.wsf	.0547918	.0287903	-5.53	0.000	.0195638	.1534542
wsm#wsf						
1 1	36.17797	22.01912	5.90	0.000	10.97424	119.2652
_cons	2.740748	1.062625	2.60	0.009	1.281878	5.859918
_all>male						
var(_cons)	1.041005	.4998442			.4062035	2.667853
_all>female						
var(_cons)	1.17438	.5438465			.4738309	2.910675

```
Note: Estimates are transformed only in the first equation to odds ratios.
Note: _cons estimates baseline odds (conditional on zero random effects).
LR test vs. logistic model: chi2(2) = 27.02                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Because we specified a crossed-effects model, `meglm` defaulted to the method of Laplacian approximation to calculate the likelihood; see *Computation time and the Laplacian approximation* in [ME] **me** for a discussion of computational complexity of mixed-effects models, and see *Methods and formulas* below for the formulas used by the Laplacian approximation method.

The estimates of the random intercepts suggest that the heterogeneity among the female salamanders, 1.17, is larger than the heterogeneity among the male salamanders, 1.04.

Setting both random intercepts to 0, the odds of successful mating for a roughbutt male–female pair are given by the estimate of _cons, 2.74. Rabe-Hesketh and Skrondal (2022, sec. 16.8) show how to calculate the odds ratios for the other three salamander pairings.

◁

The R.*varname* notation is equivalent to giving a list of overparameterized (none dropped) indicator variables for use in a random-effects specification. When you specify R.*varname*, meglm handles the calculations internally rather than creating the indicators in the data. Because the set of indicators is overparameterized, R.*varname* implies noconstant.

❑ Technical note

We fit the salamander model by using

```
. meglm y wsm##wsf || _all: R.male || _all: R.female ...
```

as a direct way to demonstrate the R. notation. However, we can technically treat female salamanders as nested within the _all group, yielding the equivalent way to fit the model:

```
. meglm y wsm##wsf || _all: R.male || female: ...
```

We leave it to you to verify that both produce identical results. As we note in example 8 of [ME] **me**, the latter specification, organized at the cluster (female) level with random-effects dimension one (a random intercept) is, in general, much more computationally efficient.

❑

Obtaining better starting values

Given the flexibility of mixed-effects models, you will find that some models "fail to converge" when used with your data; see *Diagnosing convergence problems* in [ME] **me** for details. What we say below applies regardless of how the convergence problem revealed itself. You might have seen the error message "initial values not feasible" or some other error message, or you might have an infinite iteration log.

meglm provides two options to help you obtain better starting values: startvalues() and startgrid().

startvalues(*svmethod*) allows you to specify one of four starting-value calculation methods: zero, constantonly, fixedonly, or iv. By default, meglm uses startvalues(fixedonly). Evidently, that did not work for you. Try the other methods, starting with startvalues(iv):

```
. meglm ..., ... startvalues(iv)
```

If that does not solve the problem, proceed through the others.

By the way, if you have starting values for some parameters but not others—perhaps you fit a simplified model to get them—you can combine the options startvalues() and from():

```
. meglm ..., ...                          // simplified model
. matrix b = e(b)
. meglm ..., ... from(b) startvalues(iv)  // full model
```

The other special option meglm provides is startgrid(), which can be used with or without startvalues(). startgrid() is a brute-force approach that tries various values for variances and covariances and chooses the ones that work best.

1. You may already be using a default form of startgrid() without knowing it. If you see meglm displaying Grid node 1, Grid node 2, ... following Grid node 0 in the iteration log, that is meglm doing a default search because the original starting values were not feasible.

   The default form tries 0.1, 1, and 10 for all variances of all random effects and, if applicable, for the error variance.

2. startgrid(*numlist*) specifies values to try for variances of random effects.

3. startgrid(*covspec*) specifies the particular variances and covariances in which grid searches are to be performed. Variances and covariances are specified in the usual way. startgrid(_cons[id] x[id] _cons[id]*x[id]) specifies that 0.1, 1, and 10 be tried for each member of the list.

4. startgrid(*numlist covspec*) combines the two syntaxes. You can specify startgrid() multiple times so that you can search the different ranges for different variances and covariances.

Our advice to you is the following:

1. If you receive an iteration log and it does not contain Grid node 1, Grid node 2, ..., then specify startgrid(.1 1 10). Do that whether the iteration log was infinite or ended with some other error. In this case, we know that meglm did not run startgrid() on its own because it did not report Grid node 1, Grid node 2, etc. Your problem is poor starting values, not infeasible ones.

   A synonym for startgrid(.1 1 10) is just startgrid without parentheses.

   Be careful, however, if you have many random effects. Specifying startgrid() could run a long time because it runs all possible combinations. If you have 10 random effects, that means $10^3 = 1{,}000$ likelihood evaluations.

   If you have many random effects, rerun your difficult meglm command including option iterate(#) and look at the results. Identify the problematic variances and search across them only. Do not just look for variances going to 0. Variances getting really big can be a problem, too, and even reasonable values can be a problem. Use your knowledge and intuition about the model.

   Perhaps you will try to fit your model by specifying startgrid(.1 1 10 _cons[id] x[id] _cons[id]*x[id]).

   Values 0.1, 1, and 10 are the default. Equivalent to specifying startgrid(.1 1 10 _cons[id] x[id] _cons[id]*x[id]) is startgrid(_cons[id] x[id] _cons[id]*x[id]).

   Look at covariances as well as variances. If you expect a covariance to be negative but it is positive, then try negative starting values for the covariance by specifying startgrid(-.1 -1 -10 _cons[id]*x[id]).

   Remember that you can specify startgrid() multiple times. Thus you might specify both startgrid(_cons[id] x[id]) and startgrid(-.1 -1 -10 _cons[id]*x[id]).

2. If you receive the message "initial values not feasible", you know that `meglm` already tried the default `startgrid()`.

   The default `startgrid()` only tried the values 0.1, 1, and 10, and only tried them on the variances of random effects. You may need to try different values or try the same values on covariances or variances of errors of observed endogenous variables.

   We suggest you first rerun the model causing difficulty and include the `noestimate` option. If, looking at the results, you have an idea of which variance or covariance is a problem, or if you have few variances and covariances, we would recommend running `startgrid()` first. On the other hand, if you have no idea as to which variance or covariance is the problem and you have many of them, you will be better off if you first simplify the model. After doing that, if your simplified model does not include all the variances and covariances, you can specify a combination of `from()` and `startgrid()`.

Survey data

Multilevel modeling of survey data is a little different from standard modeling in that weighted sampling can take place at multiple levels in the model, resulting in multiple sampling weights. Most survey datasets, regardless of the design, contain one overall inclusion weight for each observation in the data. This weight reflects the inverse of the probability of ultimate selection, and by "ultimate" we mean that it factors in all levels of clustered sampling, corrections for noninclusion and oversampling, poststratification, etc.

For simplicity, in what follows, assume a simple two-stage sampling design where groups are randomly sampled and then individuals within groups are sampled. Also assume that no additional weight corrections are performed; that is, sampling weights are simply the inverse of the probability of selection. The sampling weight for observation i in cluster j in our two-level sample is then $w_{ij} = 1/\pi_{ij}$, where π_{ij} is the probability that observation i, j is selected. If you were performing a standard analysis such as OLS regression with `regress`, you would simply use a variable holding w_{ij} as your `pweight` variable, and the fact that it came from two levels of sampling would not concern you. Perhaps you would type `vce(cluster` *groupvar*`)` where *groupvar* identifies the top-level groups to get standard errors that control for correlation within these groups, but you would still use only one weight variable.

Now take these same data and fit a two-level model with `meglm`. As seen in (5) in *Methods and formulas* later in this entry, it is not sufficient to use the single sampling weight w_{ij}, because weights enter the log likelihood at both the group level and the individual level. Instead, what is required for a two-level model under this sampling design is w_j, the inverse of the probability that group j is selected in the first stage, and $w_{i|j}$, the inverse of the probability that individual i from group j is selected at the second stage conditional on group j already being selected. You cannot use w_{ij} without making any assumptions about w_j.

Given the rules of conditional probability, $w_{ij} = w_j w_{i|j}$. If your dataset has only w_{ij}, then you will need to either assume equal probability sampling at the first stage ($w_j = 1$ for all j) or find some way to recover w_j from other variables in your data; see Rabe-Hesketh and Skrondal (2006) and the references therein for some suggestions on how to do this, but realize that there is little yet known about how well these approximations perform in practice.

What you really need to fit your two-level model are data that contain w_j in addition to either w_{ij} or $w_{i|j}$. If you have w_{ij}—that is, the unconditional inclusion weight for observation i, j—then you need to divide w_{ij} by w_j to obtain $w_{i|j}$.

▷ Example 5: Two-level logistic regression model with weights

Rabe-Hesketh and Skrondal (2006) analyzed data from the 2000 Programme for International Student Assessment (PISA) study on reading proficiency among 15-year-old American students, as performed by the Organisation for Economic Co-operation and Development (OECD). The original study was a three-stage cluster sample, where geographic areas were sampled at the first stage, schools at the second, and students at the third. Our version of the data does not contain the geographic-areas variable, so we treat this as a two-stage sample where schools are sampled at the first stage and students at the second.

```
. use https://www.stata-press.com/data/r19/pisa2000
(Programme for International Student Assessment (PISA) 2000 data)
. describe
Contains data from https://www.stata-press.com/data/r19/pisa2000.dta
 Observations:         2,069                  Programme for International
                                                Student Assessment (PISA) 2000
                                                data
    Variables:            11                  12 Jun 2024 10:08
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
female          byte    %8.0g                 1 if female
isei            byte    %8.0g                 International socioeconomic index
w_fstuwt        float   %9.0g                 Student-level weight
wnrschbw        float   %9.0g                 School-level weight
high_school     byte    %8.0g                 1 if highest level by either
                                                parent is high school
college         byte    %8.0g                 1 if highest level by either
                                                parent is college
one_for         byte    %8.0g                 1 if one parent foreign born
both_for        byte    %8.0g                 1 if both parents are foreign
                                                born
test_lang       byte    %8.0g                 1 if English (the test language)
                                                is spoken at home
pass_read       byte    %8.0g                 1 if passed reading proficiency
                                                threshold
id_school       int     %8.0g                 School ID
------------------------------------------------------------------------------
Sorted by:
```

For student i in school j, where the variable `id_school` identifies the schools, the variable `w_fstuwt` is a student-level overall inclusion weight (w_{ij}, not $w_{i|j}$) adjusted for noninclusion and nonparticipation of students, and the variable `wnrschbw` is the school-level weight w_j adjusted for oversampling of schools with more minority students. The weight adjustments do not interfere with the methods prescribed above, and thus we can treat the weight variables simply as w_{ij} and w_j, respectively.

Rabe-Hesketh and Skrondal (2006) fit a two-level logistic model for passing a reading proficiency threshold. We will do the same using `meglm`, but first we must reproduce the "method 1" adjusted weight variables that were used. The "method 1" adjustment scales the first-level weights so that they sum to the effective sample size of their corresponding second-level cluster.

```
. sort id_school
. generate sqw = w_fstuwt * w_fstuwt
. by id_school: egen sumw = sum(w_fstuwt)
. by id_school: egen sumsqw = sum(sqw)
. generate pst1s1 = w_fstuwt*sumw/sumsqw
```

The new variable pst1s1 holds the adjusted first-level weights. Rabe-Hesketh and Skrondal (2006) also included the school mean socioeconomic index as a covariate in their analysis. We reproduce this variable using egen.

```
. by id_school: egen mn_isei = mean(isei)
```

Here is the fitted model:

```
. meglm pass_read female isei mn_isei high_school college test_lang one_for
> both_for [pw=pst1s1], family(bernoulli) link(logit)
> || id_school:, pweight(wnrschbw)
```

(*output omitted*)

```
Mixed-effects GLM                               Number of obs     =      2,069
Family: Bernoulli
Link:   Logit
Group variable: id_school                       Number of groups  =        148
                                                Obs per group:
                                                              min =          1
                                                              avg =       14.0
                                                              max =         28
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(8)      =      88.30
Log pseudolikelihood = -197395.98               Prob > chi2       =     0.0000
                          (Std. err. adjusted for 148 clusters in id_school)
```

pass_read	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
female	.6221369	.1540088	4.04	0.000	.3202852	.9239887
isei	.018215	.0048057	3.79	0.000	.0087959	.027634
mn_isei	.0682472	.0164337	4.15	0.000	.0360378	.1004566
high_school	.1028108	.477141	0.22	0.829	-.8323683	1.03799
college	.4531688	.5053447	0.90	0.370	-.5372885	1.443626
test_lang	.6251822	.3821182	1.64	0.102	-.1237557	1.37412
one_for	-.1089314	.2739724	-0.40	0.691	-.6459075	.4280447
both_for	-.2804038	.3264681	-0.86	0.390	-.9202696	.359462
_cons	-5.877565	.954525	-6.16	0.000	-7.7484	-4.006731
id_school						
var(_cons)	.2955769	.1243375			.1295996	.6741201

Notes:

1. We specified the level-one weights using standard Stata weight syntax, that is, [pw=pst1s1].
2. We specified the level-two weights via the pweight(wnrschbw) option as part of the random-effects specification for the id_school level. As such, it is treated as a school-level weight. Accordingly, wnrschbw needs to be constant within schools, and meglm did check for that before estimating.
3. As is the case with other estimation commands in Stata, standard errors in the presence of sampling weights are robust.
4. Robust standard errors are clustered at the top level of the model, and this will always be true unless you specify vce(cluster *clustvar*), where *clustvar* identifies an even higher level of grouping.

◁

▷ Example 6: Two-level logistic regression model with survey weights

meglm also supports the svy prefix (see [SVY] **svy**) for the linearized variance estimator. Here we refit the model from the previous example using the svy prefix after we svyset (see [SVY] **svyset**) the survey design variables.

```
. svyset id_school, weight(wnrschbw) || _n, weight(pst1s1)
note: stage 1 is sampled with replacement; further stages will be ignored for
      variance estimation.

Sampling weights: <none>
             VCE: linearized
      Subpop. df: off
     Single unit: missing
        Strata 1: <one>
 Sampling unit 1: id_school
           FPC 1: <zero>
        Weight 1: wnrschbw
        Strata 2: <one>
 Sampling unit 2: <observations>
           FPC 2: <zero>
        Weight 2: pst1s1

. svy: meglm pass_read female isei mn_isei high_school college test_lang
> one_for both_for, family(bernoulli) link(logit) || id_school:
(running meglm on estimation sample)

Survey: Mixed-effects GLM

Number of strata =   1                          Number of obs    =      2,069
Number of PSUs   = 148                          Population size = 346,373.74
                                                Design df        =        147
                                                F(8, 140)        =      10.51
                                                Prob > F         =     0.0000
```

pass_read	Coefficient	Linearized std. err.	t	P>|t|	[95% conf.	interval]
female	.6221369	.1540088	4.04	0.000	.3177796	.9264943
isei	.018215	.0048057	3.79	0.000	.0087177	.0277122
mn_isei	.0682472	.0164337	4.15	0.000	.0357704	.100724
high_school	.1028108	.477141	0.22	0.830	-.8401311	1.045753
college	.4531688	.5053447	0.90	0.371	-.5455101	1.451848
test_lang	.6251822	.3821182	1.64	0.104	-.1299725	1.380337
one_for	-.1089314	.2739724	-0.40	0.692	-.6503648	.432502
both_for	-.2804038	.3264681	-0.86	0.392	-.925581	.3647734
_cons	-5.877565	.954525	-6.16	0.000	-7.763929	-3.991201
id_school						
var(_cons)	.2955769	.1243375			.1287156	.6787495

Notes:

1. We svyset the design variables: id_school is the PSU variable, wnrschbw contains weights at the PSU level, _n specifies that the students are identified by the individual observations, and pst1s1 contains our adjusted student-level conditional weights.
2. svyset notes the lack of a finite population correction in the first stage and informs us that only the first-stage unit information will be used in the linearized variance estimator. However, the svy prefix will still pass the stage-two weights to meglm.

3. svy produces a different header, giving us an estimate of the population size, the design degrees of freedom, and the number of first-stage sampling units.

◁

Video example

Tour of multilevel GLMs

Stored results

meglm stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_dv)	number of dependent variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_cat)	number of categories (with ordinal outcomes)
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(N_clust)	number of clusters
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	gsem
e(cmd2)	meglm
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	name of marginal model
e(title)	title in estimation output
e(link)	link
e(family)	family
e(clustvar)	name of cluster variable
e(offset)	offset
e(binomial)	binomial number of trials (with binomial models)
e(dispersion)	mean or constant (with negative binomial models)

e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(cat)	category values (with ordinal outcomes)
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction
Gauss–Hermite quadrature
Adaptive Gauss–Hermite quadrature
Laplacian approximation
Survey data

Introduction

Without a loss of generality, consider a two-level generalized mixed-effects model

$$E(\mathbf{y}_j|\mathbf{X}_j,\mathbf{u}_j) = g^{-1}(\mathbf{X}_j\boldsymbol{\beta}+\mathbf{Z}_j\mathbf{u}_j), \qquad \mathbf{y}\sim F$$

for $j = 1,\ldots,M$ clusters, with the jth cluster consisting of n_j observations, where, for the jth cluster, $\mathbf{y}_j$ is the $n_j \times 1$ response vector, $\mathbf{X}_j$ is the $n_j \times p$ matrix of fixed predictors, $\mathbf{Z}_j$ is the $n_j \times q$ matrix of random predictors, $\mathbf{u}_j$ is the $q\times 1$ vector of random effects, $\boldsymbol{\beta}$ is the $p\times 1$ vector of regression coefficients on the fixed predictors, and we use $\boldsymbol{\Sigma}$ to denote the unknown $q\times q$ variance matrix of the random effects. For simplicity, we consider a model with no auxiliary parameters.

Let $\boldsymbol{\eta}_j$ be the linear predictor, $\boldsymbol{\eta}_j = \mathbf{X}_j\boldsymbol{\beta}+\mathbf{Z}_j\mathbf{u}_j$, that also includes the offset or the exposure variable when `offset()` or `exposure()` is specified. Let y_{ij} and η_{ij} be the ith individual elements of $\mathbf{y}_j$ and $\boldsymbol{\eta}_j$, $i = 1,\ldots,n_j$. Let $f(y_{ij}|\eta_{ij})$ be the conditional density function for the response at observation i. Because the observations are assumed to be conditionally independent, we can overload the definition of $f(\cdot)$ with vector inputs to mean

$$\log f(\mathbf{y}_j|\boldsymbol{\eta}_j) = \sum_{j=1}^{n_i} \log f(y_{ij}|\eta_{ij})$$

The random effects $\mathbf{u}_j$ are assumed to be multivariate normal with mean $\mathbf{0}$ and variance $\boldsymbol{\Sigma}$. The likelihood function for cluster j is given by

$$\begin{aligned}\mathcal{L}_j(\boldsymbol{\beta},\boldsymbol{\Sigma}) &= (2\pi)^{-q/2}|\boldsymbol{\Sigma}|^{-1/2}\int_{\Re^q} f(\mathbf{y}_j|\boldsymbol{\eta}_j)\exp\left(-\frac{1}{2}\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j\right)\,d\mathbf{u}_j \\ &= (2\pi)^{-q/2}|\boldsymbol{\Sigma}|^{-1/2}\int_{\Re^q} \exp\left\{\log f(\mathbf{y}_j|\boldsymbol{\eta}_j)-\frac{1}{2}\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j\right\}\,d\mathbf{u}_j\end{aligned} \tag{3}$$

where $\Re$ denotes the set of values on the real line and $\Re^q$ is the analog in q-dimensional space.

The multivariate integral in (3) is generally not tractable, so we must use numerical methods to approximate the integral. We can use a change-of-variables technique to transform this multivariate integral into a set of nested univariate integrals. Each univariate integral can then be evaluated using a form of Gaussian quadrature. `meglm` supports three types of Gauss–Hermite quadratures: mean–variance adaptive Gauss–Hermite quadrature, mode-curvature adaptive Gauss–Hermite quadrature, and nonadaptive Gauss–Hermite quadrature. `meglm` also offers the Laplacian-approximation method, which is used as a default method for crossed mixed-effects models. Below we describe the four methods. The methods described below are based on Skrondal and Rabe-Hesketh (2004, chap. 6.3).

Gauss–Hermite quadrature

Let $\mathbf{u}_j = \mathbf{L}\mathbf{v}_j$, where $\mathbf{v}_j$ is a $q\times 1$ random vector whose elements are independently standard normal variables and $\mathbf{L}$ is the Cholesky decomposition of $\boldsymbol{\Sigma}$, $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}'$. Then $\boldsymbol{\eta}_j = \mathbf{X}_j\boldsymbol{\beta}+\mathbf{Z}_j\mathbf{L}\mathbf{v}_j$, and the likelihood in (3) becomes

$$\begin{aligned}\mathcal{L}_j(\boldsymbol{\beta},\boldsymbol{\Sigma}) &= (2\pi)^{-q/2}\int_{\Re^q}\exp\left\{\log f(\mathbf{y}_j|\boldsymbol{\eta}_j)-\frac{1}{2}\mathbf{v}_j'\mathbf{v}_j\right\}\,d\mathbf{v}_j \\ &= (2\pi)^{-q/2}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}\exp\left\{\log f(\mathbf{y}_j|\boldsymbol{\eta}_j)-\frac{1}{2}\sum_{k=1}^{q}v_{jk}^2\right\}\,d\mathbf{v}_{j1},\ldots,d\mathbf{v}_{jq}\end{aligned} \tag{4}$$

Consider a q-dimensional quadrature grid containing r quadrature points in each dimension. Let $\mathbf{a_k} = (a_{k_1}, \ldots, a_{k_q})'$ be a point on this grid, and let $\mathbf{w_k} = (w_{k_1}, \ldots, w_{k_q})'$ be the vector of corresponding weights. The nonadaptive Gauss–Hermite quadrature approximation to the likelihood is

$$\begin{aligned}\mathcal{L}_j^{\text{GHQ}}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= \sum_{k_1=1}^{r} \cdots \sum_{k_q=1}^{r} \left[\exp\left\{ \log f(\mathbf{y}_j|\boldsymbol{\eta}_{j\mathbf{k}}) \right\} \prod_{p=1}^{q} w_{k_p} \right] \\ &= \sum_{k_1=1}^{r} \cdots \sum_{k_q=1}^{r} \left[\exp\left\{ \sum_{i=1}^{n_j} \log f(y_{ij}|\eta_{ij\mathbf{k}}) \right\} \prod_{p=1}^{q} w_{k_p} \right]\end{aligned}$$

where

$$\boldsymbol{\eta}_{j\mathbf{k}} = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{L}\mathbf{a_k}$$

and $\eta_{ij\mathbf{k}}$ is the ith element of $\boldsymbol{\eta}_{j\mathbf{k}}$.

Adaptive Gauss–Hermite quadrature

This section sets the stage for mean–variance adaptive Gauss–Hermite quadrature and mode-curvature adaptive Gauss–Hermite quadrature.

Let's reconsider the likelihood in (4). We use $\phi(\mathbf{v}_j)$ to denote a multivariate standard normal with mean $\mathbf{0}$ and variance $\mathbf{I}_q$, and we use $\phi(\mathbf{v}_j|\boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j)$ to denote a multivariate normal with mean $\boldsymbol{\mu}_j$ and variance $\boldsymbol{\Lambda}_j$.

For fixed model parameters, the posterior density for $\mathbf{v}_j$ is proportional to

$$\phi(\mathbf{v}_j) f(\mathbf{y}_j|\boldsymbol{\eta}_j)$$

where

$$\boldsymbol{\eta}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{L}\mathbf{v}_j$$

It is reasonable to assume that this posterior density can be approximated by a multivariate normal density with mean vector $\boldsymbol{\mu}_j$ and variance matrix $\boldsymbol{\Lambda}_j$. Instead of using the prior density of $\mathbf{v}_j$ as the weighting distribution in the integral, we can use our approximation for the posterior density,

$$\mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \int_{\Re^q} \frac{f(\mathbf{y}_j|\boldsymbol{\eta}_j)\phi(\mathbf{v}_j)}{\phi(\mathbf{v}_j|\boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j)} \phi(\mathbf{v}_j|\boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j)\, d\mathbf{v}_j$$

Then the mean–variance adaptive Gauss–Hermite approximation to the likelihood is

$$\mathcal{L}_j^{\text{MVAGH}}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \sum_{k_1=1}^{r} \cdots \sum_{k_q=1}^{r} \left[\exp\left\{ \log f(\mathbf{y}_j|\boldsymbol{\eta}_{j\mathbf{k}}) \right\} \prod_{p=1}^{q} w^*_{jk_p} \right]$$

where

$$\boldsymbol{\eta}_{j\mathbf{k}} = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{L}\mathbf{a}^*_{j\mathbf{k}}$$

and $\mathbf{a}^*_{j\mathbf{k}}$ and $w^*_{jk_p}$ are the abscissas and weights after an orthogonalizing transformation of $\mathbf{a}_{j\mathbf{k}}$ and w_{jk_p}, respectively, which eliminates posterior covariances between the random effects.

Estimates of $\boldsymbol{\mu}_j$ and $\boldsymbol{\Lambda}_j$ are computed using one of two different methods. The mean $\boldsymbol{\mu}_j$ and variance $\boldsymbol{\Lambda}_j$ are computed iteratively by updating the posterior moments with the mean–variance adaptive Gauss–Hermite approximation, starting with a $\mathbf{0}$ mean vector and identity variance matrix. For the mode-curvature adaptive Gauss–Hermite approximation, $\boldsymbol{\mu}_j$ and $\boldsymbol{\Lambda}_j$ are computed by optimizing the integrand with respect to $\mathbf{v}_j$, where $\boldsymbol{\mu}_j$ is the optimal value and $\boldsymbol{\Lambda}_j$ is the curvature at $\boldsymbol{\mu}_j$.

Laplacian approximation

Consider the likelihood in (3) and denote the argument in the exponential function by

$$h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j) = \log f(\mathbf{y}_j|\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j) - \frac{1}{2}\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j$$

The Laplacian approximation is based on a second-order Taylor expansion of $h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j)$ about the value of $\mathbf{u}_j$ that maximizes it. The first and second partial derivatives with respect to $\mathbf{u}_j$ are

$$h'(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j) = \frac{\partial h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j)}{\partial \mathbf{u}_j} = \mathbf{Z}_j' \frac{\partial \log f(\mathbf{y}_j|\boldsymbol{\eta}_j)}{\partial \boldsymbol{\eta}_j} - \boldsymbol{\Sigma}^{-1}\mathbf{u}_j$$

$$h''(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j) = \frac{\partial^2 h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j)}{\partial \mathbf{u}_j \partial \mathbf{u}_j'} = \mathbf{Z}_j' \frac{\partial^2 \log f(\mathbf{y}_j|\boldsymbol{\eta}_j)}{\partial \boldsymbol{\eta}_j \partial \boldsymbol{\eta}_j'} \mathbf{Z}_j - \boldsymbol{\Sigma}^{-1}$$

The maximizer of $h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j)$ is $\hat{\mathbf{u}}_j$ such that $h'(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j) = \mathbf{0}$. The integral in (3) is proportional to the posterior density of $\mathbf{u}_j$ given the data, so $\hat{\mathbf{u}}_j$ is also the posterior mode.

Pinheiro and Chao (2006) show that the posterior mode, $\hat{\mathbf{u}}_j$, and curvature, $h''(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j)^{-1}$, can be efficiently computed as the iterative solution to a least-squares problem by using matrix decomposition methods similar to those used in fitting linear mixed-effects models (Bates and Pinheiro 1998; Pinheiro and Bates 2000).

Let

$$\hat{\mathbf{p}}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\hat{\mathbf{u}}_j$$

$$\mathbf{S}_1 = \frac{\partial \log f(\mathbf{y}_j|\hat{\mathbf{p}}_j)}{\partial \hat{\mathbf{p}}_j}$$

$$\mathbf{S}_2 = \frac{\partial \mathbf{S}_1}{\partial \hat{\mathbf{p}}_j'} = \frac{\partial^2 \log f(\mathbf{y}_j|\hat{\mathbf{p}}_j)}{\partial \hat{\mathbf{p}}_j \partial \hat{\mathbf{p}}_j'}$$

$$\mathbf{H}_j = h''(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j) = \mathbf{Z}_j'\mathbf{S}_2\mathbf{Z}_j - \boldsymbol{\Sigma}^{-1}$$

then

$$\mathbf{0} = h'(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j) = \mathbf{Z}_j'\mathbf{S}_1 - \boldsymbol{\Sigma}^{-1}\hat{\mathbf{u}}_j$$

Given the above, the second-order Taylor approximation takes the form

$$h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j) \approx h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j) + \frac{1}{2}(\mathbf{u}_j - \hat{\mathbf{u}}_j)' \mathbf{H}_j (\mathbf{u}_j - \hat{\mathbf{u}}_j)$$

because the first-order derivative term is 0. The integral is approximated by

$$\int_{\Re^q} \exp\{h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j)\}\, d\mathbf{u}_j \approx (2\pi)^{q/2} \left|-\mathbf{H}_j\right|^{-1/2} \exp\{h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j)\}$$

Thus the Laplacian approximated log likelihood is

$$\log \mathcal{L}_j^{\text{Lap}}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\log\left|-\mathbf{H}_j\right| + h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \hat{\mathbf{u}}_j)$$

The log likelihood for the entire dataset is simply the sum of the contributions of the M individual clusters, namely, $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \sum_{j=1}^{M} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma})$.

Maximization of $\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\Sigma})$ is performed with respect to $(\boldsymbol{\beta}, \boldsymbol{\sigma}^2)$, where $\boldsymbol{\sigma}^2$ is a vector comprising the unique elements of $\boldsymbol{\Sigma}$. Parameter estimates are stored in `e(b)` as $(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\sigma}}^2)$, with the corresponding variance–covariance matrix stored in `e(V)`. In the presence of auxiliary parameters, their estimates and standard errors are included in `e(b)` and `e(V)`, respectively.

Survey data

In the presence of sampling weights, following Rabe-Hesketh and Skrondal (2006), the weighted log pseudolikelihood for a two-level model is given as

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \sum_{j=1}^{M} w_j \log \int_{-\infty}^{\infty} \exp\left\{\sum_{i=1}^{n_j} w_{i|j} \log f(y_{ij}|\eta_{ij})\right\} \phi(\mathbf{v}_{j1})\, d\mathbf{v}_{j1} \tag{5}$$

where w_j is the inverse of the probability of selection for the jth cluster; $w_{i|j}$ is the inverse of the conditional probability of selection of individual i, given the selection of cluster j, $f(\cdot)$ is as defined previously; and $\phi(\cdot)$ is the standard multivariate normal density.

Weighted estimation is achieved through the direct application of w_j and $w_{i|j}$ into the likelihood calculations as detailed above to reflect replicated clusters for w_j and replicated observations within clusters for $w_{i|j}$. Because this estimation is based on replicated clusters and observations, frequency weights are handled similarly.

Weights are not allowed with crossed models or the Laplacian approximation.

References

Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Bates, D. M., and J. C. Pinheiro. 1998. "Computational methods for multilevel modelling". In *Technical Memorandum BL0112140-980226-01TM.* Murray Hill, NJ: Bell Labs, Lucent Technologies.

Breslow, N. E., and D. G. Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association* 88: 9–25. https://doi.org/10.2307/2290687.

Cameron, A. C., and P. K. Trivedi. 2022. *Microeconometrics Using Stata*. 2nd ed. College Station, TX: Stata Press.

Canette, I. 2011. Including covariates in crossed-effects models. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2010/12/22/including-covariates-in-crossed-effects-models/.

Diggle, P. J., P. J. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data.* 2nd ed. Oxford: Oxford University Press.

Flay, B. R., B. R. Brannon, C. A. Johnson, W. B. Hansen, A. L. Ulene, D. A. Whitney-Saltiel, L. R. Gleason, S. Sussman, M. D. Gavin, K. M. Glowacz, D. F. Sobol, and D. C. Spiegel. 1988. The television, school, and family smoking cessation and prevention project: I. Theoretical basis and program development. *Preventive Medicine* 17: 585–607. https://doi.org/10.1016/0091-7435(88)90053-9.

Harville, D. A. 1977. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association* 72: 320–338. https://doi.org/10.2307/2286796.

Hedeker, D., and R. D. Gibbons. 2006. *Longitudinal Data Analysis.* Hoboken, NJ: Wiley.

Hocking, R. R. 1985. *The Analysis of Linear Models.* Monterey, CA: Brooks/Cole.

Horton, N. J. 2011. Stata tip 95: Estimation of error covariances in a linear model. *Stata Journal* 11: 145–148.

Huq, N. M., and J. Cleland. 1990. *Bangladesh Fertility Survey 1989 (Main Report).* National Institute of Population Research and Training.

Karim, M. R., and S. L. Zeger. 1992. Generalized linear models with random effects; salamander mating revisited. *Biometrics* 48: 631–644. https://doi.org/10.2307/2532317.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

LaMotte, L. R. 1973. Quadratic estimation of variance components. *Biometrics* 29: 311–330. https://doi.org/10.2307/2529395.

Lin, X., and N. E. Breslow. 1996. Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association* 91: 1007–1016. https://doi.org/10.2307/2291720.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models.* 2nd ed. Hoboken, NJ: Wiley.

Ng, E. S.-W., J. R. Carpenter, H. Goldstein, and J. Rasbash. 2006. Estimation in generalised linear mixed models with binary outcomes by simulated maximum likelihood. *Statistical Modelling* 6: 23–42. https://doi.org/10.1191/1471082X06st106oa.

Nichols, A. 2007. Causal inference with observational data. *Stata Journal* 7: 507–541.

Pantazis, N., and G. Touloumi. 2010. Analyzing longitudinal data in the presence of informative drop-out: The jmre1 command. *Stata Journal* 10: 226–251.

Pinheiro, J. C., and D. M. Bates. 2000. *Mixed-Effects Models in S and S-PLUS.* New York: Springer. https://doi.org/10.1007/b98882.

Pinheiro, J. C., and E. C. Chao. 2006. Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models. *Journal of Computational and Graphical Statistics* 15: 58–81. https://doi.org/10.1198/106186006X96962.

Rabe-Hesketh, S., and A. Skrondal. 2006. Multilevel modelling of complex survey data. *Journal of the Royal Statistical Society*, A ser., 169: 805–827. https://doi.org/10.1111/j.1467-985X.2006.00426.x.

———. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Raudenbush, S. W., and A. S. Bryk. 2002. *Hierarchical Linear Models: Applications and Data Analysis Methods*. 2nd ed. Thousand Oaks, CA: Sage.

Ruppert, D., M. P. Wand, and R. J. Carroll. 2003. *Semiparametric Regression*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511755453.

Schunck, R., and F. Perales. 2017. Within- and between-cluster effects in generalized linear mixed models: A discussion of approaches and the xthybrid command. *Stata Journal* 17: 89–115.

Searle, S. R. 1989. Obituary: Charles Roy Henderson 1911–1989. *Biometrics* 45: 1333–1335.

Searle, S. R., G. Casella, and C. E. McCulloch. 1992. *Variance Components*. New York: Wiley. https://doi.org/10.1002/9780470316856.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman and Hall/CRC.

Verbeke, G., and G. Molenberghs. 2000. *Linear Mixed Models for Longitudinal Data*. New York: Springer. https://doi.org/10.1007/978-1-4419-0300-6.

Also see

[ME] **meglm postestimation** — Postestimation tools for meglm

[ME] **mixed** — Multilevel mixed-effects linear regression+

[ME] **menl** — Nonlinear mixed-effects regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: meglm** — Bayesian multilevel generalized linear model

[R] **glm** — Generalized linear models

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[U] 20 Estimation and postestimation commands

meglm postestimation — Postestimation tools for meglm

Postestimation commands predict margins Remarks and examples
Methods and formulas References Also see

Postestimation commands

The following postestimation command is of special interest after meglm:

Command	Description
estat group	summarize the composition of the nested groups
estat icc	estimate intraclass correlations
estat sd	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
contrast	contrasts and ANOVA-style joint tests of parameters
estat ic	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
estat summarize	summary statistics for the estimation sample
estat vce	variance–covariance matrix of the estimators (VCE)
estat (svy)	postestimation statistics for survey data
estimates	cataloging estimation results
etable	table of estimation results
* hausman	Hausman's specification test
lincom	point estimates, standard errors, testing, and inference for linear combinations of parameters
* lrtest	likelihood-ratio test
margins	marginal means, predictive margins, marginal effects, and average marginal effects
marginsplot	graph the results from margins (profile plots, interaction plots, etc.)
nlcom	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
predict	means, probabilities, densities, REs, residuals, etc.
predictnl	point estimates, standard errors, testing, and inference for generalized predictions
pwcompare	pairwise comparisons of parameters
test	Wald tests of simple and composite linear hypotheses
testnl	Wald tests of nonlinear hypotheses

*hausman and lrtest are not appropriate with svy estimation results.

predict

Description for predict

predict creates a new variable containing predictions such as mean responses; linear predictions; density and distribution functions; standard errors; and raw, Pearson, deviance, and Anscombe residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

```
predict [type] { stub* | newvars } [if] [in] [ , statistic options ]
```

Syntax for obtaining estimated random effects and their standard errors

```
predict [type] { stub* | newvars } [if] [in], reffects [re_options]
```

Syntax for obtaining ML scores

```
predict [type] { stub* | newvars } [if] [in], scores
```

statistic	Description
Main	
mu	mean response; the default
pr	synonym for mu for ordinal and binary response models
eta	fitted linear predictor
xb	linear predictor for the fixed portion of the model only
stdp	standard error of the fixed-portion linear prediction
density	predicted density function
distribution	predicted distribution function
residuals	raw residuals; available only with the Gaussian family
pearson	Pearson residuals
deviance	deviance residuals
anscombe	Anscombe residuals

These statistics are available both in and out of sample; type predict ... if e(sample) ... if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
`outcome(`*outcome*`)`	outcome category for predicted probabilities for ordinal models
Integration	
int_options	integration options

`pearson`, `deviance`, `anscombe` may not be combined with `marginal`.

For ordinal outcomes, you specify one or *k* new variables in *newvars* with `mu` and `pr`, where *k* is the number of outcomes. If you do not specify `outcome()`, these options assume `outcome(#1)`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`mu`, the default, calculates the expected value of the outcome.

`pr` calculates predicted probabilities and is a synonym for `mu`. This option is available only for ordinal and binary response models.

`eta` calculates the fitted linear prediction.

`xb` calculates the linear prediction $\mathbf{x}\boldsymbol{\beta}$ using the estimated fixed effects (coefficients) in the model. This is equivalent to fixing all random effects in the model to their theoretical (prior) mean value of 0.

`stdp` calculates the standard error of the fixed-effects linear predictor $\mathbf{x}\beta$.

`density` calculates the density function. This prediction is computed using the current values of the observed variables, including the dependent variable.

`distribution` calculates the distribution function. This prediction is computed using the current values of the observed variables, including the dependent variable.

`residuals` calculates raw residuals, that is, responses minus the fitted values. This option is available only for the Gaussian family.

`pearson` calculates Pearson residuals. Pearson residuals that are large in absolute value may indicate a lack of fit.

`deviance` calculates deviance residuals. Deviance residuals are recommended by McCullagh and Nelder (1989) as having the best properties for examining the goodness of fit of a GLM. They are approximately normally distributed if the model is correctly specified. They can be plotted against the fitted values or against a covariate to inspect the model fit.

`anscombe` calculates Anscombe residuals, which are designed to closely follow a normal distribution.

`conditional(`*ctype*`)` and `marginal` specify how random effects are handled in computing *statistic*.

`conditional()` specifies that *statistic* will be computed conditional on specified or estimated random effects.

`conditional(ebmeans)`, the default, specifies that empirical Bayes means be used as the estimates of the random effects. These estimates are also known as posterior mean estimates of the random effects.

`conditional(ebmodes)` specifies that empirical Bayes modes be used as the estimates of the random effects. These estimates are also known as posterior mode estimates of the random effects.

`conditional(fixedonly)` specifies that all random effects be set to zero, equivalent to using only the fixed portion of the model.

`marginal` specifies that the predicted *statistic* be computed marginally with respect to the random effects, which means that *statistic* is calculated by integrating the prediction function with respect to all the random effects over their entire support.

Although this is not the default, marginal predictions are often very useful in applied analysis. They produce what are commonly called population-averaged estimates. They are also required by `margins`.

`nooffset` is relevant only if you specified `offset(`*varname*$_o$`)` or `exposure(`*varname*$_e$`)` with `meglm`. It modifies the calculations made by `predict` so that they ignore the offset or the exposure variable; the linear prediction is treated as $\mathbf{X}\beta + \mathbf{Zu}$ rather than $\mathbf{X}\beta + \mathbf{Zu} +$ offset, or $\mathbf{X}\beta + \mathbf{Zu} + \ln(\text{exposure})$, whichever is relevant.

`outcome(`*outcome*`)` specifies the outcome for which the predicted probabilities are to be calculated. `outcome()` should contain either one value of the dependent variable or one of `#1`, `#2`, ..., with `#1` meaning the first category of the dependent variable, `#2` meaning the second category, etc.

reffects calculates estimates of the random effects using empirical Bayes predictions. By default, or if the ebmeans option is specified, empirical Bayes means are computed. With the ebmodes option, empirical Bayes modes are computed. You must specify q new variables, where q is the number of random-effects terms in the model. However, it is much easier to just specify *stub*∗ and let Stata name the variables *stub*1, *stub*2, ..., *stubq* for you.

ebmeans specifies that empirical Bayes means be used to predict the random effects.

ebmodes specifies that empirical Bayes modes be used to predict the random effects.

reses(*stub*∗ | *newvars*) calculates standard errors of the empirical Bayes estimators and stores the result in *newvarlist*. This option requires the reffects option. You must specify q new variables, where q is the number of random-effects terms in the model. However, it is much easier to just specify *stub*∗ and let Stata name the variables *stub*1, *stub*2, ..., *stubq* for you. The new variables will have the same storage type as the corresponding random-effects variables.

The reffects and reses() options often generate multiple new variables at once. When this occurs, the random effects (and standard errors) contained in the generated variables correspond to the order in which the variance components are listed in the output of meglm. The generated variables are also labeled to identify their associated random effect.

scores calculates the scores for each coefficient in e(b). This option requires a new variable list of length equal to the number of columns in e(b). Otherwise, use the *stub*∗ syntax to have predict generate enumerated variables with prefix *stub*.

Integration

intpoints(#) specifies the number of quadrature points used to compute marginal predictions and the empirical Bayes means; the default is the value from estimation.

iterate(#) specifies the maximum number of iterations when computing statistics involving empirical Bayes estimators; the default is the value from estimation.

tolerance(#) specifies convergence tolerance when computing statistics involving empirical Bayes estimators; the default is the value from estimation.

margins

Description for margins

`margins` estimates margins of response for mean responses and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`mu`	mean response; the default
`pr`	synonym for `mu` for ordinal and binary response models
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`residuals`	not allowed with `margins`
`pearson`	not allowed with `margins`
`deviance`	not allowed with `margins`
`anscombe`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.
Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects model using `meglm`. For the most part, calculation centers around obtaining predictions of the random effects. Random effects are not estimated when the model is fit but instead need to be predicted after estimation.

▷ Example 1: Obtaining estimates of random effects

In example 2 of [ME] **meglm**, we modeled the probability of contraceptive use among Bangladeshi women by fitting a mixed-effects logistic regression model. To facilitate a more direct comparison between urban and rural women, we specify no base level for the `urban` factor variable and eliminate the constant from both the fixed-effects part and the random-effects part.

```
. use https://www.stata-press.com/data/r19/bangladesh
(Bangladesh Fertility Survey, 1989)
. meglm c_use ibn.urban age i.children, nocons nolog
> || district: ibn.urban, nocons family(bernoulli) link(logit) nofvlabel
Mixed-effects GLM                               Number of obs     =      1,934
Family: Bernoulli
Link:   Logit
Group variable: district                        Number of groups  =         60
                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =     120.59
Log likelihood = -1199.3268                     Prob > chi2       =     0.0000
 ( 1)  [c_use]_cons = 0
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
urban						
0	-1.712549	.1603689	-10.68	0.000	-2.026866	-1.398232
1	-.9004495	.1674683	-5.38	0.000	-1.228681	-.5722176
age	-.0264472	.0080196	-3.30	0.001	-.0421652	-.0107291
children						
1	1.132291	.1603052	7.06	0.000	.8180983	1.446483
2	1.358692	.1769369	7.68	0.000	1.011902	1.705482
3	1.354788	.1827459	7.41	0.000	.9966122	1.712963
_cons	0	(omitted)				
district						
var(0.urban)	.3882825	.1284858			.2029918	.7427064
var(1.urban)	.239777	.1403374			.0761401	.7550947

```
LR test vs. logistic model: chi2(2) = 58.40                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

This particular model allows for district random effects that are specific to the rural and urban areas of that district and that can be interpreted as such. We can obtain predictions of posterior means of the random effects and their standard errors by typing

```
. predict re_rural re_urban, reffects reses(se_rural se_urban)
(calculating posterior means of random effects)
(using 7 quadrature points)
```

The order in which we specified the variables to be generated corresponds to the order in which the variance components are listed in `meglm` output. If in doubt, a simple `describe` will show how these newly generated variables are labeled just to be sure.

Having generated estimated random effects and standard errors, we can now list them for the first 10 districts:

```
. by district, sort: generate tag = (_n==1)
. list district re_rural se_rural re_urban se_urban if district <= 10 & tag,
> sep(0)
```

	district	re_rural	se_rural	re_urban	se_urban
1.	1	-.9523374	.316291	-.5619418	.2329456
118.	2	-.0425217	.3819309	1.01e-17	.4896702
138.	3	4.29e-18	.6231232	.2229486	.4658747
140.	4	-.2703357	.3980832	.574464	.3962131
170.	5	.0691029	.3101591	.0074569	.4650451
209.	6	-.3939819	.2759802	.2622263	.4177785
274.	7	-.1904756	.4043461	-7.77e-18	.4896702
292.	8	.0382993	.3177392	.2250237	.4654329
329.	9	-.3715211	.3919996	.0628076	.453568
352.	10	-.5624707	.4763545	1.00e-18	.4896702

The estimated standard errors are conditional on the values of the estimated model parameters: $\boldsymbol{\beta}$ and the components of $\boldsymbol{\Sigma}$. Their interpretation is therefore not one of standard sample-to-sample variability but instead one that does not incorporate uncertainty in the estimated model parameters; see *Methods and formulas*. That stated, conditional standard errors can still be used as a measure of relative precision, provided that you keep this caveat in mind.

You can also obtain predictions of posterior modes and compare them with the posterior means:

```
. predict mod_rural mod_urban, reffects ebmodes
(calculating posterior modes of random effects)
. list district re_rural mod_rural re_urban mod_urban if district <= 10 & tag,
> sep(0)
```

	district	re_rural	mod_rural	re_urban	mod_urban
1.	1	-.9523374	-.9295366	-.5619418	-.5584528
118.	2	-.0425217	-.0306312	1.01e-17	0
138.	3	4.29e-18	0	.2229486	.2223551
140.	4	-.2703357	-.2529507	.574464	.5644512
170.	5	.0691029	.0789803	.0074569	.0077525
209.	6	-.3939819	-.3803784	.2622263	.2595116
274.	7	-.1904756	-.1737696	-7.77e-18	0
292.	8	.0382993	.0488528	.2250237	.2244676
329.	9	-.3715211	-.3540084	.0628076	.0605462
352.	10	-.5624707	-.535444	1.00e-18	0

The two sets of predictions are fairly close.

Because not all districts contain both urban and rural areas, some of the posterior modes are 0 and some of the posterior means are practically 0. A closer examination of the data reveals that district 3 has no rural areas, and districts 2, 7, and 10 have no urban areas.

Had we imposed an unstructured covariance structure in our model, the estimated posterior modes and posterior means in the cases in question would not be exactly 0 because of the correlation between urban and rural effects. For instance, if a district has no urban areas, it can still yield a nonzero (albeit small) random-effects estimate for a nonexistent urban area because of the correlation with its rural counterpart; see example 2 of [ME] **melogit postestimation** for details.

◁

▷ Example 2: Calculating predicted probabilities

Continuing with the model from example 1, we can obtain predicted probabilities, and unless we specify the `fixedonly` option, these predictions will incorporate the estimated subject-specific random effects $\tilde{\mathbf{u}}_j$.

```
. predict pr, pr
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

The predicted probabilities for observation i in cluster j are obtained by applying the inverse link function to the linear predictor, $\hat{p}_{ij} = g^{-1}(\mathbf{x}_{ij}\widehat{\boldsymbol{\beta}} + \mathbf{z}_{ij}\tilde{\mathbf{u}}_j)$; see *Methods and formulas* for details. By default or with the `conditional(ebmeans)` option, the calculation uses posterior means for $\tilde{\mathbf{u}}_j$. You can use the `conditional(ebmodes)` option to obtain predictions based on the posterior modes for $\tilde{\tilde{\mathbf{u}}}_j$.

```
. predict prm, pr conditional(ebmodes)
(predictions based on fixed effects and posterior modes of random effects)
```

We can list the two sets of predicted probabilities together with the actual outcome for some district, let's say district 38:

```
. list c_use pr prm if district == 38
```

	c_use	pr	prm
1228.	Yes	.5783408	.5780864
1229.	No	.5326623	.5324027
1230.	Yes	.6411679	.6409279
1231.	Yes	.5326623	.5324027
1232.	Yes	.5718783	.5716228
1233.	No	.3447686	.344533
1234.	No	.4507973	.4505391
1235.	No	.1940524	.1976133
1236.	No	.2846738	.2893007
1237.	No	.1264883	.1290078
1238.	No	.206763	.2104961
1239.	No	.202459	.2061346
1240.	No	.206763	.2104961
1241.	No	.1179788	.1203522

The two sets of predicted probabilities are fairly close.

For mixed-effects models with many levels or many random effects, the calculation of the posterior means of random effects or any quantities that are based on the posterior means of random effects may take a long time. This is because we must resort to numerical integration to obtain the posterior means. In contrast, the calculation of the posterior modes of random effects is usually orders of magnitude faster because there is no numerical integration involved. For this reason, empirical modes are often used in practice as an approximation to empirical means. Note that for linear mixed-effects models, the two predictors are the same.

We can compare the observed values with the predicted values by constructing a classification table. Defining success as $\widehat{y}_{ij} = 1$ if $\widehat{p}_{ij} > 0.5$ and defining $\widehat{y}_{ij} = 0$ otherwise, we obtain the following table.

```
. generate p_use = pr > .5
. label var p_use "Predicted outcome"
. tab2 c_use p_use, row
-> tabulation of c_use by p_use
```

Key
frequency *row percentage*

Use contraception	Predicted outcome 0	1	Total
No	991 84.34	184 15.66	1,175 100.00
Yes	423 55.73	336 44.27	759 100.00
Total	1,414 73.11	520 26.89	1,934 100.00

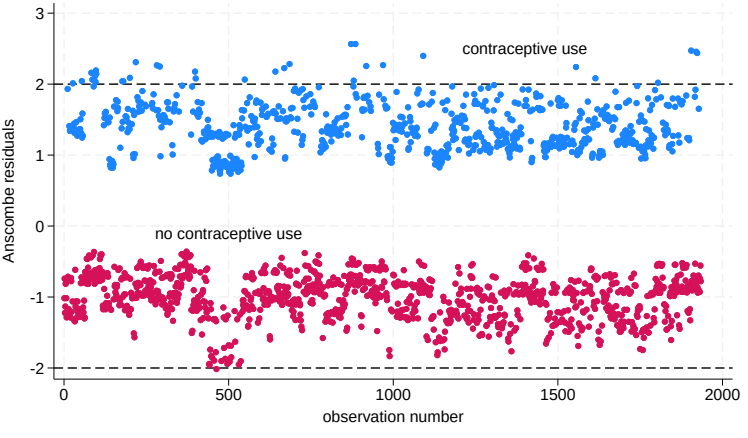
The model correctly classified 84% of women who did not use contraceptives but only 44% of women who did. In the next example, we will look at some residual diagnostics.

◁

Example 3: A look at residual diagnostics

Continuing our discussion from example 2, here we look at residual diagnostics. `meglm` offers three kinds of predicted residuals for nonlinear responses—Pearson, Anscombe, and deviance. Of the three, Anscombe residuals are designed to be approximately normally distributed; thus we can check for outliers by plotting Anscombe residuals against observation numbers and seeing which residuals are greater than 2 in absolute value.

```
. predict anscombe, anscombe
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
. generate n = _n
. label var n "observation number"
. twoway (scatter anscombe n if c_use) (scatter anscombe n if !c_use),
> yline(-2 2) legend(off) text(2.5 1400 "contraceptive use")
> text(-.1 500 "no contraceptive use")
```



There seem to be some outliers among residuals that identify women who use contraceptives. We could examine the observations corresponding to the outliers, or we could try fitting a model with perhaps a different covariance structure, which we leave as an exercise.

Example 4: Using predicted random effects for ranking purposes

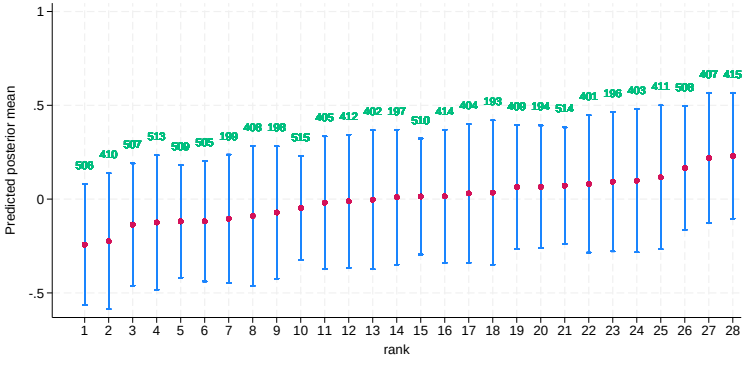
In example 3 of [ME] **meglm**, we estimated the effects of two treatments on the tobacco and health knowledge (THK) scale score of students in 28 schools. The dependent variable was collapsed into four ordered categories, and we fit a three-level ordinal logistic regression.

```
. use https://www.stata-press.com/data/r19/tvsfpors, clear
(Television, School, and Family Project)
. meologit thk prethk i.cc##i.tv || school: || class:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood =  -2125.509
Iteration 2:  Log likelihood = -2125.1034
Iteration 3:  Log likelihood = -2125.1032
Refining starting values:
Grid node 0:  Log likelihood = -2152.1514
Fitting full model:
  (output omitted)
Mixed-effects ologit regression                 Number of obs     =      1,600
        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
         school |         28         18       57.1        137
          class |        135          1       11.9         28
------------------------------------------------------------------------------
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(4)      =     124.39
Log likelihood = -2114.5881                     Prob > chi2       =     0.0000
------------------------------------------------------------------------------
         thk | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
      prethk |   .4085273   .039616    10.31   0.000     .3308814    .4861731
        1.cc |   .8844369   .2099124    4.21   0.000     .4730161    1.295858
        1.tv |    .236448   .2049065    1.15   0.249    -.1651614    .6380575
             |
       cc#tv |
        1 1 |  -.3717699   .2958887   -1.26   0.209     -.951701    .2081612
-------------+----------------------------------------------------------------
       /cut1 |  -.0959459   .1688988                    -.4269815    .2350896
       /cut2 |   1.177478   .1704946                     .8433151    1.511642
       /cut3 |   2.383672   .1786736                     2.033478    2.733865
-------------+----------------------------------------------------------------
school       |
   var(_cons) |   .0448735   .0425387                     .0069997    .2876749
-------------+----------------------------------------------------------------
school>class |
   var(_cons) |   .1482157   .0637521                      .063792    .3443674
------------------------------------------------------------------------------
LR test vs. ologit model: chi2(2) = 21.03                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Not surprisingly, the level of knowledge before the intervention is a good predictor of the level of knowledge after the intervention. The social resistance classroom curriculum is effective in raising the knowledge score, but the TV intervention and the interaction term are not.

We can rank schools by their institutional effectiveness by plotting the random effects at the school level.

```
. predict re_school re_class, reffects reses(se_school se_class)
(calculating posterior means of random effects)
(using 7 quadrature points)
. generate lower = re_school - 1.96*se_school
. generate upper = re_school + 1.96*se_school
. egen tag = tag(school)
. gsort +re_school -tag
. generate rank = sum(tag)
. generate labpos = re_school + 1.96*se_school + .1
. twoway (rcap lower upper rank) (scatter re_school rank)
> (scatter labpos rank, mlabel(school) msymbol(none) mlabpos(0)),
> xtitle(rank) ytitle(Predicted posterior mean) legend(off)
> xscale(range(0 28)) xlabel(1/28) ysize(2)
```



Although there is some variability in the predicted posterior means, we cannot see significant differences among the schools in this example.

◁

Methods and formulas

Continuing the discussion in *Methods and formulas* of [ME] **meglm** and using the definitions and formulas defined there, we begin by considering the prediction of the random effects $\mathbf{u}_j$ for the jth cluster in a two-level model. Prediction of random effects in multilevel generalized linear models involves assigning values to random effects, and there are many methods for doing so; see Skrondal and Rabe-Hesketh (2009) and Skrondal and Rabe-Hesketh (2004, chap. 7) for a comprehensive review. Stata offers two methods of predicting random effects: empirical Bayes means (also known as posterior means) and empirical Bayes modes (also known as posterior modes). Below we provide more details about the two methods.

Let $\widehat{\boldsymbol{\theta}}$ denote the estimated model parameters comprising $\widehat{\boldsymbol{\beta}}$ and the unique elements of $\widehat{\boldsymbol{\Sigma}}$. Empirical Bayes (EB) predictors of the random effects are the means or modes of the empirical posterior distribution with the parameter estimates $\boldsymbol{\theta}$ replaced with their estimates $\widehat{\boldsymbol{\theta}}$. The method is called "empirical" because $\widehat{\boldsymbol{\theta}}$ is treated as known. EB combines the prior information about the random effects with the likelihood to obtain the conditional posterior distribution of random effects. Using Bayes's theorem, the empirical conditional posterior distribution of random effects for cluster j is

$$\begin{aligned}\omega(\mathbf{u}_j|\mathbf{y}_j,\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}}) &= \frac{\Pr(\mathbf{y}_j,\mathbf{u}_j|\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}})}{\Pr(\mathbf{y}_j|\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}})}\\ &= \frac{f(\mathbf{y}_j|\mathbf{u}_j,\mathbf{X}_j,\mathbf{Z}_j;\widehat{\boldsymbol{\beta}})\,\phi(\mathbf{u}_j;\widehat{\boldsymbol{\Sigma}})}{\int f(\mathbf{y}_j|\mathbf{u}_j)\,\phi(\mathbf{u}_j)\,d\mathbf{u}_j}\\ &= \frac{f(\mathbf{y}_j|\mathbf{u}_j,\mathbf{X}_j,\mathbf{Z}_j;\widehat{\boldsymbol{\beta}})\,\phi(\mathbf{u}_j;\widehat{\boldsymbol{\Sigma}})}{\mathcal{L}_j(\hat{\boldsymbol{\theta}})}\end{aligned}$$

The denominator is just the likelihood contribution of the jth cluster.

EB mean predictions of random effects, $\tilde{\mathbf{u}}$, also known as posterior means, are calculated as

$$\begin{aligned}\tilde{\mathbf{u}} &= \int_{\Re^q} \mathbf{u}_j\,\omega(\mathbf{u}_j|\mathbf{y}_j,\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}})\,d\mathbf{u}_j\\ &= \frac{\int_{\Re^q} \mathbf{u}_j\,f(\mathbf{y}_j|\mathbf{u}_j,\mathbf{X}_j,\mathbf{Z}_j;\widehat{\boldsymbol{\beta}})\,\phi(\mathbf{u}_j;\widehat{\boldsymbol{\Sigma}})\,d\mathbf{u}_j}{\int_{\Re^q} f(\mathbf{y}_j|\mathbf{u}_j)\,\phi(\mathbf{u}_j)\,d\mathbf{u}_j}\end{aligned}$$

where we use the notation $\tilde{\mathbf{u}}$ rather than $\hat{\mathbf{u}}$ to distinguish predicted values from estimates. This multivariate integral is approximated by the mean–variance adaptive Gaussian quadrature; see *Methods and formulas* of [ME] **meglm** for details about the quadrature. If you have multiple random effects within a level or random effects across levels, the calculation involves orthogonalizing transformations using the Cholesky transformation because the random effects are no longer independent under the posterior distribution.

In a linear mixed-effects model, the posterior density is multivariate normal, and EB means are also best linear unbiased predictors (BLUPs); see Skrondal and Rabe-Hesketh (2004, 227). In generalized mixed-effects models, the posterior density tends to multivariate normal as cluster size increases.

EB modal predictions can be approximated by solving for the mode $\tilde{\tilde{\mathbf{u}}}_j$ in

$$\frac{\partial}{\partial \mathbf{u}_j}\log\omega(\tilde{\tilde{\mathbf{u}}}_j|\mathbf{y}_j,\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}}) = \mathbf{0}$$

Because the denominator in $\omega(\cdot)$ does not depend on $\mathbf{u}$, we can omit it from the calculation to obtain

$$\begin{aligned}&\frac{\partial}{\partial \mathbf{u}_j}\log\left\{f(\mathbf{y}_j|\mathbf{u}_j,\mathbf{X}_j,\mathbf{Z}_j;\widehat{\boldsymbol{\beta}})\,\phi(\mathbf{u}_j;\widehat{\boldsymbol{\Sigma}})\right\}\\ &= \frac{\partial}{\partial \mathbf{u}_j}\log f\left(\mathbf{y}_j|\mathbf{u}_j,\mathbf{X}_j,\mathbf{Z}_j;\widehat{\boldsymbol{\beta}}\right) + \frac{\partial}{\partial \mathbf{u}_j}\log\phi\left(\mathbf{u}_j;\widehat{\boldsymbol{\Sigma}}\right) = 0\end{aligned}$$

The calculation of EB modes does not require numerical integration, and for that reason they are often used in place of EB means. As the posterior density gets closer to being multivariate normal, EB modes get closer and closer to EB means.

Just like there are many methods of assigning values to the random effects, there exist many methods of calculating standard errors of the predicted random effects; see Skrondal and Rabe-Hesketh (2009) for a comprehensive review.

Stata uses the posterior standard deviation as the standard error of the posterior means predictor of random effects. The EB posterior covariance matrix of the random effects is given by

$$\text{cov}(\tilde{\mathbf{u}}_j|\mathbf{y}_j,\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}}) = \int_{\Re^q} (\mathbf{u}_j-\tilde{\mathbf{u}}_j)(\mathbf{u}_j-\tilde{\mathbf{u}}_j)'\,\omega(\mathbf{u}_j|\mathbf{y}_j,\mathbf{X}_j,\mathbf{Z}_j;\hat{\boldsymbol{\theta}})\,d\mathbf{u}_j$$

The posterior covariance matrix and the integrals are approximated by the mean–variance adaptive Gaussian quadrature; see *Methods and formulas* of [ME] **meglm** for details about the quadrature.

Conditional standard errors for the estimated posterior modes are derived from standard theory of maximum likelihood, which dictates that the asymptotic variance matrix of $\tilde{\tilde{\mathbf{u}}}_j$ is the negative inverse of the Hessian, $g''(\boldsymbol{\beta},\boldsymbol{\Sigma},\tilde{\tilde{\mathbf{u}}}_j)$.

In what follows, we show formulas using the posterior means estimates of random effects $\tilde{\mathbf{u}}_j$, which are used by default or if the `means` option is specified. If the `modes` option is specified, $\tilde{\mathbf{u}}_j$ are simply replaced with the posterior modes $\tilde{\tilde{\mathbf{u}}}_j$ in these formulas.

For any ith observation in the jth cluster in a two-level model, define the linear predictor as

$$\hat{\eta}_{ij} = \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}} + \mathbf{z}_{ij}\tilde{\mathbf{u}}_j$$

The linear predictor includes the offset or exposure variable if one was specified during estimation, unless the `nooffset` option is specified. If the `fixedonly` option is specified, $\hat{\eta}$ contains the linear predictor for only the fixed portion of the model, $\hat{\eta}_{ij} = \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}$.

The predicted mean, conditional on the random effects $\tilde{\mathbf{u}}_j$, is

$$\hat{\mu}_{ij} = g^{-1}(\hat{\eta}_{ij})$$

where $g^{-1}(\cdot)$ is the inverse link function for $\mu_{ij} = g^{-1}(\eta_{ij})$ defined as follows for the available links in `link(`*link*`)`:

link	Inverse link
`identity`	η_{ij}
`logit`	$1/\{1+\exp(-\eta_{ij})\}$
`probit`	$\Phi(\eta_{ij})$
`log`	$\exp(\eta_{ij})$
`cloglog`	$1-\exp\{-\exp(\eta_{ij})\}$

By default, random effects and any statistic based on them—`mu`, `fitted`, `pearson`, `deviance`, `anscombe`—are calculated using posterior means of random effects unless option `modes` is specified, in which case the calculations are based on posterior modes of random effects.

Raw residuals are calculated as the difference between the observed and fitted outcomes,

$$\nu_{ij} = y_{ij} - \hat{\mu}_{ij}$$

and are only defined for the Gaussian family.

Let r_{ij} be the number of Bernoulli trials in a binomial model, α be the conditional overdispersion parameter under the mean parameterization of the negative binomial model, and δ be the conditional overdispersion parameter under the constant parameterization of the negative binomial model.

Pearson residuals are raw residuals divided by the square root of the variance function

$$\nu_{ij}^{P} = \frac{\nu_{ij}}{\{V(\widehat{\mu}_{ij})\}^{1/2}}$$

where $V(\widehat{\mu}_{ij})$ is the family-specific variance function defined as follows for the available families in `family(`*family*`)`:

family	Variance function $V(\widehat{\mu}_{ij})$
`bernoulli`	$\widehat{\mu}_{ij}(1-\widehat{\mu}_{ij})$
`binomial`	$\widehat{\mu}_{ij}(1-\widehat{\mu}_{ij}/r_{ij})$
`gamma`	$\widehat{\mu}_{ij}^2$
`gaussian`	1
`nbinomial mean`	$\widehat{\mu}_{ij}(1+\alpha\widehat{\mu}_{ij})$
`nbinomial constant`	$\widehat{\mu}_{ij}(1+\delta)$
`ordinal`	not defined
`poisson`	$\widehat{\mu}_{ij}$

Deviance residuals are calculated as

$$\nu_{ij}^{D} = \text{sign}(\nu_{ij})\sqrt{\widehat{d}_{ij}^{2}}$$

where the squared deviance residual $\widehat{d}_{ij}^2$ is defined as follows:

family	Squared deviance $\widehat{d}_{ij}^2$
`bernoulli`	$-2\log(1-\widehat{\mu}_{ij})$ if $y_{ij}=0$
	$-2\log(\widehat{\mu}_{ij})$ if $y_{ij}=1$
`binomial`	$2r_{ij}\log\left(\dfrac{r_{ij}}{r_{ij}-\widehat{\mu}_{ij}}\right)$ if $y_{ij}=0$
	$2y_{ij}\log\left(\dfrac{y_{ij}}{\widehat{\mu}_{ij}}\right)+2(r_{ij}-y_{ij})\log\left(\dfrac{r_{ij}-y_{ij}}{r_{ij}-\widehat{\mu}_{ij}}\right)$ if $0<y_{ij}<r_{ij}$
	$2r_{ij}\log\left(\dfrac{r_{ij}}{\widehat{\mu}_{ij}}\right)$ if $y_{ij}=r_{ij}$
`gamma`	$-2\left\{\log\left(\dfrac{y_{ij}}{\widehat{\mu}_{ij}}\right)-\dfrac{\widehat{\nu}_{ij}}{\widehat{\mu}_{ij}}\right\}$
`gaussian`	$\widehat{\nu}_{ij}^2$
`nbinomial mean`	$2\log\left(1+\alpha\widehat{\mu}_{ij}\right)\alpha$ if $y_{ij}=0$
	$2y_{ij}\log\left(\dfrac{y_{ij}}{\widehat{\mu}_{ij}}\right)-\frac{2}{\alpha}(1+\alpha y_{ij})\log\left(\dfrac{1+\alpha y_{ij}}{1+\alpha\widehat{\mu}_{ij}}\right)$ otherwise
`nbinomial constant`	not defined
`ordinal`	not defined
`poisson`	$2\widehat{\mu}_{ij}$ if $y_{ij}=0$
	$2y_{ij}\log\left(\dfrac{y_{ij}}{\widehat{\mu}_{ij}}\right)-2\widehat{\nu}_{ij}$ otherwise

Anscombe residuals, denoted ν_{ij}^A, are calculated as follows:

family	Anscombe residual ν_{ij}^A
`bernoulli`	$\dfrac{3\left\{y_{ij}^{2/3}\mathcal{H}(y_{ij}) - \hat{\mu}_{ij}^{2/3}\mathcal{H}(\hat{\mu}_{ij})\right\}}{2\left(\hat{\mu}_{ij} - \hat{\mu}_{ij}^2\right)^{1/6}}$
`binomial`	$\dfrac{3\left\{y_{ij}^{2/3}\mathcal{H}(y_{ij}/r_{ij}) - \hat{\mu}_{ij}^{2/3}\mathcal{H}(\hat{\mu}_{ij}/r_{ij})\right\}}{2\left(\hat{\mu}_{ij} - \hat{\mu}_{ij}^2/r_{ij}\right)^{1/6}}$
`gamma`	$\dfrac{3(y_{ij}^{1/3} - \hat{\mu}_{ij}^{1/3})}{\hat{\mu}_{ij}^{1/3}}$
`gaussian`	ν_{ij}
`nbinomial mean`	$\dfrac{\mathcal{H}(-\alpha y_{ij}) - \mathcal{H}(-\alpha\hat{\mu}_{ij}) + 1.5(y_{ij}^{2/3} - \hat{\mu}_{ij}^{2/3})}{(\hat{\mu}_{ij} + \alpha\hat{\mu}_{ij}^2)^{1/6}}$
`nbinomial constant`	not defined
`ordinal`	not defined
`poisson`	$\dfrac{3(y_{ij}^{2/3} - \hat{\mu}_{ij}^{2/3})}{2\hat{\mu}_{ij}^{1/6}}$

where $\mathcal{H}(t)$ is a specific univariate case of the Hypergeometric2F1 function (Wolfram 2003, 780), defined here as $\mathcal{H}(t) = {}_2F_1(2/3, 1/3, 5/3, t)$.

For a discussion of the general properties of the various residuals, see Hardin and Hilbe (2018, chap. 4).

References

Hardin, J. W., and J. M. Hilbe. 2018. *Generalized Linear Models and Extensions*. 4th ed. College Station, TX: Stata Press.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman and Hall/CRC.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman and Hall/CRC.

———. 2009. Prediction in multilevel generalized linear models. *Journal of the Royal Statistical Society*, A ser., 172: 659–687. https://doi.org/10.1111/j.1467-985X.2009.00587.x.

Wolfram, S. 2003. *The Mathematica Book*. 5th ed. Champaign, IL: Wolfram Media.

Also see

[ME] **meglm** — Multilevel mixed-effects generalized linear models

[U] 20 Estimation and postestimation commands

meintreg — Multilevel mixed-effects interval regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`meintreg` fits mixed-effects models for continuous responses where the dependent variable may be measured as point data, interval-censored data, left-censored data, or right-censored data. Thus, it is a generalization of the models fit by `metobit`. The dependent variable must be specified using two variables that indicate how it was measured.

Quick start

Two-level interval regression on `x` with random intercepts by `lev2` of the interval-measured dependent variable with lower endpoint `y_lower` and upper endpoint `y_upper`

```
meintreg y_lower y_upper x || lev2:
```

Same as above, but with random coefficients for `x`

```
meintreg y_lower y_upper x || lev2: x
```

Three-level random-intercept model with `lev2` nested within `lev3`

```
meintreg y_lower y_upper x || lev3: || lev2:
```

Crossed-effects model with two-way crossed random effects by factors `a` and `b`

```
meintreg y_lower y_upper x || _all:R.a || b:
```

Menu

Statistics > Multilevel mixed-effects models > Interval regression

Syntax

```
meintreg depvar_lower depvar_upper fe_equation [ || re_equation ] [ || re_equation ... ]
    [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

The values in $depvar_{lower}$ and $depvar_{upper}$ should have the following form:

Type of data		$depvar_{lower}$	$depvar_{upper}$
point data	$a = [a, a]$	a	a
interval data	$[a, b]$	a	b
left-censored data	$(-\infty, b]$	.	b
right-censored data	$[a, +\infty)$	a	.
missing		.	.

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress constant term from the fixed-effects equation
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar$_{\text{lower}}$, *depvar*$_{\text{upper}}$, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

bayes, by, collect, and svy are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: meintreg**.

vce() and weights are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

startvalues(), startgrid, noestimate, dnumerical, collinear, and coeflegend do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

noconstant suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

offset(*varname*) specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

covariance(*vartype*) specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: independent, exchangeable, identity, unstructured, fixed(*matname*), or pattern(*matname*).

covariance(independent) covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is covariance(independent) unless a crossed random-effects model is fit, in which case the default is covariance(identity).

covariance(exchangeable) structure specifies one common variance for all random effects and one common pairwise covariance.

covariance(identity) is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

covariance(unstructured) allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

covariance(fixed(*matname*)) and covariance(pattern(*matname*)) covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a

`fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(#)`, `nocnsreport`; see [R] **Estimation options**.

`notable` suppresses the estimation table, either at estimation or upon replay.

`noheader` suppresses the output header, either at estimation or upon replay.

`nogroup` suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

Integration

`intmethod(`*intmethod*`)` specifies the integration method to be used for the random-effects model. `mvaghermite` performs mean–variance adaptive Gauss–Hermite quadrature; `mcaghermite` performs mode-curvature adaptive Gauss–Hermite quadrature; `ghermite` performs nonadaptive Gauss–Hermite quadrature; and `laplace` performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is `mvaghermite` unless a crossed random-effects model is fit, in which case the default integration method is `laplace`. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

`intpoints(`#`)` sets the number of integration points for quadrature. The default is `intpoints(7)`, which means that seven quadrature points are used for each level of random effects. This option is not allowed with `intmethod(laplace)`.

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(`#`)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(`#`)`, `ltolerance(`#`)`, `nrtolerance(`#`)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize**. Those that require special mention for `meintreg` are listed below.

`from()` accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with `meintreg` but are not shown in the dialog box:

`startvalues(`*svmethod*`)`, `startgrid[(`*gridspec*`)]`, `noestimate`, and `dnumerical`; see [ME] **meglm**.

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

Mixed-effects interval regression is regression for censored data containing both fixed effects and random effects. `meintreg` fits mixed-effects regression models that account for left-, right-, and interval-censoring. Thus, it is a generalization of the models fit by `metobit`. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

Interval data arise naturally in many contexts, such as wage data where often you know only that a person's salary is between two values. If one of the interval's endpoints is unknown, the observation is censored. Interval data and right-censored data also arise in the area of survival analysis. `meintreg` can

fit models for data where each observation represents interval data, left-censored data, right-censored data, or point data. Regardless of the type of observation, the data should be stored in the dataset as interval data; see *Syntax*.

Regardless of the type of censoring, the expected value of the underlying dependent variable—say, $\mathbf{y}^\star$—is modeled using the following linear prediction:

$$E(\mathbf{y}^\star|\mathbf{X},\mathbf{u}) = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} \tag{1}$$

$\mathbf{X}$ is an $n \times p$ design/covariate matrix, analogous to the covariates you would find in a standard linear regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. $\mathbf{Z}$ is the $n \times q$ design/covariate matrix for the random effects $\mathbf{u}$. This linear prediction also contains the offset when `offset()` is specified.

The columns of matrix $\mathbf{Z}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercepts model, $\mathbf{Z}$ is simply the scalar 1. The random effects $\mathbf{u}$ are realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{Z} = \mathbf{X}$ so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

Below we present a short example of mixed-effects censored regression; refer to [ME] **me** and [ME] **meglm** for additional examples of random-effects models. A two-level interval regression model can also be fit using `xtintreg`; see [XT] **xtintreg**. In the absence of random effects, mixed-effects censored regression reduces to standard censored regression; see [R] **intreg**.

Example 1: Three-level random-intercept model

Mastitis is a disease affecting dairy cows, consisting of an inflammatory reaction of the udder tissue. Our fictional study was performed on 10 farms using a sample of 10 dairy cows taken from each farm, and time to infection was recorded for each udder quarter for each cow in the sample. The four udder quarters are clustered within the cow, and cows are nested within farms. This is loosely based on nonfictional studies by Goethals et al. (2009) and Elghafghuf et al. (2014).

Cows were examined periodically. Thus, if a cow developed an infection, we do not know the exact day the infection occurred; we only know that it occurred between the last infection-free examination and the first examination where the infection was present. Some udder quarters did not develop an infection by the end of the study, so these observations are right-censored. We include a binary covariate, `multiparous`, which is equal to 1 for cows that have experienced more than one calving, and 0 for cows with only one calving.

To fit a log-normal model to the data, which assumes that the outcome is always positive, we take the log of our dependent variables and then use `meintreg` to apply a multilevel Gaussian model for interval- and right-censored data.

```
. use https://www.stata-press.com/data/r19/mastitis
(Simulated data on udder infection of dairy cows)
. generate lnleft = ln(left)
(5 missing values generated)
. generate lnright = ln(right)
(82 missing values generated)
```

```
. meintreg lnleft lnright i.multiparous || farm: || cow:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -912.93005
Iteration 1:  Log likelihood = -901.90184
Iteration 2:  Log likelihood = -901.48206
Iteration 3:  Log likelihood = -901.48176
Iteration 4:  Log likelihood = -901.48176
Refining starting values:
Grid node 0:  Log likelihood = -897.92167
Fitting full model:
Iteration 0:  Log likelihood = -897.92167  (not concave)
Iteration 1:  Log likelihood =  -863.2033  (not concave)
Iteration 2:  Log likelihood = -857.45304  (not concave)
Iteration 3:  Log likelihood = -855.18135
Iteration 4:  Log likelihood = -850.84325
Iteration 5:  Log likelihood = -846.31976
Iteration 6:  Log likelihood = -846.24446
Iteration 7:  Log likelihood = -846.24426
Iteration 8:  Log likelihood = -846.24426
Mixed-effects interval regression               Number of obs     =        400
                                                  Uncensored      =          0
                                                  Left-censored   =          5
                                                  Right-censored  =         82
                                                  Interval-cens.  =        313

        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
           farm |         10         40       40.0         40
            cow |        100          4        4.0          4
------------------------------------------------------------------------------

Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(1)      =       8.75
Log likelihood = -846.24426                     Prob > chi2       =     0.0031
------------------------------------------------------------------------------
             |      Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
1.multiparous|      -.5689113   .1923729    -2.96   0.003    -.9459552   -.1918674
       _cons |       5.644119   .1896383    29.76   0.000     5.272435    6.015803
-------------+----------------------------------------------------------------
farm         |
   var(_cons)|       .0246795   .0258621                      .0031648    .1924544
-------------+----------------------------------------------------------------
farm>cow     |
   var(_cons)|       .2481394   .0497735                      .1674773     .367651
-------------+----------------------------------------------------------------
 var(e.lnleft)|      .2626232   .0257671                      .2166796    .3183084
------------------------------------------------------------------------------
LR test vs. interval model: chi2(2) = 110.47              Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We see that infection was observed in 318 udder quarters, the 5 observations that are left-censored and the 313 that are interval censored. The coefficient for `multiparous` is negative, which means that the time to infection will be about 56.9% shorter for cows that experienced multiple calvings.

The estimates of the within-cow variance is 0.248, and the error variance is 0.263, while the within-farm variance is smaller, about 0.025. A likelihood-ratio test comparing the model to an interval regression model without random effects is provided under the table and indicates that the three-level interval regression model is preferred.

◁

Stored results

meintreg stores the following in e():

Scalars

e(N)	number of observations
e(N_unc)	number of uncensored observations
e(N_lc)	number of left-censored observations
e(N_rc)	number of right-censored observations
e(N_int)	number of interval-censored observations
e(k)	number of parameters
e(k_dv)	number of dependent variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(N_clust)	number of clusters
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	meintreg
e(cmdline)	command as typed
e(depvar)	names of dependent variables
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	interval
e(title)	title in estimation output
e(link)	identity
e(family)	gaussian
e(clustvar)	name of cluster variable
e(offset)	offset
e(intmethod)	integration method
e(n_quad)	number of integration points

e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Without a loss of generality, consider a two-level regression model

$$E(\mathbf{y}_j^\star|\mathbf{X}_j,\mathbf{u}_j) = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j, \qquad \mathbf{y}^\star \sim \text{normal}$$

for $j = 1,\ldots,M$ clusters, with the jth cluster consisting of n_j observations, where, for the jth cluster, $\mathbf{y}_j^\star$ is the $n_j \times 1$ response vector, $\mathbf{X}_j$ is the $n_j \times p$ matrix of fixed predictors, $\mathbf{Z}_j$ is the $n_j \times q$ matrix of random predictors, $\mathbf{u}_j$ is the $q \times 1$ vector of random effects, and $\boldsymbol{\beta}$ is the $p \times 1$ vector of regression coefficients on the fixed predictors. The random effects, $\mathbf{u}_j$, are assumed to be multivariate normal with mean $\mathbf{0}$ and variance $\boldsymbol{\Sigma}$.

Let $\boldsymbol{\eta}_j$ be the linear predictor, $\boldsymbol{\eta}_j = \mathbf{X}_j\boldsymbol{\beta}+\mathbf{Z}_j\mathbf{u}_j$, that also includes the offset variable when offset() is specified. Let $y_{ij}^\star$ and η_{ij} be the ith individual elements of $\mathbf{y}_j^\star$ and $\boldsymbol{\eta}_j$, $i = 1,\ldots,n_j$.

The dependent variable, y_{ij}, is a possibly left-, right-, or interval-censored version of $y_{ij}^\star$, and it is recorded using two variables.

The conditional density function for the response at observation ij is then,

$$f(y_{ij}^L, y_{ij}^U|\eta_{ij}) = \begin{cases} \left(\sqrt{2\pi}\sigma_\epsilon\right)^{-1} \exp^{-(y_{ij}^L-\eta_{ij})^2/(2\sigma_\epsilon^2)} & \text{if } (y_{ij}^L, y_{ij}^U) \in C \\ \Phi\left(\frac{y_{ij}^U-\eta_{ij}}{\sigma_\epsilon}\right) & \text{if } (y_{ij}^L, y_{ij}^U) \in L \\ 1-\Phi\left(\frac{y_{ij}^L-\eta_{ij}}{\sigma_\epsilon}\right) & \text{if } (y_{ij}^L, y_{ij}^U) \in R \\ \Phi\left(\frac{y_{ij}^U-\eta_{ij}}{\sigma_\epsilon}\right) - \Phi\left(\frac{y_{ij}^L-\eta_{ij}}{\sigma_\epsilon}\right) & \text{if } (y_{ij}^L, y_{ij}^U) \in I \end{cases}$$

where C is the set of uncensored observations ($y_{ij}^L = y_{ij}^U$ and both nonmissing), L is the set of left-censored observations (y_{ij}^L missing and y_{ij}^U nonmissing), R is the set of right-censored observations (y_{ij}^L nonmissing and y_{ij}^U missing), I is the set of interval-censored observations ($y_{ij}^L < y_{ij}^U$ and both nonmissing), and $\Phi(\cdot)$ is the cumulative normal distribution.

Because the observations are assumed to be conditionally independent, the conditional log density function for cluster j is

$$\log f(\mathbf{y}_j|\boldsymbol{\eta}_j) = \sum_{j=1}^{n_i} \log f(y_{ij}|\eta_{ij})$$

and the likelihood function for cluster j is given by

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2}|\boldsymbol{\Sigma}|^{-1/2} \int_{\Re^q} f(\mathbf{y}_j|\boldsymbol{\eta}_j) \exp\left(-\frac{1}{2}\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j\right) d\mathbf{u}_j \\ &= (2\pi)^{-q/2}|\boldsymbol{\Sigma}|^{-1/2} \int_{\Re^q} \exp\left\{\log f(\mathbf{y}_j|\boldsymbol{\eta}_j) - \frac{1}{2}\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j\right\} d\mathbf{u}_j \end{aligned} \tag{2}$$

where $\Re$ denotes the set of values on the real line and $\Re^q$ is the analog in q-dimensional space.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`meintreg` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Elghafghuf, A., S. Dufour, K. Reyher, I. Dohoo, and H. Stryhn. 2014. Survival analysis of clinical mastitis data using a nested frailty Cox model fit as a mixed-effects Poisson model. *Preventive Veterinary Medicine* 117: 456–468. https://doi.org/10.1016/j.prevetmed.2014.09.013.

Goethals, K., B. Ampe, D. Berkvens, H. Laevens, P. Janssen, and L. Duchateau. 2009. Modeling interval-censored, clustered cow udder quarter infection times through the shared gamma frailty model. *Journal of Agricultural, Biological, and Environmental Statistics* 14: 1–14. https://doi.org/10.1198/jabes.2009.0001.

Also see

[ME] **meintreg postestimation** — Postestimation tools for meintreg

[ME] **metobit** — Multilevel mixed-effects tobit regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: meintreg** — Bayesian multilevel interval regression

[R] **intreg** — Interval regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[ST] **stintreg** — Parametric models for interval-censored survival-time data

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtintreg** — Random-effects interval-data regression model

[U] 20 Estimation and postestimation commands

meintreg postestimation — Postestimation tools for meintreg

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation commands are of special interest after `meintreg`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

* `hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as linear predictions, standard errors, probabilities, and expected values.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`eta`	fitted linear predictor; the default
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`pr(`*a*`,`*b*`)`	$\Pr(a < y < b)$
`e(`*a*`,`*b*`)`	$E(y \mid a < y < b)$
`ystar(`*a*`,`*b*`)`	$E(y^*)$, $y^* = \max\{a, \min(y, b)\}$

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

where *a* and *b* may be numbers or variables; *a* missing ($a \geq .$) means $-\infty$, and *b* missing ($b \geq .$) means $+\infty$; see **[U] 12.2.1 Missing values**.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`eta`, the default, calculates the fitted linear prediction.

`pr(`*a*`,`*b*`)` calculates estimates of $\Pr(a < y < b)$, which is the probability that y would be observed in the interval (a, b).

a and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;
`pr(20,30)` calculates $\Pr(20 < y < 30)$;
`pr(`*lb*`,`*ub*`)` calculates $\Pr(lb < y < ub)$; and
`pr(20,`*ub*`)` calculates $\Pr(20 < y < ub)$.

a missing ($a \geq .$) means $-\infty$; `pr(.,30)` calculates $\Pr(-\infty < y < 30)$;
`pr(`*lb*`,30)` calculates $\Pr(-\infty < y < 30)$ in observations for which $lb \geq .$
(and calculates $\Pr(lb < y < 30)$ elsewhere).

b missing ($b \geq .$) means $+\infty$; `pr(20,.)` calculates $\Pr(+\infty > y > 20)$;
`pr(20,`*ub*`)` calculates $\Pr(+\infty > y > 20)$ in observations for which $ub \geq .$
(and calculates $\Pr(20 < y < ub)$ elsewhere).

`e(`*a*`,`*b*`)` calculates estimates of $E(y \mid a < y < b)$, which is the expected value of y conditional on y being in the interval (a, b), meaning that y is truncated. *a* and *b* are specified as they are for `pr()`.

`ystar(`*a*`,`*b*`)` calculates estimates of $E(y^*)$, where $y^* = a$ if $y \leq a$, $y^* = b$ if $y \geq b$, and $y^* = y$ otherwise, meaning that y^* is the censored version of y. *a* and *b* are specified as they are for `pr()`.

`xb`, `stdp`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for linear predictions, probabilities, and expected values.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`eta`	fitted linear predictor; the default
`xb`	linear predictor for the fixed portion of the model only
`pr(`*a*`,`*b*`)`	$\Pr(a < y < b)$
`e(`*a*`,`*b*`)`	$E(y \mid a < y < b)$
`ystar(`*a*`,`*b*`)`	$E(y^*)$, $y^* = \max\{a, \min(y, b)\}$
`stdp`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects interval regression model with `meintreg`.

The `predict` command allows us to compute marginal and conditional predictions. Unless stated differently, we use the word "conditional" to mean "conditional on the empirical Bayes predictions of the random effects". The default prediction is the linear prediction, `eta`, which is the expected value of the unobserved censored variable. Predictions of expected values for censored and truncated versions of the response are also available.

▷ Example 1: Obtaining conditional and marginal probabilities

In example 1 of [ME] **meintreg**, we fit a three-level mixed-effects interval regression to model log time to udder tissue infection in dairy cows.

```
. use https://www.stata-press.com/data/r19/mastitis
(Simulated data on udder infection of dairy cows)
. generate lnleft = ln(left)
(5 missing values generated)
. generate lnright = ln(right)
(82 missing values generated)
. meintreg lnleft lnright i.multiparous || farm: || cow:
```

(*output omitted*)

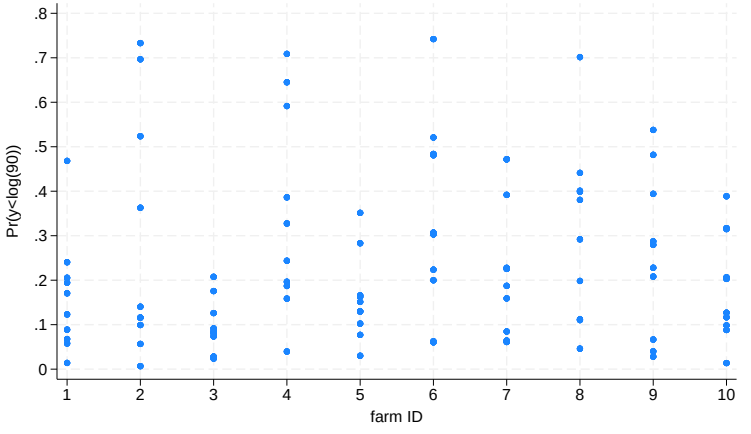
Let's assume that we want to predict the probability of infection within the first 90 days. Because our dependent variable is $\log(y)$, we need to compute

$$\Pr(0 < y < 90) = \Pr\{-\infty < \log(y) < \log(90)\}$$

We can use the `pr()` option for `predict` to compute the probability that our dependent variable lies in the interval $[-\infty, \log(90)]$.

We first compute the probability conditional on the random effects. Because the lower level on which we are conditioning on is cow, and we have only cow-level covariates, these predictions will be constant within cow. We can see that all the predicted probabilities for farm 3 are below 0.21, while the probabilities for farms 2 and 6 reach above 0.70 in some cases.

```
. predict pr_cond, pr(.,log(90))
(predictions based on fixed effects and posterior means of random effects)
. twoway scatter pr_cond farm, ylabel(0(.1).8) xlabel(1(1)10)
```



Now, we compute the marginal probabilities of infection within the first 90 days.

```
. predict pr_marg, pr(.,log(90)) marginal
. tabulate pr_marg multiparous
```

Marginal Pr(y<log(90))	=1 if cows experienced more than one calving, 0 otherwise 0	1	Total
.0589298	40	0	40
.2158333	0	360	360
Total	40	360	400

Marginal predictions depend only on the covariate pattern (including covariates in the random-effects part, if present in the model). Because we included only a binary covariate in the model, there are only two predicted values, one for each value of the covariate. We see that the probability of developing an infection in the first 90 days is higher for multiparous cows.

Alternatively, we can use `margins` to calculate the marginal probabilities. One advantage of using `margins` is that we can obtain confidence intervals for the probabilities and the difference between them.

```
. margins multiparous, predict(pr(.,log(90)))
Adjusted predictions                                    Number of obs = 400
Model VCE: OIM
Expression: Pr(y<log(90)), predict(pr(.,log(90)))
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
multiparous						
0	.0589298	.0305541	1.93	0.054	-.0009551	.1188147
1	.2158333	.0314158	6.87	0.000	.1542595	.2774071

```
. margins, dydx(multiparous) predict(pr(.,log(90)))
Conditional marginal effects                            Number of obs = 400
Model VCE: OIM
Expression: Pr(y<log(90)), predict(pr(.,log(90)))
dy/dx wrt:  1.multiparous
```

	dy/dx	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
1.multipar~s	.1569036	.0396889	3.95	0.000	.0791148	.2346923

```
Note: dy/dx for factor levels is the discrete change from the base level.
```

The default option for `predict`, `eta`, computes the fitted linear prediction; we can use this option to perform predictions for the uncensored unobserved response. We compute the conditional and marginal predictions for the log time to infection.

```
. predict eta_cond
(option eta assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
. predict eta_marg, marginal
(option eta assumed)
. sort cow
. list cow multiparous eta_cond eta_marg in 1/8, sepby(cow)
```

	cow	multip~s	eta_cond	eta_marg
1.	1	0	5.486386	5.644119
2.	1	0	5.486386	5.644119
3.	1	0	5.486386	5.644119
4.	1	0	5.486386	5.644119
5.	2	1	5.101668	5.075207
6.	2	1	5.101668	5.075207
7.	2	1	5.101668	5.075207
8.	2	1	5.101668	5.075207

Comparing the conditional and marginal predictions, we see that the predicted log time to infection for the first cow is slightly shorter than the one expected for a cow with this covariate pattern, and the log time to infection for the second cow is slightly longer.

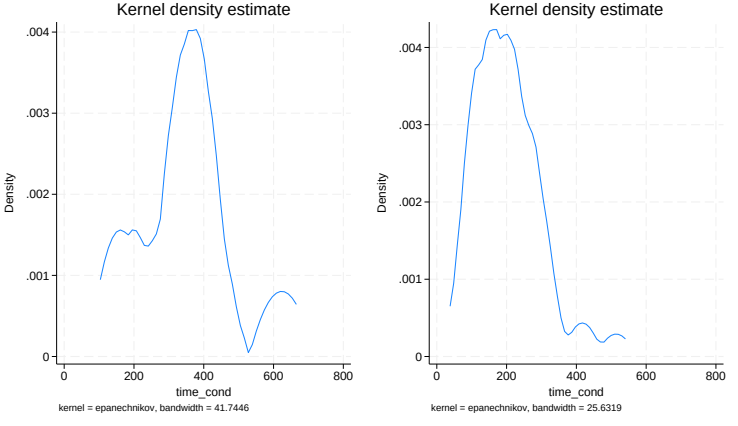
◁

▷ Example 2: Calculating transformed predictions

Because our dependent variable is log transformed, we might want to compute predictions on the original scale. To do that, we need to obtain predictions for the exponentiated dependent variable.

This exercise is helpful to understand the distribution of the different statistics. If we want to predict the individual conditional time to infection, we need to obtain the conditional mean for $\exp(y)$. Because the conditional distribution of $\exp(y)$ is lognormal with location parameter equal to $\hat{\eta}$ and scale parameter equal to σ_ϵ (error variance), then its (conditional) expected value is equal to $\exp(\hat{\eta} + \sigma_\epsilon^2/2)$. Here we calculate the conditional time to infection and plot kernel densities for multiparous and uniparous cows.

```
. generate time_cond = exp(eta_cond + _b[/var(e.lnleft)]/2)
. kdensity time_cond if multiparous == 0, xlabel(0(200)800) name(gr1)
. kdensity time_cond if multiparous == 1, xlabel(0(200)800) name(gr2)
. graph combine gr1 gr2
```



The density estimator of the time to infection shows that multiparous cows tend to have shorter times to infection than uniparous cows.

The marginal distribution of y is lognormal with location parameter $\mathbf{x}\boldsymbol{\beta}$ and the scale parameter equal to the marginal variance; see *Methods and formulas* of [ME] **metobit postestimation** for the description of the marginal variance. Thus the marginal expected value of the time to infection is calculated as

```
. predict xb, xb
. generate time_marg = exp( xb + (_b[/var(_cons[farm])] +
> _b[/var(_cons[farm>cow])] + _b[/var(e.lnleft)])/2)
. tabulate time_marg multiparous
```

time_marg	=1 if cows experienced more than one calving, 0 otherwise		Total
	0	1	
209.1242	0	360	360
369.3851	40	0	40
Total	40	360	400

As before, we see that the unconditional expected value for the time to infection is shorter for multiparous cows.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **metobit postestimation**.

Also see

[ME] **meintreg** — Multilevel mixed-effects interval regression

[U] 20 Estimation and postestimation commands

melogit — Multilevel mixed-effects logistic regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`melogit` fits mixed-effects models for binary and binomial responses. The conditional distribution of the response given the random effects is assumed to be Bernoulli, with success probability determined by the logistic cumulative distribution function.

Quick start

Without weights

Two-level logistic regression of `y` on `x` with random intercepts by `lev2`

```
melogit y x || lev2:
```

Mixed-effects model adding random coefficients for `x`

```
melogit y x || lev2: x
```

Same as above, but allow the random effects to be correlated

```
melogit y x || lev2: x, covariance(unstructured)
```

Three-level random-intercept model of `y` on `x` with `lev2` nested within `lev3`

```
melogit y x || lev3: || lev2:
```

Crossed-effects model of `y` on `x` with two-way crossed random effects by factors `a` and `b`

```
melogit y x || _all:R.a || b:
```

With weights

Two-level logistic regression of `y` on `x` with random intercepts by `lev2` and observation-level frequency weights `wvar1`

```
melogit y x [fweight=wvar1] || lev2:
```

Two-level random-intercept model from a two-stage sampling design with PSUs identified by `psu` using PSU-level and observation-level sampling weights `wvar2` and `wvar1`, respectively

```
melogit y x [pweight=wvar1] || psu:, pweight(wvar2)
```

Add secondary sampling stage with units identified by `ssu` having weights `wvar2` and PSU-level weights `wvar3` for a three-level random-intercept model

```
melogit y x [pw=wvar1] || psu:, pw(wvar3) || ssu:, pw(wvar2)
```

Same as above, but `svyset` data first

```
svyset psu, weight(wvar3) || ssu, weight(wvar2) || _n, weight(wvar1)
svy: melogit y x || psu: || ssu:
```

Menu

Statistics > Multilevel mixed-effects models > Logistic regression

Syntax

```
melogit depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress constant term from the fixed-effects equation
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1
`asis`	retain perfect predictor variables

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
binomial(*varname* | #)	set binomial trials if data are in binomial form
constraints(*constraints*)	apply specified linear constraints
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, opg, robust, or cluster *clustvar*
Reporting	
level(#)	set confidence level; default is level(95)
or	report fixed-effects coefficients as odds ratios
nocnsreport	do not display constraints
notable	suppress coefficient table
noheader	suppress output header
nogroup	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
intmethod(*intmethod*)	integration method
intpoints(#)	set the number of integration (quadrature) points for all levels; default is intpoints(7)
Maximization	
maximize_options	control the maximization process; seldom used
startvalues(*svmethod*)	method for obtaining starting values
startgrid[(*gridspec*)]	perform a grid search to improve starting values
noestimate	do not fit the model; show starting values instead
dnumerical	use numerical derivative techniques
collinear	keep collinear variables
coeflegend	display legend instead of statistics

vartype	Description
independent	one unique variance parameter per random effect and all covariances 0; the default unless the R. notation is used
exchangeable	equal variances for random effects and one common pairwise covariance
identity	equal variances for random effects and all covariances 0; the default if the R. notation is used
unstructured	all variances and covariances to be distinctly estimated
fixed(*matname*)	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
pattern(*matname*)	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
pcaghermite	Pinheiro–Chao mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models
pclaplace	Pinheiro–Chao Laplacian approximation

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: melogit**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

`offset(`*varname*`)` specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`asis` forces retention of perfect predictor variables and their associated, perfectly predicted observations and may produce instabilities in maximization; see [R] **probit**.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`binomial(`*varname* | #`)` specifies that the data are in binomial form; that is, *depvar* records the number of successes from a series of binomial trials. This number of trials is given either as *varname*, which allows this number to vary over the observations, or as the constant #. If `binomial()` is not specified (the default), *depvar* is treated as Bernoulli, with any nonzero, nonmissing values indicating positive responses.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(`#`)`; see [R] **Estimation options**.

or reports estimated fixed-effects coefficients transformed to odds ratios, that is, $\exp(\beta)$ rather than β. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. or may be specified either at estimation or upon replay.

nocnsreport; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite and pcaghermite perform mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace and pclaplace perform the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point. Techniques pcaghermite and pclaplace obtain the random-effects mode and curvature using the efficient hierarchical decomposition algorithm described in Pinheiro and Chao (2006). For hierarchical models, this algorithm takes advantage of the design structure to minimize memory use and utilizes a series of orthogonal triangulations to compute the factored random-effects Hessian indirectly, avoiding the sparse full Hessian. Techniques mcaghermite and laplace use Cholesky factorization on the full Hessian. For four- and higher-level hierarchical designs, there can be dramatic computation-time differences.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for melogit are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with melogit but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

For a general introduction to me commands, see [ME] **me**.

melogit is a convenience command for meglm with a logit link and a bernoulli or binomial family; see [ME] **meglm**.

Remarks are presented under the following headings:

Introduction
Two-level models
Other covariance structures
Three-level models
Crossed-effects models

Introduction

Mixed-effects logistic regression is logistic regression containing both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

melogit allows for many levels of random effects. However, for simplicity, for now we consider the two-level model, where for a series of M independent clusters, and conditional on a set of random effects $\mathbf{u}_j$,

$$\Pr(y_{ij} = 1|\mathbf{x}_{ij}, \mathbf{u}_j) = H(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j) \tag{1}$$

for $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The responses are the binary-valued y_{ij}, and we follow the standard Stata convention of treating $y_{ij} = 1$ if $\textit{depvar}_{ij} \neq 0$ and treating $y_{ij} = 0$ otherwise. The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard logistic regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. For notational convenience here and throughout this manual entry, we suppress the dependence of y_{ij} on $\mathbf{x}_{ij}$.

The $1 \times q$ vector $\mathbf{z}_{ij}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$ so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

Finally, because this is logistic regression, $H(\cdot)$ is the logistic cumulative distribution function, which maps the linear predictor to the probability of a success ($y_{ij} = 1$), with $H(v) = \exp(v)/\{1 + \exp(v)\}$.

Model (1) may also be stated in terms of a latent linear response, where only $y_{ij} = I(y^*_{ij} > 0)$ is observed for the latent

$$y^*_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij}$$

The errors ϵ_{ij} are distributed as logistic with mean 0 and variance $\pi^2/3$ and are independent of $\mathbf{u}_j$.

A two-level logistic model can also be fit using `xtlogit` with the `re` option; see [XT] **xtlogit**. In the absence of random effects, mixed-effects logistic regression reduces to standard logistic regression; see [R] **logit**.

Two-level models

▷ Example 1: Two-level random-intercept model

Ng et al. (2006) analyze a subsample of data from the 1989 Bangladesh fertility survey (Huq and Cleland 1990), which polled 1,934 Bangladeshi women on their use of contraception.

```
. use https://www.stata-press.com/data/r19/bangladesh
(Bangladesh Fertility Survey, 1989)
. describe
Contains data from https://www.stata-press.com/data/r19/bangladesh.dta
 Observations:         1,934                  Bangladesh Fertility Survey,
                                                1989
    Variables:             8                  28 May 2024 20:27
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
district        byte    %9.0g                 District
c_use           byte    %9.0g      yesno      Use contraception
urban           byte    %9.0g      urban      Urban or rural
age             float   %6.2f                 Age, mean centered
child1          byte    %9.0g                 1 child
child2          byte    %9.0g                 2 children
child3          byte    %9.0g                 3 or more children
children        byte    %18.0g     childlbl   Number of children
------------------------------------------------------------------------------
Sorted by: district
```

The women sampled were from 60 districts, identified by the variable `district`. Each district contained either urban or rural areas (variable `urban`) or both. The variable `c_use` is the binary response, with a value of 1 indicating contraceptive use. Other covariates include mean-centered `age` and a factor variable for the number of children.

Consider a standard logistic regression model, amended to have random effects for each district. Defining $\pi_{ij} = \Pr(\texttt{c_use}_{ij} = 1)$, we have

$$\begin{aligned}\text{logit}(\pi_{ij}) = \beta_0 + \beta_1 1.\texttt{urban}_{ij} + \beta_2 \texttt{age}_{ij} + \beta_3 1.\texttt{children}_{ij} + \beta_4 2.\texttt{children}_{ij} + \\ \beta_5 3.\texttt{children}_{ij} + u_j\end{aligned} \tag{2}$$

for $j = 1, \ldots, 60$ districts, with $i = 1, \ldots, n_j$ women in district j.

```
. melogit c_use i.urban age i.children, nofvlabel|| district:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -1229.5485
Iteration 1:  Log likelihood = -1228.5268
Iteration 2:  Log likelihood = -1228.5263
Iteration 3:  Log likelihood = -1228.5263
Refining starting values:
Grid node 0:  Log likelihood = -1219.2681
Fitting full model:
Iteration 0:  Log likelihood = -1219.2681  (not concave)
Iteration 1:  Log likelihood = -1207.5978
Iteration 2:  Log likelihood = -1206.8428
Iteration 3:  Log likelihood = -1206.8322
Iteration 4:  Log likelihood = -1206.8322
Mixed-effects logistic regression               Number of obs     =      1,934
Group variable: district                        Number of groups  =         60
                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(5)      =     109.60
Log likelihood = -1206.8322                     Prob > chi2       =     0.0000
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
1.urban	.7322765	.1194857	6.13	0.000	.4980888	.9664641
age	-.0264981	.0078916	-3.36	0.001	-.0419654	-.0110309
children						
1	1.116001	.1580921	7.06	0.000	.8061465	1.425856
2	1.365895	.1746691	7.82	0.000	1.02355	1.70824
3	1.344031	.1796549	7.48	0.000	.9919139	1.696148
_cons	-1.68929	.1477591	-11.43	0.000	-1.978892	-1.399687
district						
var(_cons)	.215618	.0733222			.1107208	.4198954

```
LR test vs. logistic model: chibar2(01) = 43.39       Prob >= chibar2 = 0.0000
```

The estimation table reports the fixed effects and the estimated variance components. The fixed effects can be interpreted just as you would the output from `logit`. You can also specify the `or` option at estimation or on replay to display the fixed effects as odds ratios instead. If you did display results as odds ratios, you would find urban women to have roughly double the odds of using contraception as that of their rural counterparts. Having any number of children will increase the odds from three- to fourfold when compared with the base category of no children. Contraceptive use also decreases with age. The `nofvlabel` option requested the values of factor variables `urban` and `children` be displayed instead of the value labels.

Underneath the fixed effect, the table shows the estimated variance components. The random-effects equation is labeled `district`, meaning that these are random effects at the `district` level. Because we have only one random effect at this level, the table shows only one variance component. The estimate of σ_u^2 is 0.22 with standard error 0.07.

A likelihood-ratio test comparing the model with ordinary logistic regression is provided and is highly significant for these data.

We now store our estimates for later use.

```
. estimates store r_int
```

◁

In what follows, we will be extending (2), focusing on the variable urban. Before we begin, to keep things short we restate (2) as

$$\text{logit}(\pi_{ij}) = \beta_0 + \beta_1 \texttt{1.urban}_{ij} + \mathcal{F}_{ij} + u_j$$

where $\mathcal{F}_{ij}$ is merely shorthand for the portion of the fixed-effects specification having to do with age and children.

▷ Example 2: Two-level random-slope model

Extending (2) to allow for a random slope on the indicator variable 1.urban yields the model

$$\text{logit}(\pi_{ij}) = \beta_0 + \beta_1 \texttt{1.urban}_{ij} + \mathcal{F}_{ij} + u_j + v_j \texttt{1.urban}_{ij} \tag{3}$$

which we can fit by typing

```
. melogit c_use i.urban age i.children, nofvlabel || district: i.urban
  (output omitted)
. estimates store r_urban
```

Extending the model was as simple as adding i.urban to the random-effects specification so that the model now includes a random intercept and a random coefficient on 1.urban. We dispense with the output because, although this is an improvement over the random-intercept model (2),

```
. lrtest r_int r_urban
Likelihood-ratio test
Assumption: r_int nested within r_urban
 LR chi2(1) =   3.66
Prob > chi2 = 0.0558
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

we find the default covariance structure for (u_j, v_j), covariance(independent),

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

to be inadequate. We state that the random-coefficient model is an "improvement" over the random-intercept model because the null hypothesis of the likelihood-ratio comparison test ($H_0 : \sigma_v^2 = 0$) is on the boundary of the parameter test. This makes the reported p-value, 5.6%, an upper bound on the actual p-value, which is actually half of that; see *Distribution theory for likelihood-ratio test* in [ME] **me**.

We see below that we can reject this model in favor of one that allows correlation between u_j and v_j.

```
. melogit c_use i.urban age i.children, nofvlabel
> || district: i.urban, covariance(unstructured)

Fitting fixed-effects model:

Iteration 0:  Log likelihood = -1229.5485
Iteration 1:  Log likelihood = -1228.5268
Iteration 2:  Log likelihood = -1228.5263
Iteration 3:  Log likelihood = -1228.5263

Refining starting values:

Grid node 0:  Log likelihood = -1215.8592

Fitting full model:

Iteration 0:  Log likelihood = -1215.8592  (not concave)
Iteration 1:  Log likelihood = -1201.0652
Iteration 2:  Log likelihood = -1199.6394
Iteration 3:  Log likelihood = -1199.3157
Iteration 4:  Log likelihood =  -1199.315
Iteration 5:  Log likelihood =  -1199.315

Mixed-effects logistic regression               Number of obs     =      1,934
Group variable: district                        Number of groups  =         60

                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118

Integration method: mvaghermite                 Integration pts.  =          7

                                                Wald chi2(5)      =      97.50
Log likelihood = -1199.315                      Prob > chi2       =     0.0000
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
1.urban	.8157875	.1715519	4.76	0.000	.4795519	1.152023
age	-.026415	.008023	-3.29	0.001	-.0421398	-.0106902
children						
1	1.13252	.1603285	7.06	0.000	.818282	1.446758
2	1.357739	.1770522	7.67	0.000	1.010723	1.704755
3	1.353827	.1828801	7.40	0.000	.9953882	1.712265
_cons	-1.71165	.1605618	-10.66	0.000	-2.026345	-1.396954
district						
var(1.urban)	.6663237	.3224689			.258074	1.720387
var(_cons)	.3897448	.1292463			.203473	.7465413
district						
cov(1.urban, _cons)	-.4058861	.1755414	-2.31	0.021	-.7499408	-.0618313

```
LR test vs. logistic model: chi2(3) = 58.42               Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

. estimates store r_urban_corr

. lrtest r_urban r_urban_corr

Likelihood-ratio test
Assumption: r_urban nested within r_urban_corr

 LR chi2(1) =  11.38
Prob > chi2 = 0.0007
```

By specifying `covariance(unstructured)` above, we told `melogit` to allow correlation between random effects at the `district` level; that is,

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} \sigma_u^2 & \sigma_{uv} \\ \sigma_{uv} & \sigma_v^2 \end{bmatrix}$$

◁

▷ Example 3: Alternative parameterization of random slopes

The purpose of introducing a random coefficient on the binary variable `urban` in (3) was to allow for separate random effects, within each district, for the urban and rural areas of that district. Hence, if we turn off base levels for factor variable `i.urban` via `ibn.urban`, then we can reformulate (3) as

$$\text{logit}(\pi_{ij}) = \beta_0 \texttt{0.urban}_{ij} + (\beta_0 + \beta_1)\texttt{1.urban}_{ij} + \mathcal{F}_{ij} + u_j \texttt{0.urban}_{ij} + (u_j + v_j)\texttt{1.urban}_{ij} \quad (3a)$$

where we have translated both the fixed portion and the random portion to be in terms of `0.urban` rather than a random intercept. Translating the fixed portion is not necessary to make the point we make below, but we do so anyway for uniformity.

Translating the estimated random-effects parameters from the previous output to ones appropriate for (3a), we get $\text{Var}(u_j) = \hat{\sigma}_u^2 = 0.39$,

$$\begin{aligned} \text{Var}(u_j + v_j) &= \hat{\sigma}_u^2 + \hat{\sigma}_v^2 + 2\hat{\sigma}_{uv} \\ &= 0.39 + 0.67 - 2(0.41) = 0.24 \end{aligned}$$

and $\text{Cov}(u_j, u_j + v_j) = \hat{\sigma}_u^2 + \hat{\sigma}_{uv} = 0.39 - 0.41 = -0.02$.

An alternative that does not require remembering how to calculate variances and covariances involving sums—and one that also gives you standard errors—is to let Stata do the work for you:

```
. melogit c_use ibn.urban age i.children, noconstant nofvlabel
> || district: ibn.urban, noconstant cov(unstructured)

Fitting fixed-effects model:

Iteration 0:  Log likelihood = -1229.5485
Iteration 1:  Log likelihood = -1228.5268
Iteration 2:  Log likelihood = -1228.5263
Iteration 3:  Log likelihood = -1228.5263

Refining starting values:

Grid node 0:  Log likelihood = -1208.3922

Fitting full model:

Iteration 0:  Log likelihood = -1208.3922  (not concave)
Iteration 1:  Log likelihood =  -1203.556  (not concave)
Iteration 2:  Log likelihood = -1200.5896
Iteration 3:  Log likelihood = -1199.7288
Iteration 4:  Log likelihood = -1199.3373
Iteration 5:  Log likelihood = -1199.3151
Iteration 6:  Log likelihood =  -1199.315

Mixed-effects logistic regression               Number of obs     =      1,934
Group variable: district                        Number of groups  =         60
                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =     120.24
Log likelihood = -1199.315                      Prob > chi2       =     0.0000
 ( 1)  [c_use]_cons = 0
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
urban						
0	-1.711652	.1605617	-10.66	0.000	-2.026347	-1.396956
1	-.8958623	.1704954	-5.25	0.000	-1.230027	-.5616974
age	-.026415	.008023	-3.29	0.001	-.0421398	-.0106903
children						
1	1.13252	.1603285	7.06	0.000	.8182819	1.446758
2	1.357739	.1770522	7.67	0.000	1.010724	1.704755
3	1.353827	.18288	7.40	0.000	.9953883	1.712265
_cons	0	(omitted)				
district						
var(0.urban)	.3897485	.1292403			.2034823	.7465212
var(1.urban)	.2442899	.1450625			.0762871	.7822759
district						
cov(0.urban, 1.urban)	-.0161411	.1057462	-0.15	0.879	-.2233999	.1911177

```
LR test vs. logistic model: chi2(3) = 58.42               Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

The above output demonstrates an equivalent fit to that we displayed for model (3), with the added benefit of a more direct comparison of the parameters for rural and urban areas.

◁

❑ Technical note

Our model fits for (3) and (3a) are equivalent only because we allowed for correlation in the random effects for both. Had we used the default independent covariance structure, we would be fitting different models; in (3) we would be making the restriction that $\text{Cov}(u_j, v_j) = 0$, whereas in (3a) we would be assuming that $\text{Cov}(u_j, u_j + v_j) = 0$.

The moral here is that although `melogit` will do this by default, one should be cautious when imposing an independent covariance structure, because the correlation between random effects is not invariant to model translations that would otherwise yield equivalent results in standard regression models. In our example, we remapped an intercept and binary coefficient to two complementary binary coefficients, something we could do in standard logistic regression without consequence but that here required more consideration.

Rabe-Hesketh and Skrondal (2022, sec. 11.4) provide a nice discussion of this phenomenon in the related case of recentering a continuous covariate.

❑

Other covariance structures

In the above examples, we demonstrated the `independent` and `unstructured` covariance structures. Also available are `identity` (seen previously in output but not directly specified), which restricts random effects to be uncorrelated and share a common variance, and `exchangeable`, which assumes a common variance and a common pairwise covariance.

You can also specify multiple random-effects equations at the same level, in which case the above four covariance types can be combined to form more complex blocked-diagonal covariance structures. This could be used, for example, to impose an equality constraint on a subset of variance components or to otherwise group together a set of related random effects.

Continuing the previous example, typing

```
. melogit c_use i.urban age i.children,
>         || district: i.children, cov(exchangeable)
>         || district:
```

would fit a model with the same fixed effects as (3) but with random-effects structure

$$\text{logit}(\pi_{ij}) = \beta_0 + \cdots + u_{1j}\texttt{1.children}_{ij} + u_{2j}\texttt{2.children}_{ij} + u_{3j}\texttt{3.children}_{ij} + v_j$$

That is, we have random coefficients on the children factor levels (the first `district:` specification) and an overall district random intercept (the second `district:` specification). The above syntax fits a model with overall covariance structure

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \\ v_j \end{bmatrix} = \begin{bmatrix} \sigma_u^2 & \sigma_c & \sigma_c & 0 \\ \sigma_c & \sigma_u^2 & \sigma_c & 0 \\ \sigma_c & \sigma_c & \sigma_u^2 & 0 \\ 0 & 0 & 0 & \sigma_v^2 \end{bmatrix}$$

reflecting the relationship among the random coefficients for children. We did not have to specify `noconstant` on the first `district:` specification. `melogit` automatically avoids collinearity by including an intercept on only the final specification among repeated-level equations.

Of course, if we fit the above model, we would heed our own advice from the previous technical note and make sure that not only our data but also our specification characterization of the random effects permitted the above structure. That is, we would check the above against a model that had an `unstructured` covariance for all four random effects and then perhaps against a model that assumed an `unstructured` covariance among the three random coefficients on children, coupled with independence with the random intercept. All comparisons can be made by storing estimates (command `estimates store`) and then using `lrtest`, as demonstrated previously.

Three-level models

▷ Example 4: Three-level random-intercept model

Rabe-Hesketh, Toulopoulou, and Murray (2001) analyzed data from a study measuring the cognitive ability of patients with schizophrenia compared with their relatives and control subjects. Cognitive ability was measured as the successful completion of the "Tower of London", a computerized task, measured at three levels of difficulty. For all but one of the 226 subjects, there were three measurements (one for each difficulty level). Because patients' relatives were also tested, a family identifier, `family`, was also recorded.

```
. use https://www.stata-press.com/data/r19/towerlondon, clear
(Tower of London data)
. describe
Contains data from https://www.stata-press.com/data/r19/towerlondon.dta
 Observations:           677                  Tower of London data
    Variables:             5                  31 May 2024 10:41
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
family          int     %8.0g                 Family ID
subject         int     %9.0g                 Subject ID
dtlm            byte    %9.0g                 1 = task completed
difficulty      byte    %9.0g                 Level of difficulty: -1, 0, or 1
group           byte    %8.0g                 1: controls; 2: relatives; 3:
                                                schizophrenics
------------------------------------------------------------------------------
Sorted by: family  subject
```

We fit a logistic model with response `dtlm`, the indicator of cognitive function, and with covariates `difficulty` and a set of indicator variables for `group`, with the controls (`group==1`) being the base category. We allow for random effects due to families and due to subjects within families, and we request to see odds ratios.

```
. melogit dtlm difficulty i.group || family: || subject: , or
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -317.35042
Iteration 1:  Log likelihood = -313.90007
Iteration 2:  Log likelihood = -313.89079
Iteration 3:  Log likelihood = -313.89079
Refining starting values:
Grid node 0:  Log likelihood = -310.28429
Fitting full model:
Iteration 0:  Log likelihood = -310.28429
Iteration 1:  Log likelihood = -307.36653
Iteration 2:  Log likelihood = -305.19357
Iteration 3:  Log likelihood = -305.12073
Iteration 4:  Log likelihood = -305.12041
Iteration 5:  Log likelihood = -305.12041
Mixed-effects logistic regression               Number of obs     =        677
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
family	118	2	5.7	27
subject	226	2	3.0	3

```
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(3)      =      74.90
Log likelihood = -305.12041                     Prob > chi2       =     0.0000
```

dtlm	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
difficulty	.1923372	.037161	-8.53	0.000	.1317057	.2808806
group						
2	.7798263	.2763763	-0.70	0.483	.3893369	1.561961
3	.3491318	.13965	-2.63	0.009	.15941	.764651
_cons	.226307	.0644625	-5.22	0.000	.1294902	.3955112
family						
var(_cons)	.5692105	.5215654			.0944757	3.429459
family>subject						
var(_cons)	1.137917	.6854853			.3494165	3.705762

```
Note: Estimates are transformed only in the first equation to odds ratios.
Note: _cons estimates baseline odds (conditional on zero random effects).
LR test vs. logistic model: chi2(2) = 17.54              Prob > chi2 = 0.0002
Note: LR test is conservative and provided only for reference.
```

This is a three-level model with two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `family` level, and the second is a random intercept at the `subject` level. The order in which these are specified (from left to right) is significant—`melogit` assumes that `subject` is nested within `family`.

The information on groups is now displayed as a table, with one row for each upper level. Among other things, we see that we have 226 subjects from 118 families.

After adjusting for the random-effects structure, the odds of successful completion of the Tower of London decrease dramatically as the level of difficulty increases. Also, schizophrenics (group==3) tended not to perform as well as the control subjects. Of course, we would make similar conclusions from a standard logistic model fit to the same data, but the odds ratios would differ somewhat.

◁

❑ Technical note

In the previous example, the subjects are coded with unique values between 1 and 251 (with some gaps), but such coding is not necessary to produce nesting within families. Once we specified the nesting structure to melogit, all that was important was the relative coding of subject within each unique value of family. We could have coded subjects as the numbers 1, 2, 3, and so on, restarting at 1 with each new family, and melogit would have produced the same results.

Group identifiers may also be coded using string variables.

❑

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by ||. The order of nesting goes from left to right as the groups go from biggest (highest level) to smallest (lowest level).

Crossed-effects models

▷ Example 5: Crossed-effects model

Rabe-Hesketh and Skrondal (2022, 493–512) perform an analysis on school data from Fife, Scotland. The data, originally from Paterson (1991), are from a study measuring students' attainment as an integer score from 1 to 10, based on the Scottish school exit examination taken at age 16. The study comprises 3,435 students who first attended any one of 148 primary schools and then any one of 19 secondary schools.

```
. use https://www.stata-press.com/data/r19/fifeschool
(School data from Fife, Scotland)
. describe
Contains data from https://www.stata-press.com/data/r19/fifeschool.dta
 Observations:         3,435                  School data from Fife, Scotland
    Variables:             5                  28 May 2024 10:08
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
pid             int     %9.0g                 Primary school ID
sid             byte    %9.0g                 Secondary school ID
attain          byte    %9.0g                 Attainment score at age 16
vrq             int     %9.0g                 Verbal-reasoning score from final
                                                year of primary school
sex             byte    %9.0g                 1: female; 0: male
------------------------------------------------------------------------------
Sorted by:
. generate byte attain_gt_6 = attain > 6
```

To make the analysis relevant to our present discussion, we focus not on the attainment score itself but instead on whether the score is greater than 6. We wish to model this indicator as a function of the fixed effect sex and of random effects due to primary and secondary schools.

For this analysis, it would make sense to assume that the random effects are not nested, but instead crossed, meaning that the effect due to primary school is the same regardless of the secondary school attended. Our model is thus

$$\text{logit}\{\Pr(\texttt{attain}_{ijk} > 6)\} = \beta_0 + \beta_1 \texttt{sex}_{ijk} + u_j + v_k \tag{4}$$

for student i, $i = 1, \ldots, n_{jk}$, who attended primary school j, $j = 1, \ldots, 148$, and then secondary school k, $k = 1, \ldots, 19$.

Because there is no evident nesting, one solution would be to consider the data as a whole and fit a two-level, one-cluster model with random-effects structure

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_{148} \\ v_1 \\ \vdots \\ v_{19} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}); \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_u^2 \mathbf{I}_{148} & \mathbf{0} \\ \mathbf{0} & \sigma_v^2 \mathbf{I}_{19} \end{bmatrix}$$

We can fit such a model by using the group designation `_all:`, which tells `melogit` to treat the whole dataset as one cluster, and the `R.`*varname* notation, which mimics the creation of indicator variables identifying schools:

```
. melogit attain_gt_6 sex || _all:R.pid || _all:R.sid, or
```

But we do not recommend fitting the model this way because of high total dimension ($148 + 19 = 167$) of the random effects. This would require working with matrices of column dimension 167, which is probably not a problem for most current hardware, but would be a problem if this number got much larger.

An equivalent way to fit (4) that has a smaller dimension is to treat the clusters identified by primary schools as nested within all the data, that is, as nested within the `_all` group.

```
. melogit attain_gt_6 sex || _all:R.sid || pid:, or
note: crossed random-effects model specified; option intmethod(laplace)
      implied.

Fitting fixed-effects model:

Iteration 0:  Log likelihood = -2320.2374
Iteration 1:  Log likelihood = -2317.9062
Iteration 2:  Log likelihood = -2317.9059
Iteration 3:  Log likelihood = -2317.9059

Refining starting values:

Grid node 0:  Log likelihood = -2234.6403

Fitting full model:

Iteration 0:  Log likelihood = -2234.6403  (not concave)
Iteration 1:  Log likelihood = -2227.9507  (not concave)
Iteration 2:  Log likelihood = -2227.9287  (not concave)
Iteration 3:  Log likelihood = -2227.9265  (not concave)
Iteration 4:  Log likelihood = -2227.9263
Iteration 5:  Log likelihood = -2221.6884  (not concave)
Iteration 6:  Log likelihood = -2221.1707  (not concave)
Iteration 7:  Log likelihood = -2221.1232
Iteration 8:  Log likelihood = -2220.1709  (not concave)
Iteration 9:  Log likelihood = -2220.1556
Iteration 10: Log likelihood = -2220.0522
Iteration 11: Log likelihood = -2220.0039
Iteration 12: Log likelihood = -2220.0035
Iteration 13: Log likelihood = -2220.0035

Mixed-effects logistic regression               Number of obs     =      3,435
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
_all	1	3,435	3,435.0	3,435
pid	148	1	23.2	72

```
Integration method: laplace

                                                Wald chi2(1)      =      14.33
Log likelihood = -2220.0035                     Prob > chi2       =     0.0002
```

attain_gt_6	Odds ratio	Std. err.	z	P>|z|	[95% conf.	interval]
sex	1.325119	.0985247	3.79	0.000	1.145425	1.533003
_cons	.5311554	.0621222	-5.41	0.000	.4223454	.6679984
_all>sid						
var(_cons)	.1239764	.0693322			.0414301	.37099
pid						
var(_cons)	.452049	.0951708			.2992129	.6829528

```
Note: Estimates are transformed only in the first equation to odds ratios.
Note: _cons estimates baseline odds (conditional on zero random effects).
LR test vs. logistic model: chi2(2) = 195.80              Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

Choosing the primary schools as those to nest was no accident; because there are far fewer secondary schools than primary schools, the above required only 19 random coefficients for the secondary schools and one random intercept at the primary school level, for a total dimension of 20. Our data also include a measurement of verbal reasoning, the variable `vrq`. Adding a fixed effect due to `vrq` in (4) would negate the effect due to secondary school, a fact we leave to you to verify as an exercise.

◁

See [ME] **mixed** for a similar discussion of crossed effects in the context of linear mixed models. Also see Rabe-Hesketh and Skrondal (2022) for more examples of crossed-effects models, including models with random interactions, and for more techniques on how to avoid high-dimensional estimation.

❑ Technical note

The estimation in the previous example was performed using a Laplacian approximation, even though we did not specify this. Whenever the `R.` notation is used in random-effects specifications, estimation reverts to the Laplacian method because of the high dimension induced by having the `R.` variables.

In the above example, through some creative nesting, we reduced the dimension of the random effects to 20, but this is still too large to permit estimation via adaptive Gaussian quadrature; see *Computation time and the Laplacian approximation* in [ME] **me**. Even with two quadrature points, our rough formula for computation time would contain within it a factor of $2^{20} = 1{,}048{,}576$.

The `intmethod(laplace)` option is therefore assumed when you use `R.` notation. If the number of distinct levels of your `R.` variables is small enough (say, five or fewer) to permit estimation via quadrature, you can override the imposition of `laplace` by specifying a different integration method in the `intmethod()` option.

❑

Stored results

`melogit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_dv)`	number of dependent variables
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_f)`	number of fixed-effects parameters
`e(k_r)`	number of random-effects parameters
`e(k_rs)`	number of variances
`e(k_rc)`	number of covariances
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(N_clust)`	number of clusters
`e(chi2)`	χ^2
`e(p)`	p-value for model test
`e(ll_c)`	log likelihood, comparison model
`e(chi2_c)`	χ^2, comparison test
`e(df_c)`	degrees of freedom, comparison test
`e(p_c)`	p-value for comparison test
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	melogit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	logistic
e(title)	title in estimation output
e(link)	logit
e(family)	bernoulli or binomial
e(clustvar)	name of cluster variable
e(offset)	offset
e(binomial)	binomial number of trials
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

`melogit` is a convenience command for `meglm` with a `logit` link and a `bernoulli` or `binomial` family; see [ME] **meglm**.

Model (1) assumes Bernoulli data, a special case of the binomial. Because binomial data are also supported by `melogit` (option `binomial()`), the methods presented below are in terms of the more general binomial mixed-effects model.

For a two-level binomial model, consider the response y_{ij} as the number of successes from a series of r_{ij} Bernoulli trials (replications). For cluster j, $j = 1, \ldots, M$, the conditional distribution of $\mathbf{y}_j = (y_{j1}, \ldots, y_{jn_j})'$, given a set of cluster-level random effects $\mathbf{u}_j$, is

$$\begin{aligned} f(\mathbf{y}_j|\mathbf{u}_j) &= \prod_{i=1}^{n_j} \left[\binom{r_{ij}}{y_{ij}} \left\{H(\boldsymbol{\eta}_{ij})\right\}^{y_{ij}} \left\{1 - H(\boldsymbol{\eta}_{ij})\right\}^{r_{ij}-y_{ij}} \right] \\ &= \exp\left(\sum_{i=1}^{n_j} \left[y_{ij}\boldsymbol{\eta}_{ij} - r_{ij} \log\left\{1 + \exp(\boldsymbol{\eta}_{ij})\right\} + \log\binom{r_{ij}}{y_{ij}} \right] \right) \end{aligned}$$

for $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \text{offset}_{ij}$ and $H(v) = \exp(v)/\{1 + \exp(v)\}$.

Defining $\mathbf{r}_j = (r_{j1}, \ldots, r_{jn_j})'$ and

$$c\left(\mathbf{y}_j, \mathbf{r}_j\right) = \sum_{i=1}^{n_j} \log\binom{r_{ij}}{y_{ij}}$$

where $c(\mathbf{y}_j, \mathbf{r}_j)$ does not depend on the model parameters, we can express the above compactly in matrix notation,

$$f(\mathbf{y}_j|\mathbf{u}_j) = \exp\left[\mathbf{y}_j'\boldsymbol{\eta}_j - \mathbf{r}_j' \log\left\{\mathbf{1} + \exp(\boldsymbol{\eta}_j)\right\} + c\left(\mathbf{y}_j, \mathbf{r}_j\right)\right]$$

where $\boldsymbol{\eta}_j$ is formed by stacking the row vectors $\boldsymbol{\eta}_{ij}$. We extend the definitions of the functions $\log(\cdot)$ and $\exp(\cdot)$ to be vector functions where necessary.

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j, \mathbf{u}_j)$,

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2} \left|\boldsymbol{\Sigma}\right|^{-1/2} \int f(\mathbf{y}_j|\mathbf{u}_j) \exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right) d\mathbf{u}_j \\ &= \exp\left\{c\left(\mathbf{y}_j, \mathbf{r}_j\right)\right\} (2\pi)^{-q/2} \left|\boldsymbol{\Sigma}\right|^{-1/2} \int \exp\left\{h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right)\right\} d\mathbf{u}_j \end{aligned} \tag{5}$$

where

$$h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right) = \mathbf{y}_j'\boldsymbol{\eta}_j - \mathbf{r}_j' \log\left\{\mathbf{1} + \exp(\boldsymbol{\eta}_j)\right\} - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j, \mathbf{r}_j, \mathbf{X}_j, \mathbf{Z}_j)$.

The integration in (5) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`melogit` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Harbord, R. M., and P. Whiting. 2009. metandi: Meta-analysis of diagnostic accuracy using hierarchical logistic regression. *Stata Journal* 9: 211–229.

Huq, N. M., and J. Cleland. 1990. *Bangladesh Fertility Survey 1989 (Main Report).* National Institute of Population Research and Training.

Joe, H. 2008. Accuracy of Laplace approximation for discrete response mixed models. *Computational Statistics and Data Analysis* 52: 5066–5074. https://doi.org/10.1016/j.csda.2008.05.002.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

Lin, X., and N. E. Breslow. 1996. Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association* 91: 1007–1016. https://doi.org/10.2307/2291720.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models.* 2nd ed. Hoboken, NJ: Wiley.

McLachlan, G. J., and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering.* New York: Dekker.

Ng, E. S.-W., J. R. Carpenter, H. Goldstein, and J. Rasbash. 2006. Estimation in generalised linear mixed models with binary outcomes by simulated maximum likelihood. *Statistical Modelling* 6: 23–42. https://doi.org/10.1191/1471082X06st106oa.

Paterson, L. 1991. Socio-economic status and educational attainment: A multidimensional and multilevel study. *Evaluation and Research in Education* 5: 97–121. https://doi.org/10.1080/09500799109533303.

Pinheiro, J. C., and E. C. Chao. 2006. Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models. *Journal of Computational and Graphical Statistics* 15: 58–81. https://doi.org/10.1198/106186006X96962.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Rabe-Hesketh, S., T. Toulopoulou, and R. M. Murray. 2001. Multilevel modeling of cognitive function in schizophrenic patients and their first degree relatives. *Multivariate Behavioral Research* 36: 279–298. https://doi.org/10.1207/S15327906MBR3602_07.

Self, S. G., and K.-Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association* 82: 605–610. https://doi.org/10.2307/2289471.

Also see

[ME] **melogit postestimation** — Postestimation tools for melogit

[ME] **mecloglog** — Multilevel mixed-effects complementary log–log regression

[ME] **meprobit** — Multilevel mixed-effects probit regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: melogit** — Bayesian multilevel logistic regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtlogit** — Fixed-effects, random-effects, and population-averaged logit models

[U] 20 Estimation and postestimation commands

melogit postestimation — Postestimation tools for melogit

Postestimation commands | predict | margins | Remarks and examples
Methods and formulas | References | Also see

Postestimation commands

The following postestimation commands are of special interest after `melogit`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as mean responses; linear predictions; density and distribution functions; standard errors; and Pearson, deviance, and Anscombe residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function
`pearson`	Pearson residuals
`deviance`	deviance residuals
`anscombe`	Anscombe residuals

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

`pearson`, `deviance`, `anscombe` may not be combined with `marginal`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`mu`, the default, calculates the predicted mean, that is, the probability of a positive outcome.

`eta`, `xb`, `stdp`, `density`, `distribution`, `pearson`, `deviance`, `anscombe`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, and `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for mean responses and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`pearson`	not allowed with `margins`
`deviance`	not allowed with `margins`
`anscombe`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.

Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a logistic mixed-effects model with melogit. For the most part, calculation centers around obtaining estimates of the subject/group-specific random effects. Random effects are not provided as estimates when the model is fit but instead need to be predicted after estimation. Calculation of intraclass correlations, estimating the dependence between latent linear responses for different levels of nesting, may also be of interest.

Example 1: Estimating the residual intraclass correlation

Following Rabe-Hesketh and Skrondal (2022, chap. 10), we consider a two-level mixed-effects model for onycholysis (separation of toenail plate from nail bed) among those who contract toenail fungus. The data are obtained from De Backer et al. (1998) and were also studied by Lesaffre and Spiessens (2001). The onycholysis outcome is dichotomously coded as 1 (moderate or severe onycholysis) or 0 (none or mild onycholysis). Fixed-effects covariates include treatment (0: itraconazole; 1: terbinafine), the month of measurement, and their interaction.

We fit the two-level model with melogit:

```
. use https://www.stata-press.com/data/r19/toenail
(Onycholysis severity)
. melogit outcome treatment month trt_month || patient:, intpoints(30)
  (iteration log omitted)
Mixed-effects logistic regression               Number of obs     =      1,908
Group variable: patient                         Number of groups  =        294
                                                Obs per group:
                                                              min =          1
                                                              avg =        6.5
                                                              max =          7
Integration method: mvaghermite                 Integration pts.  =         30
                                                Wald chi2(3)      =     150.61
Log likelihood = -625.38557                     Prob > chi2       =     0.0000
------------------------------------------------------------------------------
     outcome | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
   treatment |  -.1608934   .5802058    -0.28   0.782    -1.298076    .9762891
       month |  -.3911056   .0443906    -8.81   0.000    -.4781097   -.3041016
   trt_month |  -.1368286   .0680213    -2.01   0.044    -.2701479   -.0035093
       _cons |  -1.620355   .4322382    -3.75   0.000    -2.467526   -.7731834
-------------+----------------------------------------------------------------
patient      |
   var(_cons) |   16.0841   3.062625                      11.07431    23.36021
------------------------------------------------------------------------------
LR test vs. logistic model: chibar2(01) = 565.24      Prob >= chibar2 = 0.0000
```

It is of interest to determine the dependence among responses for the same subject (between-subject heterogeneity). Under the latent-linear-response formulation, this dependence can be obtained with the intraclass correlation. Formally, in a two-level random-effects model, the intraclass correlation corresponds to the correlation of latent responses within the same individual and also to the proportion of variance explained by the individual random effect.

We use `estat icc` to estimate the residual intraclass correlation:

```
. estat icc
Residual intraclass correlation
```

Level	ICC	Std. err.	[95% conf.	interval]
patient	.8301913	.0268433	.7709672	.8765531

In the presence of fixed-effects covariates, `estat icc` reports the residual intraclass correlation, which is the correlation between latent linear responses conditional on the fixed-effects covariates.

Conditional on treatment and month of treatment, we estimate that latent responses within the same patient have a large correlation of approximately 0.83. Further, 83% of the variance of a latent response is explained by the between-patient variability.

◁

▷ Example 2: Predicting random effects

In example 3 of [ME] **melogit**, we represented the probability of contraceptive use among Bangladeshi women by using the model (stated with slightly different notation here)

$$\begin{aligned}\text{logit}(\pi_{ij}) = \beta_0 \texttt{0.urban}_{ij} + \beta_1 \texttt{1.urban}_{ij} + \beta_2 \texttt{age}_{ij} + \\ \beta_3 \texttt{1.children}_{ij} + \beta_4 \texttt{2.children}_{ij} + \beta_5 \texttt{3.children}_{ij} + \\ a_j \texttt{0.urban}_{ij} + b_j \texttt{1.urban}_{ij}\end{aligned}$$

where π_{ij} is the probability of contraceptive use, $j = 1, \ldots, 60$ districts, $i = 1, \ldots, n_j$ women within each district, and a_j and b_j are normally distributed with mean 0 and variance–covariance matrix

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} a_j \\ b_j \end{bmatrix} = \begin{bmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ab} & \sigma_b^2 \end{bmatrix}$$

```
. use https://www.stata-press.com/data/r19/bangladesh
(Bangladesh Fertility Survey, 1989)
. melogit c_use ibn.urban age i.children, noconstant nofvlabel
> || district: ibn.urban, noconstant cov(unstructured)
Fitting fixed-effects model:
  (iteration log omitted)
Mixed-effects logistic regression               Number of obs     =      1,934
Group variable: district                        Number of groups  =         60
                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =     120.24
Log likelihood = -1199.315                      Prob > chi2       =     0.0000
 ( 1)  [c_use]_cons = 0
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
urban						
0	-1.711652	.1605617	-10.66	0.000	-2.026347	-1.396956
1	-.8958623	.1704954	-5.25	0.000	-1.230027	-.5616974
age	-.026415	.008023	-3.29	0.001	-.0421398	-.0106903
children						
1	1.13252	.1603285	7.06	0.000	.8182819	1.446758
2	1.357739	.1770522	7.67	0.000	1.010724	1.704755
3	1.353827	.18288	7.40	0.000	.9953883	1.712265
_cons	0	(omitted)				
district						
var(0.urban)	.3897485	.1292403			.2034823	.7465212
var(1.urban)	.2442899	.1450625			.0762871	.7822759
district						
cov(0.urban, 1.urban)	-.0161411	.1057462	-0.15	0.879	-.2233999	.1911177

```
LR test vs. logistic model: chi2(3) = 58.42                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

The purpose of using this particular model was to allow for district random effects that were specific to the rural and urban areas of that district and that could be interpreted as such. We can obtain predictions of these random effects and their corresponding standard errors,

```
. predict re_rural re_urban, reffects reses(se_rural se_urban)
(calculating posterior means of random effects)
(using 7 quadrature points)
```

The order in which we specified the variables to be generated corresponds to the order in which the variance components are listed in `melogit` output. If in doubt, a simple `describe` will show how these newly generated variables are labeled just to be sure.

Having generated estimated random effects and standard errors, we can now list them for the first 10 districts:

```
. by district, sort: generate tolist = (_n==1)
. list district re_rural se_rural re_urban se_urban if district <= 10 & tolist,
>  sep(0)
```

	district	re_rural	se_rural	re_urban	se_urban
1.	1	-.9432691	.3156852	-.558359	.2332665
118.	2	-.0427011	.3822029	.0017684	.493834
138.	3	-.0149571	.6242161	.2263715	.4698407
140.	4	-.2864846	.3990107	.5869046	.3988538
170.	5	.0688648	.3102899	.0046016	.4685461
209.	6	-.3979315	.2762392	.2761181	.4204175
274.	7	-.1910399	.4046772	.0079117	.4938647
292.	8	.034071	.3180097	.2266263	.4689558
329.	9	-.3737992	.3923573	.0764026	.4569863
352.	10	-.5640147	.4769353	.0233582	.4939753

◁

❑ Technical note

When these data were first introduced in [ME] **melogit**, we noted that not all districts contained both urban and rural areas. This fact is somewhat demonstrated by the random effects that are nearly 0 in the above. A closer examination of the data would reveal that district 3 has no rural areas, and districts 2, 7, and 10 have no urban areas.

The estimated random effects are not exactly 0 in these cases because of the correlation between urban and rural effects. For instance, if a district has no urban areas, it can still yield a nonzero (albeit small) random-effects estimate for a nonexistent urban area because of the correlation with its rural counterpart.

Had we imposed an independent covariance structure in our model, the estimated random effects in the cases in question would be exactly 0.

❑

❑ Technical note

The estimated standard errors produced above with the `reses()` option are conditional on the values of the estimated model parameters: $\boldsymbol{\beta}$ and the components of $\boldsymbol{\Sigma}$. Their interpretation is therefore not one of standard sample-to-sample variability but instead one that does not incorporate uncertainty in the estimated model parameters; see *Methods and formulas*.

That stated, conditional standard errors can still be used as a measure of relative precision, provided that you keep this caveat in mind.

❑

▷ Example 3: Obtaining predicted probabilities

Continuing with example 2, we can obtain predicted probabilities, the default prediction:

```
. predict p
(option mu assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

These predictions are based on a linear predictor that includes both the fixed effects and the random effects due to district. Specifying the `conditional(fixedonly)` option gives predictions that set the random effects to their prior mean of 0. Below we compare both over the first 20 observations:

```
. predict p_fixed, conditional(fixedonly)
(option mu assumed)
. list c_use p p_fixed age children in 1/20
```

	c_use	p	p_fixed	age	children
1.	No	.3572114	.4927182	18.44	3 or more children
2.	No	.21293	.3210403	-5.56	No children
3.	No	.4664207	.6044016	1.44	2 children
4.	No	.4198625	.5584863	8.44	3 or more children
5.	No	.2504834	.3687281	-13.56	No children
6.	No	.2406963	.3565185	-11.56	No children
7.	No	.3572114	.4927182	18.44	3 or more children
8.	No	.4984106	.6345998	-3.56	3 or more children
9.	No	.4564025	.594723	-5.56	1 child
10.	No	.465447	.6034657	1.44	3 or more children
11.	Yes	.2406963	.3565185	-11.56	No children
12.	No	.1999512	.3040173	-2.56	No children
13.	No	.4498569	.5883406	-4.56	1 child
14.	No	.439278	.5779263	5.44	3 or more children
15.	No	.4786124	.6160359	-0.56	3 or more children
16.	Yes	.4457945	.584356	4.44	3 or more children
17.	No	.21293	.3210403	-5.56	No children
18.	Yes	.4786124	.6160359	-0.56	3 or more children
19.	Yes	.4629632	.6010735	-6.56	1 child
20.	No	.4993888	.6355067	-3.56	2 children

◁

▷ Example 4: Estimating the conditional intraclass correlation

Continuing with example 2, we can also compute intraclass correlations for the model using `estat icc`; see [ME] **estat icc**.

In the presence of random-effects covariates, the intraclass correlation is no longer constant and depends on the values of the random-effects covariates. In this case, `estat icc` reports conditional intraclass correlations assuming 0 values for all random-effects covariates. For example, in a two-level model, this conditional correlation represents the correlation of the latent responses for two measurements on the same subject, both of which have random-effects covariates equal to 0. Similarly to the interpretation of intercept variances in random-coefficients models (Rabe-Hesketh and Skrondal 2022, chap. 16), interpretation of this conditional intraclass correlation relies on the usefulness of the 0 baseline values of random-effects covariates. For example, mean centering of the covariates is often used to make a 0 value a useful reference.

Estimation of the conditional intraclass correlation in the Bangladeshi contraceptive study setting of example 2 is of interest. In example 2, the random-effects covariates `0.urban` and `1.urban` for the random level `district` are mutually exclusive indicator variables and can never be simultaneously 0. Thus we could not use `estat icc` to estimate the conditional intraclass correlation for this model, because `estat icc` requires that the random intercept is included in all random-effects specifications.

Instead, we consider an alternative model for the Bangladeshi contraceptive study. In example 2 of [ME] **melogit**, we represented the probability of contraceptive use among Bangladeshi women with fixed-effects for urban residence (`urban`), age (`age`), and the number of children (`children`). The random effects for urban and rural residence are represented as a random slope for urban residence and a random intercept at the `district` level.

We fit the model with `melogit`:

```
. use https://www.stata-press.com/data/r19/bangladesh, clear
(Bangladesh Fertility Survey, 1989)
. melogit c_use i.urban age i.children, nofvlabel
> || district: i.urban, covariance(unstructured)
```

(*iteration log omitted*)

```
Mixed-effects logistic regression               Number of obs     =      1,934
Group variable: district                        Number of groups  =         60

                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118

Integration method: mvaghermite                 Integration pts.  =          7

                                                Wald chi2(5)      =      97.50
Log likelihood = -1199.315                      Prob > chi2       =     0.0000
```

c_use	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
1.urban	.8157875	.1715519	4.76	0.000	.4795519	1.152023
age	-.026415	.008023	-3.29	0.001	-.0421398	-.0106902
children						
1	1.13252	.1603285	7.06	0.000	.818282	1.446758
2	1.357739	.1770522	7.67	0.000	1.010723	1.704755
3	1.353827	.1828801	7.40	0.000	.9953882	1.712265
_cons	-1.71165	.1605618	-10.66	0.000	-2.026345	-1.396954
district						
var(1.urban)	.6663237	.3224689			.258074	1.720387
var(_cons)	.3897448	.1292463			.203473	.7465413
district						
cov(1.urban, _cons)	-.4058861	.1755414	-2.31	0.021	-.7499408	-.0618313

```
LR test vs. logistic model: chi2(3) = 58.42                 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

We use `estat icc` to estimate the intraclass correlation conditional on `urban` being equal to 0:

```
. estat icc
Conditional intraclass correlation
```

Level	ICC	Std. err.	[95% conf.	interval]
district	.1059201	.0314045	.058246	.1849518

```
Note: ICC is conditional on zero values of random-effects covariates.
```

This estimate suggests that the latent responses are not strongly correlated for rural residents (0.urban) within the same district, conditional on the fixed-effects covariates.

◁

▷ Example 5: Intraclass correlations for higher-level models

In example 4 of [ME] **melogit**, we fit a three-level model for the cognitive ability of schizophrenia patients as compared with their relatives and a control. Fixed-effects covariates include the difficulty of the test, difficulty, and an individual's category, group (control, family member of patient, or patient). Family units (family) represent the third nesting level, and individual subjects (subject) represent the second nesting level. Three measurements were taken on all but one subject, one for each difficulty measure.

We fit the model with melogit:

```
. use https://www.stata-press.com/data/r19/towerlondon
(Tower of London data)
. melogit dtlm difficulty i.group || family: || subject:
  (iteration log omitted)
Mixed-effects logistic regression               Number of obs     =        677
        Grouping information
 ------------------------------------------------------------------------------
  Group variable |     No. of       Observations per group
                 |     groups    Minimum    Average    Maximum
 ----------------+-------------------------------------------------------------
          family |        118          2        5.7         27
         subject |        226          2        3.0          3
 ------------------------------------------------------------------------------
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(3)      =      74.90
Log likelihood = -305.12041                     Prob > chi2       =     0.0000
------------------------------------------------------------------------------
        dtlm | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
  difficulty |  -1.648505   .1932075    -8.53   0.000    -2.027185   -1.269826
             |
       group |
          2 |  -.2486841   .3544076    -0.70   0.483    -.9433102     .445942
          3 |  -1.052306   .3999921    -2.63   0.009    -1.836276   -.2683357
             |
       _cons |  -1.485863   .2848455    -5.22   0.000     -2.04415   -.9275762
-------------+----------------------------------------------------------------
family       |
   var(_cons)|   .5692105   .5215654                      .0944757    3.429459
-------------+----------------------------------------------------------------
family>      |
subject      |
   var(_cons)|   1.137917   .6854853                      .3494165    3.705762
------------------------------------------------------------------------------
LR test vs. logistic model: chi2(2) = 17.54                 Prob > chi2 = 0.0002
Note: LR test is conservative and provided only for reference.
```

We can use `estat icc` to estimate the residual intraclass correlation (conditional on the difficulty level and the individual's category) between the latent responses of subjects within the same family or between the latent responses of the same subject and family:

```
. estat icc
Residual intraclass correlation
```

Level	ICC	Std. err.	[95% conf.	interval]
family	.1139105	.0997727	.0181851	.4715289
subject|family	.3416307	.0889471	.192923	.5297291

`estat icc` reports two intraclass correlations for this three-level nested model. The first is the level-3 intraclass correlation at the family level, the correlation between latent measurements of the cognitive ability in the same family. The second is the level-2 intraclass correlation at the subject-within-family level, the correlation between the latent measurements of cognitive ability in the same subject and family.

There is not a strong correlation between individual realizations of the latent response, even within the same subject.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

References

De Backer, M., C. De Vroey, E. Lesaffre, I. Scheys, and P. De Keyser. 1998. Twelve weeks of continuous oral therapy for toenail onychomycosis caused by dermatophytes: A double-blind comparative trial of terbinafine 250 mg/day versus itraconazole 200 mg/day. *Journal of the American Academy of Dermatology* 38: S57–S63. https://doi.org/10.1016/s0190-9622(98)70486-4.

Lesaffre, E., and B. Spiessens. 2001. On the effect of the number of quadrature points in a logistic random-effects model: An example. *Journal of the Royal Statistical Society*, C ser., 50: 325–335. https://doi.org/10.1111/1467-9876.00237.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Also see

[ME] **melogit** — Multilevel mixed-effects logistic regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

menbreg — Multilevel mixed-effects negative binomial regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`menbreg` fits mixed-effects negative binomial models to count data. The conditional distribution of the response given random effects is assumed to follow a Poisson-like process, except that the variation is greater than that of a true Poisson process.

Quick start

Mixed-effects negative binomial regression of `y` on `x` with random intercepts by `v1`

```
menbreg y x || v1:
```

Add `evar` measuring exposure

```
menbreg y x, exposure(evar) || v1:
```

Same as above, but report incidence-rate ratios instead of coefficients

```
menbreg y x, exposure(evar) || v1:, irr
```

Add random coefficients for `x`

```
menbreg y x, exposure(evar) || v1: x, irr
```

Three-level random-intercept model of `y` on `x` with `v1` nested within `v2`

```
menbreg y x || v2: || v1:
```

Menu

Statistics > Multilevel mixed-effects models > Negative binomial regression

Syntax

menbreg *depvar fe_equation* [|| *re_equation*] [|| *re_equation* ...] [, *options*]

where the syntax of *fe_equation* is

[*indepvars*] [*if*] [*in*] [*weight*] [, *fe_options*]

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

levelvar : [*varlist*] [, *re_options*]

for random effects among the values of a factor variable in a crossed-effects model

levelvar : R.*varname*

levelvar is a variable identifying the group structure for the random effects at that level or is _all representing one group comprising all observations.

fe_options	Description
Model	
noconstant	suppress the constant term from the fixed-effects equation
exposure(*varname*$_e$)	include ln(*varname*$_e$) in model with coefficient constrained to 1
offset(*varname*$_o$)	include *varname*$_o$ in model with coefficient constrained to 1

re_options	Description
Model	
covariance(*vartype*)	variance–covariance structure of the random effects
noconstant	suppress constant term from the random-effects equation
fweight(*varname*)	frequency weights at higher levels
iweight(*varname*)	importance weights at higher levels
pweight(*varname*)	sampling weights at higher levels

options	Description
Model	
`dispersion(`*dispersion*`)`	parameterization of the conditional overdispersion; *dispersion* may be `mean` (default) or `constant`
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`irr`	report fixed-effects coefficients as incidence-rate ratios
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: menbreg**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

`exposure(`*varname*$_e$`)` specifies a variable that reflects the amount of exposure over which the *depvar* events were observed for each observation; ln(*varname*$_e$) is included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`offset(`*varname*$_o$`)` specifies that *varname*$_o$ be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covari-

ances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`dispersion(mean | constant)` specifies the parameterization of the conditional overdispersion given random effects. `dispersion(mean)`, the default, yields a model where the conditional overdispersion is a function of the conditional mean given random effects. For example, in a two-level model, the conditional overdispersion is equal to $1+\alpha E(y_{ij}|\mathbf{u}_j)$. `dispersion(constant)` yields a model where the conditional overdispersion is constant and is equal to $1 + \delta$. α and δ are the respective conditional overdispersion parameters.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

level(#); see [R] **Estimation options**.

irr reports estimated fixed-effects coefficients transformed to incidence-rate ratios, that is, $\exp(\beta)$ rather than β. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. irr may be specified either at estimation or upon replay.

nocnsreport; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite performs mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for menbreg are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with menbreg but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

Mixed-effects negative binomial regression is negative binomial regression containing both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

menbreg allows for many levels of random effects. However, for simplicity, consider a two-level model, where for a series of M independent clusters, and conditional on the latent variable ζ_{ij} and a set of random effects $\mathbf{u}_j$,

$$y_{ij}|\zeta_{ij} \sim \text{Poisson}(\zeta_{ij})$$

and

$$\zeta_{ij}|\mathbf{u}_j \sim \text{Gamma}(r_{ij}, p_{ij})$$

and

$$\mathbf{u}_j \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

where y_{ij} is the count response of the ith observation, $i = 1, \ldots, n_j$, from the jth cluster, $j = 1, \ldots, M$, and r_{ij} and p_{ij} have two different parameterizations, (2) and (3) below. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components.

The probability that a random response y_{ij} takes the value y is then given by

$$\Pr(y_{ij} = y|\mathbf{u}_j) = \frac{\Gamma(y + r_{ij})}{\Gamma(y+1)\Gamma(r_{ij})} p_{ij}^{r_{ij}} (1 - p_{ij})^y \tag{1}$$

where for convenience we suppress the dependence of the observable data y_{ij} on r_{ij} and p_{ij}.

Model (1) is an extension of the standard negative binomial model (see [R] **nbreg**) to incorporate normally distributed random effects at different hierarchical levels. (The negative binomial model itself can be viewed as a random-effects model, a Poisson model with a gamma-distributed random effect.) The standard negative binomial model is used to model overdispersed count data for which the variance is greater than that of a Poisson model. In a Poisson model, the variance is equal to the mean, and thus overdispersion is defined as the extra variability compared with the mean. According to this definition, the negative binomial model presents two different parameterizations of the overdispersion: the mean parameterization, where the overdispersion is a function of the mean, $1 + \alpha E(Y|\mathbf{x}), \alpha > 0$; and the constant parameterization, where the overdispersion is a constant function, $1 + \delta, \delta \geq 0$. We refer to α and δ as conditional overdispersion parameters.

Let $\mu_{ij} = E(y_{ij}|\mathbf{x}, \mathbf{u_j}) = \exp(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j)$, where $\mathbf{x}_{ij}$ is the $1 \times p$ row vector of the fixed-effects covariates, analogous to the covariates you would find in a standard negative binomial regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$; $\mathbf{z}_{ij}$ is the $1 \times q$ vector of the random-effects covariates and

can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. One special case places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$, so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

Similarly to the standard negative binomial model, we can consider two parameterizations of what we call the conditional overdispersion, the overdispersion conditional on random effects, in a random-effects negative binomial model. For the mean-overdispersion (or, more technically, mean-conditional-overdispersion) parameterization,

$$r_{ij} = 1/\alpha \;\text{ and }\; p_{ij} = \frac{1}{1+\alpha\mu_{ij}} \tag{2}$$

and the conditional overdispersion is equal to $1 + \alpha\mu_{ij}$. For the constant-overdispersion (or, more technically, constant-conditional-overdispersion) parameterization,

$$r_{ij} = \mu_{ij}/\delta \;\text{ and }\; p_{ij} = \frac{1}{1+\delta} \tag{3}$$

and the conditional overdispersion is equal to $1 + \delta$. In what follows, for brevity, we will use the term overdispersion parameter to mean conditional overdispersion parameter, unless stated otherwise.

In the context of random-effects negative binomial models, it is important to decide which model is used as a reference model for the definition of the overdispersion. For example, if we consider a corresponding random-effects Poisson model as a comparison model, the parameters α and δ can still be viewed as unconditional overdispersion parameters, as we show below, although the notion of a constant overdispersion is no longer applicable.

If we retain the definition of the overdispersion as the excess variation with respect to a Poisson process for which the variance is equal to the mean, we need to carefully distinguish between the marginal (unconditional) mean with random effects integrated out and the conditional mean given random effects.

In what follows, for simplicity, we omit the dependence of the formulas on $\mathbf{x}$. Contingent on random effects, the (conditional) dispersion $\text{Var}(y_{ij}|\mathbf{u_j}) = (1 + \alpha\mu_{ij})\mu_{ij}$ for the mean parameterization and $\text{Var}(y_{ij}|\mathbf{u_j}) = (1 + \delta)\mu_{ij}$ for the constant parameterization; the usual interpretation of the parameters holds (conditionally).

If we consider the marginal mean or, specifically, the marginal dispersion for, for example, a two-level random-intercept model, then

$$\text{Var}(y_{ij}) = \left[1 + \{\exp(\sigma^2)(1+\alpha) - 1\}E(y_{ij})\right] E(y_{ij})$$

for the mean parameterization and

$$\text{Var}(y_{ij}) = \left[1 + \delta + \{\exp(\sigma^2) - 1\}E(y_{ij})\right] E(y_{ij})$$

for the constant parameterization, where σ^2 is the variance component corresponding to the random intercept.

A few things of interest compared with the standard negative binomial model. First, the random-effects negative binomial model is not strictly an overdispersed model. The combination of values of α and σ^2 can lead to an underdispersed model, a model with smaller variability than the Poisson variability. Underdispersed models are not as common in practice, so we will concentrate on the overdispersion in this entry. Second, α (or δ) no longer solely determine the overdispersion and thus cannot be viewed as unconditional overdispersion parameters. Overdispersion is now a function of both α (or δ) and σ^2. Third, the notion of a constant overdispersion is not applicable.

Two special cases are worth mentioning. When $\sigma^2 = 0$, the dispersion reduces to that of a standard negative binomial model. When $\alpha = 0$ (or $\delta = 0$), the dispersion reduces to that of a two-level random-intercept Poisson model, which itself is, in general, an overdispersed model; see Rabe-Hesketh and Skrondal (2022, sec. 13.7) for more details. As such, α and δ retain the typical interpretation as dispersion parameters relative to a random-intercept Poisson model.

Below we present two short examples of mixed-effects negative binomial regression; refer to [ME] **me** and [ME] **meglm** for more examples including crossed-effects models.

▷ Example 1: Two-level random-intercept model

Rabe-Hesketh and Skrondal (2022, sec. 13.7) analyze the data from Winkelmann (2004) on the impact of the 1997 health reform in Germany on the number of doctor visits. The intent of policymakers was to reduce government expenditures on healthcare. We use a subsample of the data restricted to 1,158 women who were employed full time the year before or after the reform.

```
. use https://www.stata-press.com/data/r19/drvisits
(Doctor visits)
. describe
Contains data from https://www.stata-press.com/data/r19/drvisits.dta
 Observations:         2,227                  Doctor visits
    Variables:             8                  23 Jan 2024 18:39
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
id                int     %9.0g                Person ID
numvisit          byte    %9.0g                Number of doctor visits in the
                                                 last 3 months before interview
age               byte    %9.0g                Age in years
educ              float   %9.0g                Education in years
married           byte    %9.0g                1 if married; 0 otherwise
badh              byte    %9.0g                Self-reported health status; 1 if
                                                 bad
loginc            float   %9.0g                Log of household income
reform            byte    %9.0g                0 if interview before reform; 1
                                                 if interview after reform
-------------------------------------------------------------------------------
Sorted by:
```

The dependent variable, `numvisit`, is a count of doctor visits. The covariate of interest is a dummy variable, `reform`, which indicates whether a doctor visit took place before or after the reform. Other covariates include a self-reported health status, age, education, marital status, and a log of household income.

We first fit a two-level random-intercept Poisson model. We specify the random intercept at the `id` level, that is, an individual-person level.

```
. mepoisson numvisit reform age educ married badh loginc || id:, irr
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -9326.8542
Iteration 1:  Log likelihood = -5989.7308
Iteration 2:  Log likelihood = -5942.7581
Iteration 3:  Log likelihood = -5942.7243
Iteration 4:  Log likelihood = -5942.7243
Refining starting values:
Grid node 0:  Log likelihood = -4761.1257
Fitting full model:
Iteration 0:  Log likelihood = -4761.1257
Iteration 1:  Log likelihood = -4683.2239
Iteration 2:  Log likelihood = -4646.9329
Iteration 3:  Log likelihood =  -4645.736
Iteration 4:  Log likelihood = -4645.7371
Iteration 5:  Log likelihood = -4645.7371
Mixed-effects Poisson regression                Number of obs     =      2,227
Group variable: id                              Number of groups  =      1,518
                                                Obs per group:
                                                              min =          1
                                                              avg =        1.5
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =     249.37
Log likelihood = -4645.7371                     Prob > chi2       =     0.0000
```

numvisit	IRR	Std. err.	z	P>|z|	[95% conf.	interval]
reform	.9517026	.0309352	-1.52	0.128	.8929617	1.014308
age	1.005821	.002817	2.07	0.038	1.000315	1.011357
educ	1.008788	.0127394	0.69	0.488	.9841258	1.034068
married	1.082078	.0596331	1.43	0.152	.9712905	1.205503
badh	2.471857	.151841	14.73	0.000	2.191471	2.788116
loginc	1.094144	.0743018	1.32	0.185	.9577909	1.249909
_cons	.5216748	.2668604	-1.27	0.203	.191413	1.421766
id						
var(_cons)	.8177932	.0503902			.724761	.9227673

```
Note: Estimates are transformed only in the first equation to incidence-rate
      ratios.
Note: _cons estimates baseline incidence rate (conditional on zero random
      effects).
LR test vs. Poisson model: chibar2(01) = 2593.97      Prob >= chibar2 = 0.0000
. estimates store mepoisson
```

Because we specified the `irr` option, the parameters are reported as incidence-rate ratios. The healthcare reform seems to reduce the expected number of visits by 5% but without statistical significance.

Because we have only one random effect at the `id` level, the table shows only one variance component. The estimate of σ_u^2 is 0.82 with standard error 0.05. The reported likelihood-ratio test shows that there is enough variability between women to favor a mixed-effects Poisson regression over a standard Poisson regression; see *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

It is possible that after conditioning on the person-level random effect, the counts of doctor visits are overdispersed. For example, medical problems occurring during the time period leading to the survey can result in extra doctor visits. We thus reexamine the data with `menbreg`.

```
. menbreg numvisit reform age educ married badh loginc || id:, irr
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -4610.7165
Iteration 1:  Log likelihood = -4563.4682
Iteration 2:  Log likelihood = -4562.3241
Iteration 3:  Log likelihood = -4562.3238
Refining starting values:
Grid node 0:  Log likelihood = -4643.5216
Fitting full model:
Iteration 0:  Log likelihood = -4643.5216  (not concave)
Iteration 1:  Log likelihood =  -4555.961
Iteration 2:  Log likelihood = -4518.7353
Iteration 3:  Log likelihood = -4513.1951
Iteration 4:  Log likelihood = -4513.1853
Iteration 5:  Log likelihood = -4513.1853
Mixed-effects nbinomial regression              Number of obs     =      2,227
Overdispersion: mean
Group variable: id                              Number of groups  =      1,518
                                                Obs per group:
                                                              min =          1
                                                              avg =        1.5
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =     237.35
Log likelihood = -4513.1853                     Prob > chi2       =     0.0000
```

numvisit	IRR	Std. err.	z	P>|z|	[95% conf.	interval]
reform	.9008536	.042022	-2.24	0.025	.8221449	.9870975
age	1.003593	.0028206	1.28	0.202	.9980799	1.009137
educ	1.007026	.012827	0.55	0.583	.9821969	1.032483
married	1.089597	.064213	1.46	0.145	.970738	1.223008
badh	3.043562	.2366182	14.32	0.000	2.613404	3.544523
loginc	1.136342	.0867148	1.67	0.094	.9784833	1.319668
_cons	.5017199	.285146	-1.21	0.225	.1646994	1.528377
/lnalpha	-.7962692	.1190614			-1.029625	-.5629132
id						
var(_cons)	.4740088	.0582404			.3725642	.6030754

```
Note: Estimates are transformed only in the first equation to incidence-rate
      ratios.
Note: _cons estimates baseline incidence rate (conditional on zero random
      effects).
LR test vs. nbinomial model: chibar2(01) = 98.28      Prob >= chibar2 = 0.0000
```

The estimated effect of the healthcare reform now corresponds to the reduction in the number of doctor visits by 10%—twice as much compared with the Poisson model—and this effect is significant at the 5% level.

The estimate of the variance component σ_u^2 drops down to 0.47 compared with `mepoisson`, which is not surprising given that now we have an additional parameter that controls the variability of the data.

Because the conditional overdispersion α is assumed to be greater than 0, it is parameterized on the log scale, and its log estimate is reported as `/lnalpha` in the output. In our model, $\widehat{\alpha} = \exp(-0.80) = 0.45$. We can also compute the unconditional overdispersion in this model by using $\exp(0.47)\times(1+0.45)-1 = 1.32$.

The reported likelihood-ratio test shows that there is enough variability between women to favor a mixed-effects negative binomial regression over negative binomial regression without random effects.

We can also perform a likelihood-ratio test comparing the mixed-effects negative binomial model to the mixed-effects Poisson model. Because we are comparing two different estimators, we need to use the `force` option with `lrtest`. In general, there is no guarantee as to the validity or interpretability of the resulting likelihood-ratio test, but in our case we know the test is valid because the mixed-effects Poisson model is nested within the mixed-effects negative binomial model.

```
. lrtest mepoisson ., force

Likelihood-ratio test
Assumption: mepoisson nested within .

 LR chi2(1) = 265.10
Prob > chi2 = 0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

The reported likelihood-ratio test favors the mixed-effects negative binomial model. The reported test is conservative because the test of $H_0 : \alpha = 0$ occurs on the boundary of the parameter space; see *Distribution theory for likelihood-ratio test* in [ME] **me** for details.

◁

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by `||`. The order of nesting goes from left to right as the groups go from biggest (highest level) to smallest (lowest level). To demonstrate a three-level model, we revisit example 3 from [ME] **mepoisson**.

▷ Example 2: Three-level random-intercept model

Rabe-Hesketh and Skrondal (2022, exercise 13.7) describe data from the *Atlas of Cancer Mortality in the European Economic Community* (EEC) (Smans, Mair, and Boyle 1992). The data were analyzed in Langford, Bentham, and McDonald (1998) and record the number of deaths among males due to malignant melanoma during 1971–1980.

```
. use https://www.stata-press.com/data/r19/melanoma
(Skin cancer (melanoma) data)
. describe
Contains data from https://www.stata-press.com/data/r19/melanoma.dta
 Observations:           354                  Skin cancer (melanoma) data
    Variables:             6                  30 May 2024 17:10
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
nation	byte	%11.0g	n	Nation ID
region	byte	%9.0g		Region ID: EEC level-I areas
county	int	%9.0g		County ID: EEC level-II/level-III areas
deaths	int	%9.0g		No. deaths during 1971-1980
expected	float	%9.0g		No. expected deaths
uv	float	%9.0g		UV dose, mean-centered

```
Sorted by:
```

Nine European nations (variable `nation`) are represented, and data were collected over geographical regions defined by EEC statistical services as level I areas (variable `region`), with deaths being recorded for each of 354 counties, which are level II or level III EEC-defined areas (variable `county`, which identifies the observations). Counties are nested within regions, and regions are nested within nations.

The variable `deaths` records the number of deaths for each county, and `expected` records the expected number of deaths (the exposure) on the basis of crude rates for the combined countries. The variable `uv` is a measure of exposure to ultraviolet (UV) radiation.

In example 3 of [ME] **mepoisson**, we noted that because counties also identified the observations, we could model overdispersion by using a four-level Poisson model with a random intercept at the county level. Here we fit a three-level negative binomial model with the default mean-dispersion parameterization.

```
. menbreg deaths uv, exposure(expected) || nation: || region:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -1361.855
Iteration 1:  Log likelihood = -1230.0211
Iteration 2:  Log likelihood =  -1211.049
Iteration 3:  Log likelihood = -1202.5641
Iteration 4:  Log likelihood = -1202.5329
Iteration 5:  Log likelihood = -1202.5329
Refining starting values:
Grid node 0:  Log likelihood = -1209.6951
Fitting full model:
  (output omitted)
Mixed-effects nbinomial regression              Number of obs     =        354
Overdispersion: mean
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
nation	9	3	39.3	95
region	78	1	4.5	13

```
Integration method: mvaghermite                 Integration pts.  =         7
                                                Wald chi2(1)      =      8.73
Log likelihood = -1086.3902                     Prob > chi2       =    0.0031
```

deaths	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
uv	-.0335933	.0113725	-2.95	0.003	-.055883	-.0113035
_cons	-.0790606	.1295931	-0.61	0.542	-.3330583	.1749372
ln(expected)	1	(exposure)				
/lnalpha	-4.182603	.3415036			-4.851937	-3.513268
nation						
var(_cons)	.1283614	.0678971			.0455187	.3619758
nation>region						
var(_cons)	.0401818	.0104855			.0240938	.067012

```
LR test vs. nbinomial model: chi2(2) = 232.29              Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

The estimates are very close to those of `mepoisson`. The conditional overdispersion in our model is $\hat{\alpha} = \exp(-4.18) = 0.0153$. It is in agreement with the estimate of the random intercept at the county level, 0.0147, in a four-level random-effects Poisson model reported by `mepoisson`. Because the negative binomial is a three-level model, we gained some computational efficiency over the four-level Poisson model.

◁

Stored results

`menbreg` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_dv)`	number of dependent variables
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_f)`	number of fixed-effects parameters
`e(k_r)`	number of random-effects parameters
`e(k_rs)`	number of variances
`e(k_rc)`	number of covariances
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(N_clust)`	number of clusters
`e(chi2)`	χ^2
`e(p)`	p-value for model test
`e(ll_c)`	log likelihood, comparison model
`e(chi2_c)`	χ^2, comparison test
`e(df_c)`	degrees of freedom, comparison test
`e(p_c)`	p-value for comparison test
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	menbreg
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	nbreg
e(title)	title in estimation output
e(link)	log
e(family)	nbinomial
e(clustvar)	name of cluster variable
e(dispersion)	mean or constant
e(offset)	offset
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

`menbreg` is a convenience command for `meglm` with a `log` link and an `nbinomial` family; see [ME] **meglm**.

Without a loss of generality, consider a two-level negative binomial model. For cluster j, $j = 1,\dots,M$, the conditional distribution of $\mathbf{y}_j = (y_{j1},\dots,y_{jn_j})'$, given a set of cluster-level random effects $\mathbf{u}_j$ and the conditional overdispersion parameter α in a mean-overdispersion parameterization, is

$$\begin{aligned} f(\mathbf{y}_j|\mathbf{u}_j,\alpha) &= \prod_{i=1}^{n_j}\left\{\frac{\Gamma(y_{ij}+r)}{\Gamma(y_{ij}+1)\Gamma(r)}p_{ij}^r(1-p_{ij})^{y_{ij}}\right\} \\ &= \exp\left[\sum_{i=1}^{n_j}\left\{\log\Gamma(y_{ij}+r) - \log\Gamma(y_{ij}+1) - \log\Gamma(r) + c(y_{ij},\alpha)\right\}\right] \end{aligned}$$

where $c(y_{ij},\alpha)$ is defined as

$$-\frac{1}{\alpha}\log\{1+\exp(\eta_{ij}+\log\alpha)\} - y_{ij}\log\{1+\exp(-\eta_{ij}-\log\alpha)\}$$

and $r = 1/\alpha$, $p_{ij} = 1/(1+\alpha\mu_{ij})$, and $\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j$.

For the constant-overdispersion parameterization with the conditional overdispersion parameter δ, the conditional distribution of $\mathbf{y}_j$ is

$$\begin{aligned} f(\mathbf{y}_j|\mathbf{u}_j,\delta) &= \prod_{i=1}^{n_j}\left\{\frac{\Gamma(y_{ij}+r_{ij})}{\Gamma(y_{ij}+1)\Gamma(r_{ij})}p^{r_{ij}}(1-p)^{y_{ij}}\right\} \\ &= \exp\left[\sum_{i=1}^{n_j}\left\{\log\Gamma(y_{ij}+r_{ij}) - \log\Gamma(y_{ij}+1) - \log\Gamma(r_{ij}) + c(y_{ij},\delta)\right\}\right] \end{aligned}$$

where $c(y_{ij},\delta)$ is defined as

$$-\left(\frac{\mu_{ij}}{\delta}+y_{ij}\right)\log(1+\delta) + y_{ij}\log\delta$$

and $r_{ij} = \mu_{ij}/\delta$ and $p = 1/(1+\delta)$.

For conciseness, let γ denote either conditional overdispersion parameter. Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j,\mathbf{u}_j,\gamma)$,

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta},\boldsymbol{\Sigma},\gamma) &= (2\pi)^{-q/2}\,|\boldsymbol{\Sigma}|^{-1/2}\int f(\mathbf{y}_j|\mathbf{u}_j,\gamma)\exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right)d\mathbf{u}_j \\ &= (2\pi)^{-q/2}\,|\boldsymbol{\Sigma}|^{-1/2}\int \exp\left\{h\left(\boldsymbol{\beta},\boldsymbol{\Sigma},\mathbf{u}_j,\gamma\right)\right\}d\mathbf{u}_j \end{aligned} \tag{4}$$

where

$$h\left(\boldsymbol{\beta},\boldsymbol{\Sigma},\mathbf{u}_j,\gamma\right) = f(\mathbf{y}_j|\mathbf{u}_j,\gamma) - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j, \mathbf{X}_j, \mathbf{Z}_j)$.

The integration in (4) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`menbreg` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Langford, I. H., G. Bentham, and A. McDonald. 1998. Multi-level modelling of geographically aggregated health data: A case study on malignant melanoma mortality and UV exposure in the European community. *Statistics in Medicine* 17: 41–57. https://doi.org/10.1002/(SICI)1097-0258(19980115)17:1<41::AID-SIM712>3.0.CO;2-0.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Smans, M., C. S. Mair, and P. Boyle, eds. 1992. *Atlas of Cancer Mortality in the European Economic Community*. Lyon: IARC.

Winkelmann, R. 2004. Health care reform and the number of doctor visits—an econometric analysis. *Journal of Applied Econometrics* 19: 455–472. https://doi.org/10.1002/jae.764.

Also see

[ME] **menbreg postestimation** — Postestimation tools for menbreg

[ME] **mepoisson** — Multilevel mixed-effects Poisson regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: menbreg** — Bayesian multilevel negative binomial regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtnbreg** — Fixed-effects, random-effects, & population-averaged negative binomial models

[U] 20 Estimation and postestimation commands

menbreg postestimation — Postestimation tools for menbreg

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation command is of special interest after `menbreg`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as mean responses; linear predictions; density and distribution functions; standard errors; and Pearson, deviance, and Anscombe residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function
`pearson`	Pearson residuals
`deviance`	deviance residuals
`anscombe`	Anscombe residuals

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

`pearson`, `deviance`, `anscombe` may not be combined with `marginal`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`mu`, the default, calculates the predicted mean, that is, the predicted number of events.

`eta`, `xb`, `stdp`, `density`, `distribution`, `pearson`, `deviance`, `anscombe`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, and `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for mean responses and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`pearson`	not allowed with `margins`
`deviance`	not allowed with `margins`
`anscombe`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.
Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects negative binomial model with `menbreg`. For the most part, calculation centers around obtaining estimates of the subject/group-specific random effects. Random effects are not estimated when the model is fit but instead need to be predicted after estimation.

Here we show a short example of predicted counts and predicted random effects; refer to [ME] **meglm postestimation** for additional examples applicable to mixed-effects generalized linear models.

▷ Example 1: Predicting counts and random effects

In example 2 of [ME] **menbreg**, we modeled the number of deaths among males in nine European nations as a function of exposure to ultraviolet radiation (`uv`). We used a three-level negative binomial model with random effects at the nation and region levels.

```
. use https://www.stata-press.com/data/r19/melanoma
(Skin cancer (melanoma) data)
. menbreg deaths uv, exposure(expected) || nation: || region:
  (output omitted)
```

We can use `predict` to obtain the predicted counts as well as the estimates of the random effects at the nation and region levels.

```
. predict mu
(option mu assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
. predict re_nat re_reg, reffects
(calculating posterior means of random effects)
(using 7 quadrature points)
```

Stata displays a note that the predicted values of `mu` are based on the posterior means of random effects. You can use option `modes` to obtain predictions based on the posterior modes of random effects.

Here we list the data for the first nation in the dataset, which happens to be Belgium:

```
. list nation region deaths mu re_nat re_reg if nation==1, sepby(region)
```

	nation	region	deaths	mu	re_nat	re_reg
1.	Belgium	1	79	64.4892	-.0819939	.2937711
2.	Belgium	2	80	77.64736	-.0819939	.024005
3.	Belgium	2	51	44.56528	-.0819939	.024005
4.	Belgium	2	43	53.10434	-.0819939	.024005
5.	Belgium	2	89	65.35963	-.0819939	.024005
6.	Belgium	2	19	35.18457	-.0819939	.024005
7.	Belgium	3	19	8.770186	-.0819939	-.3434432
8.	Belgium	3	15	43.95521	-.0819939	-.3434432
9.	Belgium	3	33	34.17878	-.0819939	-.3434432
10.	Belgium	3	9	7.332448	-.0819939	-.3434432
11.	Belgium	3	12	12.93873	-.0819939	-.3434432

We can see that the predicted random effects at the nation level, `re_nat`, are the same for all the observations. Similarly, the predicted random effects at the region level, `re_reg`, are the same within each region.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

Also see

[ME] **menbreg** — Multilevel mixed-effects negative binomial regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] **20 Estimation and postestimation commands**

Description Quick start Menu Syntax Options
Remarks and examples Stored results Methods and formulas References Also see

Description

`menl` fits nonlinear mixed-effects models in which some or all fixed and random effects enter nonlinearly. These models are also known as multilevel nonlinear models or hierarchical nonlinear models. The overall error distribution of the nonlinear mixed-effects model is assumed to be Gaussian. Different covariance structures are provided to model random effects and to model heteroskedasticity and correlations within lowest-level groups.

Quick start

Nonlinear mixed-effects regression of `y` on `x1` and `x2` with random intercepts `B0` by `id`

```
menl y = {a}*(1-exp(-({b0}+{b1}*x1+{b2}*x2+{B0[id]})))
```

Same as above, but using the more efficient specification of the linear combination

```
menl y = {a}*(1-exp(-{xb: x1 x2 B0[id]}))
```

Same as above, but using `define()` to specify the linear combination

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id])
```

Same as above, but perform restricted maximum-likelihood estimation instead of the default maximum-likelihood estimation

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id]) reml
```

Specify your own initial values for fixed effects, but use the default expectation-maximization (EM) method to obtain initial values for random-effects parameters

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id])        ///
  initial({a} 1 {xb:x1} 1 {xb:x2} 0.5 {xb:_cons} 2, fixed)
```

Include random intercepts `A0` by `id` to allow parameter `a` to vary between levels of `id`, and specify the `xb` suboption to indicate that `a:` contains a linear combination rather than a scalar parameter

```
menl y = {a:}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id])        ///
  define(a: A0[id], xb)
```

Include a random slope on continuous variable `x2` in the linear combination, and allow correlation between random slopes `B1` and intercepts `B0`

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id] c.x2#B1[id]) ///
  covariance(B0 B1, unstructured)
```

Specify a heteroskedastic within-subject error variance that varies as a power of `x2`

```
menl y = {a}*(1-exp(-{xb:})), define(xb: x1 x2 B0[id] c.x2#B1[id]) ///
  covariance(B0 B1, unstructured) resvariance(power x2)
```

Display random-effects and within-group error parameters as standard deviations and correlations

```
menl, stddeviations
```

Fit a nonlinear marginal regression of y on variables x1, x2, and x3 with an exchangeable covariance structure for the within-id errors

```
menl y = {phi1}*(1-exp(-0.5*(x1-{phi2: x2 i.x3}))),       ///
  rescovariance(exchangeable, group(id))
```

Three-level nonlinear regression of y on variable time and factor variable f with random intercepts S0 by lev3 and W0 by lev2 nested within lev3, using an AR(1) correlation structure for the errors

```
menl y = {phi1:}+{phi2:}*exp(-{phi3}*time),           ///
  define(phi1: i.f S0[lev3]) define(phi2: i.f W0[lev3>lev2]) ///
  rescorrelation(ar 1, t(time))
```

Three-level nonlinear regression of y on x1 with random intercepts W0 and slopes W1 on continuous x1 by lev3 and with random intercepts S0 and slopes S1 on x1 by lev2 nested within lev3, using unstructured covariance for W0 and W1 and exchangeable covariance for S0 and S1

```
menl y = {phi1:}+{b1}*cos({b2}*x1),                 ///
  define(phi1:x1 W0[lev3] S0[lev3>lev2]               ///
         c.x1#(W1[lev3] S1[lev3>lev2]))             ///
  covariance(W0 W1, unstructured)                   ///
  covariance(S0 S1, exchangeable)
```

Same as above, but assume that the errors are independent but have different variances for males and females

```
menl y = {phi1:}+{b1}*cos({b2}*x1),                 ///
  define(phi1:x1 W0[lev3] S0[lev3>lev2]               ///
         c.x1#(W1[lev3] S1[lev3>lev2]))             ///
  covariance(W0 W1, unstructured)                   ///
  covariance(S0 S1, exchangeable)                   ///
  rescovariance(identity, by(female))
```

Menu

Statistics > Multilevel mixed-effects models > Nonlinear regression

Syntax

```
menl depvar = <menlexpr> [ if ] [ in ] [ , options ]
```

<menlexpr> defines a nonlinear regression function as a substitutable expression that contains model parameters and random effects specified in braces `{}`, as in `exp({b}+{U[id]})`; see *Random-effects substitutable expressions* for details.

options	Description
Model	
`mle`	fit model via maximum likelihood; the default
`reml`	fit model via restricted maximum likelihood
`define(`*name*`:<`*resubexpr*`>)`	define a function of model parameters; this option may be repeated
`covariance(`*covspec*`)`	variance–covariance structure of the random effects; this option may be repeated
`initial(`*initial_values*`)`	initial values for parameters
Residuals	
`rescovariance(`*rescovspec*`)`	covariance structure for within-group errors
`resvariance(`*resvarspec*`)`	heteroskedastic variance structure for within-group errors
`rescorrelation(`*rescorrspec*`)`	correlation structure for within-group errors
Time series	
`tsorder(`*varname*`)`	specify time variable to determine the ordering for time-series operators
`tsinit({`*name*`:}=<`*resubexpr*`>)`	specify initial conditions for lag operators used with named expressions; this option may be repeated
`tsmissing`	keep observations with missing values in *depvar* in computation
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`variance`	show random-effects and within-group error parameter estimates as variances and covariances; the default
`stddeviations`	show random-effects and within-group error parameter estimates as standard deviations and correlations
`noretable`	suppress random-effects table
`nofetable`	suppress fixed-effects table
`estmetric`	show parameter estimates as stored in `e(b)`
`nolegend`	suppress table expression legend
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
`nostderr`	do not estimate standard errors of random-effects parameters
`lrtest`	perform a likelihood-ratio test to compare the nonlinear mixed-effects model with ordinary nonlinear regression
`notsshow`	do not show ts setting information
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling

EM options

emiterate(#)	number of EM iterations; default is emiterate(25)
emtolerance(#)	EM convergence tolerance; default is emtolerance(1e-10)
emlog	show EM iteration log

Maximization

menlmaxopts	control the maximization process
coeflegend	display legend instead of statistics

collect is allowed; see [U] 11.1.10 Prefix commands.

coeflegend does not appear in the dialog box.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

The syntax of *covspec* is

rename1 rename2 [. . .] , *vartype*

vartype	Description
independent	one unique variance parameter per random effect; all covariances are 0; the default
exchangeable	equal variances for random effects and one common pairwise covariance
identity	equal variances for random effects; all covariances are 0
unstructured	all variances and covariances to be distinctly estimated

The syntax of *rescovspec* is

rescov [, *rescovopts*]

rescov	Description
identity	uncorrelated within-group errors with one common variance; the default
independent	uncorrelated within-group errors with distinct variances
exchangeable	within-group errors with equal variances and one common covariance
ar [#]	within-group errors with autoregressive (AR) structure of order #, AR(#); ar 1 is implied by ar
ma [#]	within-group errors with moving-average (MA) structure of order #, MA(#); ma 1 is implied by ma
ctar1	within-group errors with continuous-time AR(1) structure
toeplitz [#]	within-group errors have Toeplitz structure of order #; toeplitz implies that all matrix off-diagonals be estimated
banded [#]	within-group errors with distinct variances and covariances within first # off-diagonals; banded implies all matrix bands (unstructured)
unstructured	within-group errors with distinct variances and covariances

The syntax of *resvarspec* is

resvarfunc [, *resvaropts*]

resvarfunc	Description
identity	equal within-group error variances; the default
linear *varname*	within-group error variance varies linearly with *varname*
power *varname* | _yhat	variance function is a power of *varname* or of predicted mean
exponential *varname* | _yhat	variance function is exponential of *varname* or of predicted mean
distinct	distinct within-group error variances

The syntax of *rescorrspec* is

rescorr [, *rescorropts*]

rescorr	Description
identity	uncorrelated within-group errors; the default
exchangeable	within-group errors with one common correlation
ar [#]	within-group errors with AR(#) structure; ar 1 is implied by ar
ma [#]	within-group errors with MA(#) structure; ma 1 is implied by ma
ctar1	within-group errors with continuous-time AR(1) structure
toeplitz [#]	within-group errors have Toeplitz correlation structure of order #; toeplitz implies that all matrix off-diagonals be estimated
banded [#]	within-group errors with distinct correlations within first # off-diagonals; banded implies all matrix bands (unstructured)
unstructured	within-group errors with distinct correlations

Options

Model

mle and reml specify the statistical method for fitting the model.

mle, the default, specifies that the model be fit using maximum likelihood (ML).

reml specifies that the model be fit using restricted maximum likelihood (REML), also known as residual maximum likelihood.

define(*name*:<*resubexpr*>) defines a function of model parameters, <*resubexpr*>, and labels it as *name*. This option can be repeated to define multiple functions. The define() option is useful for expressions that appear multiple times in the main nonlinear specification *menlexpr*: you define the expression once and then simply refer to it by using {*name*:} in the nonlinear specification. This option can also be used for notational convenience. See *Random-effects substitutable expressions* for how to specify <*resubexpr*>. <*resubexpr*> within define() may not contain the lagged predicted mean function.

covariance(*rename1 rename2* [. . .], *vartype*) specifies the structure of the covariance matrix for the random effects. *rename1*, *rename2*, and so on, are the names of the random effects to be correlated (see *Random effects*), and *vartype* is one of the following: independent, exchangeable, identity, or unstructured. Instead of *rename*s, you can specify *restub*∗ to refer to random effects that share the same *restub* in their names.

`independent` allows for a distinct variance for each random effect and assumes that all covariances are 0; the default.

`exchangeable` specifies one common variance for all random effects and one common pairwise covariance.

`identity` is short for "multiple of the identity"; that is, all variances are equal, and all covariances are 0.

`unstructured` allows for all variances and covariances to be distinct. If p random effects are specified, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`initial(`*initial_values*`)` specifies the initial values for model parameters. You can specify a $1 \times k$ matrix, where k is the total number of parameters in the model, or you can specify a parameter name, its initial value, another parameter name, its initial value, and so on. For example, to initialize `{alpha}` to 1.23 and `{delta}` to 4.57, you would type

```
. menl ..., initial(alpha 1.23 delta 4.57) ...
```

To initialize multiple parameters that have the same group name, for example, `{y:x1}` and `{y:x2}`, with the same initial value, you can simply type

```
. menl ..., initial({y:} 1) ...
```

For the full specification, see *Specifying initial values*.

Residuals

`menl` provides two ways to model the within-group error covariance structure, sometimes also referred to as residual covariance structure in the literature. You can model the covariance directly by using the `rescovariance()` option or indirectly by using the `resvariance()` and `rescorrelation()` options.

`rescovariance(`*rescov* [`,` *rescovopts*]`)` specifies the within-group errors covariance structure or covariance structure of the errors within the lowest-level group of the nonlinear mixed-effects model. For example, if you are modeling random effects for classes nested within schools, then `rescovariance()` refers to the residual variance–covariance structure of the observations within classes, the lowest-level groups.

rescov is one of the following: `identity`, `independent`, `exchangeable`, `ar` [*#*], `ma` [*#*], `ctar1`, `toeplitz` [*#*], `banded` [*#*], or `unstructured`. Below, we describe each *rescov* with its specific options *rescovopts*:

`identity` [`, by(`*byvar*`)`], the default, specifies that all within-group errors be independent and identically distributed (i.i.d.) with one common error variance σ^2_ϵ. When combined with `by(`*byvar*`)`, independence is still assumed, but you estimate a distinct variance for each category of *byvar*.

`independent, index(`*varname*`)` [`group(`*grpvar*`)`] specifies that within-group errors are independent with distinct variances for each value (index) of *varname*. `index(`*varname*`)` is required. `group(`*grpvar*`)` is required if there are no random effects in the model.

`exchangeable` [`, by(`*byvar*`) group(`*grpvar*`)`] assumes that within-group errors have equal variances and a common covariance.

`ar` [*#*]`, t(`*timevar*`)` [`by(`*byvar*`) group(`*grpvar*`)`] assumes that within-group errors have an AR(#) structure. If # is omitted, `ar 1` is assumed. `t(`*timevar*`)` is required. For this structure, $\#+1$ parameters are estimated: # AR coefficients and one overall error variance, σ^2_ϵ.

`ma` [*#*], `t(`*timevar*`)` [`by(`*byvar*`) group(`*grpvar*`)`] assumes that within-group errors have an MA(*#*) structure. If *#* is omitted, `ma 1` is assumed. `t(`*timevar*`)` is required. For this structure, $\# + 1$ parameters are estimated: *#* MA coefficients and one overall error variance, σ^2_ϵ.

`ctar1, t(`*timevar*`)` [`by(`*byvar*`) group(`*grpvar*`)`] assumes that within-group errors have a continuous-time AR(1) structure. This is a generalization of the AR covariance structure that allows for unequally spaced and noninteger time values. `t(`*timevar*`)` is required. For this structure, two parameters are estimated: the correlation parameter, ρ, and one overall error variance, σ^2_ϵ. The correlation between two error terms is the parameter ρ raised to a power equal to the absolute value of the difference between the `t()` values for those errors.

`toeplitz` [*#*], `t(`*timevar*`)` [`by(`*byvar*`) group(`*grpvar*`)`] assumes that within-group errors have a Toeplitz structure of order *#*, for which correlations are constant with respect to time lags less than or equal to *#* and are 0 for lags greater than *#*. *#* is an integer between 1 and the maximum observed lag (the default). `t(`*timevar*`)` is required. For this structure, $\# + 1$ parameters are estimated: *#* correlations and one overall error variance, σ^2_ϵ.

`banded` [*#*], `index(`*varname*`)` [`group(`*grpvar*`)`] is a special case of `unstructured` that restricts estimation to the covariances within the first *#* off-diagonals and sets the covariances outside this band to 0. `index(`*varname*`)` is required. *#* is an integer between 0 and $L - 1$, where L is the number of levels of `index()`. By default, *#* is $L - 1$; that is, all elements of the covariance matrix are estimated. When *#* is 0, only the diagonal elements of the covariance matrix are estimated. `group(`*grpvar*`)` is required if there are no random effects in the model.

`unstructured, index(`*varname*`)` [`group(`*grpvar*`)`] assumes that within-group errors have distinct variances and covariances. This is the most general covariance structure in that no structure is imposed on the covariance parameters. `index(`*varname*`)` is required. When you have L levels of `index()`, then $L(L+1)/2$ parameters are estimated. `group(`*grpvar*`)` is required if there are no random effects in the model.

rescovopts are `index(`*varname*`)`, `t(`*timevar*`)`, `by(`*byvar*`)`, and `group(`*grpvar*`)`.

`index(`*varname*`)` is used within the `rescovariance()` option with *rescov* `independent`, `banded`, or `unstructured`. *varname* is a nonnegative-integer–valued variable that identifies the observations within the lowest-level groups (for example, `obsid`). The groups may be unbalanced in that different groups may have different `index()` values, but you may not have repeated `index()` values within any particular group.

`t(`*timevar*`)` is used within the `rescovariance()` option to specify a time variable for the `ar`, `ma`, `ctar1`, and `toeplitz` structures.

With *rescov* `ar`, `ma`, and `toeplitz`, *timevar* is an integer-valued time variable used to order the observations within the lowest-level groups and to determine the lags between successive observations. Any nonconsecutive time values will be treated as gaps.

With *rescov* `ctar1`, *timevar* is a real-valued time variable.

`by(`*byvar*`)` is for use within the `rescovariance()` option and specifies that a set of distinct within-group error covariance parameters be estimated for each category of *byvar*. In other words, you can use `by()` to model heteroskedasticity. *byvar* must be nonnegative-integer valued and constant within the lowest-level groups.

`group(`*grpvar*`)` is used to identify the lowest-level groups (panels) when modeling within-group error covariance structures. *grpvar* is a nonnegative-integer–valued group membership variable. This option lets you model within-group error covariance structures at the lowest level

of your model hierarchy without having to include random effects at that level in your model. This is useful, for instance, when fitting nonlinear marginal or population-averaged models that model the dependence between error terms directly, without introducing random effects; see example 19. In the presence of random effects at other levels of hierarchy in your model, *grpvar* is assumed to be nested within those levels.

`resvariance(`*resvarfunc* [`,` *resvaropts*]`)` specifies a heteroskedastic variance structure of the within-group errors. It can be used with the `rescorrelation()` option to specify flexible within-group error covariance structures. The heteroskedastic variance structure is modeled as $\text{Var}\left(\epsilon_{ij}\right) = \sigma^2 g^2\left(\boldsymbol{\delta}, v_{ij}\right)$, where σ is an unknown scale parameter, $g(\cdot)$ is a function that models heteroskedasticity (also known as variance function in the literature), $\boldsymbol{\delta}$ is a vector of unknown parameters of the variance function, and v_{ij}'s are the values of a fixed covariate x_{ij} or of the predicted mean $\widehat{\mu}_{ij}$.

resvarfunc, omitting the arguments, is one of the following: `identity`, `linear`, `power`, `exponential`, or `distinct`, and *resvaropts* are options specific to each variance function.

`identity`, the default, specifies a homoskedastic variance structure for the within-group errors; $g\left(\boldsymbol{\delta}, v_{ij}\right) = 1$, so that $\text{Var}\left(\epsilon_{ij}\right) = \sigma^2 = \sigma^2_\epsilon$.

`linear` *varname* specifies that the within-group error variance vary linearly with *varname*; that is, $g\left(\boldsymbol{\delta}, v_{ij}\right) = \sqrt{varname_{ij}}$, so that $\text{Var}\left(\epsilon_{ij}\right) = \sigma^2 varname_{ij}$. *varname* must be positive.

`power` *varname*`,` | `_yhat` [`, strata(`*stratavar*`)` noconstant] specifies that the within-group error variance, or more precisely the variance function, be expressed in terms of a power of either *varname* or the predicted mean `_yhat`, plus a constant term; $g\left(\boldsymbol{\delta}, v_{ij}\right) = |v_{ij}|^{\delta_1} + \delta_2$. If `noconstant` is specified, the constant term δ_2 is suppressed. In general, three parameters are estimated: a scale parameter σ, the power δ_1, and the constant term δ_2. When `strata(`*stratavar*`)` is specified, the power and constant parameters (but not the scale) are distinctly estimated for each stratum. A total number of $2L + 1$ parameters are estimated (L power parameters, L constant parameters, and scale σ), where L is the number of strata defined by variable *stratavar*.

`exponential` *varname* | `_yhat` [`, strata(`*stratavar*`)`] specifies that the within-group error variance vary exponentially with *varname* or with the predicted mean `_yhat`; $g\left(\gamma, v_{ij}\right) = \exp(\gamma v_{ij})$. Two parameters are estimated: a scale parameter σ and an exponential parameter γ. When `strata(`*stratavar*`)` is specified, the exponential parameter γ (but not scale σ) is distinctly estimated for each stratum. A total number of $L + 1$ parameters are estimated (L exponential parameters and scale σ), where L is the number of strata defined by variable *stratavar*.

`distinct, index(`*varname*`)` [`group(`*grpvar*`)`] specifies that the within-group errors have distinct variances, σ^2_l, for each value (index), l, of *varname*, v_{ij}; $g\left(\boldsymbol{\delta}, v_{ij}\right) = \delta_{v_{ij}}$ with $\delta_{v_{ij}} = \sigma_{v_{ij}}/\sigma_1$ ($\delta_1 = 1$ for identifiability purposes) such that $\text{Var}\left(\epsilon_{ij}\right) = \sigma^2_{v_{ij}} = \sigma^2_1 \delta^2_{v_{ij}}$ for $l = 1, 2, \ldots, L$ and $v_{ij} \in \{1, 2, \ldots, L\}$. `index(`*varname*`)` is required. `group(`*grpvar*`)` is required if there are no random effects in the model. `resvariance(distinct)` in combination with `rescorrelation(identity)` is equivalent to `rescovariance(independent)`.

resvaropts are `strata(`*stratavar*`)`, noconstant, `index()`, and `group(`*grpvar*`)`.

`strata(`*stratavar*`)` is used within the `resvariance()` option with *resvarfunc* `power` and `exponential`. `strata()` specifies that the parameters of the variance function $g(\cdot)$ be distinctly estimated for each stratum. The scale parameter σ remains constant across strata. In contrast, `rescovariance()`'s `by(`*byvar*`)` suboption specifies that all covariance parameters, including σ (whenever applicable), be estimated distinctly for each category of *byvar*. *stratavar* must be nonnegative-integer valued and constant within the lowest-level groups.

noconstant is used within the resvariance() option with *resvarfunc* power. noconstant specifies that the constant parameter be suppressed in the expression of the variance function $g(\cdot)$.

index(*varname*) is used within the resvariance() option with *resvarfunc* distinct. *varname* is a nonnegative-integer–valued variable that identifies the observations within the lowest-level groups (for example, obsid). The groups may be unbalanced in that different groups may have different index() values, but you may not have repeated index() values within any particular group.

group(*grpvar*) is used within the resvariance() option with *resvarfunc* distinct. It identifies the lowest-level groups (panels) when no random effects are included in the model specification such as with nonlinear marginal models. *grpvar* is a nonnegative-integer–valued group membership variable.

rescorrelation(*rescorr* [, *rescorropts*]) specifies a correlation structure of the within-group errors. It can be used with the resvariance() option to specify flexible within-group error covariance structures.

rescorr is one of the following: identity, exchangeable, ar [#], ma [#], ctar1, toeplitz [#], banded [#], or unstructured.

identity, the default, specifies that all within-group error correlations be zeros.

exchangeable [, by(*byvar*) group(*grpvar*)] assumes that within-group errors have a common correlation.

ar [#], t(*timevar*) [by(*byvar*) group(*grpvar*)] assumes that within-group errors have an AR(#) correlation structure. If # is omitted, ar 1 is assumed. The t(*timevar*) option is required. For this structure, # AR coefficients are estimated.

ma [#], t(*timevar*) [by(*byvar*) group(*grpvar*)] assumes that within-group errors have an MA(#) correlation structure. If # is omitted, ma 1 is assumed. The t(*timevar*) option is required. For this structure, # MA coefficients are estimated.

ctar1, t(*timevar*) [by(*byvar*) group(*grpvar*)] assumes that within-group errors have a continuous-time AR(1) correlation structure. The t(*timevar*) option is required. The correlation between two errors is the parameter ρ raised to a power equal to the absolute value of the difference between the t() values for those errors.

toeplitz [#], t(*timevar*) [by(*byvar*) group(*grpvar*)] assumes that within-group errors have a Toeplitz correlation structure of order #, for which correlations are constant with respect to time lags less than or equal to # and are 0 for lags greater than #. # is an integer between 1 and the maximum observed lag (the default). t(*timevar*) is required. For this structure, # correlation parameters are estimated.

banded [#], index(*varname*) [group(*grpvar*)] is a special case of unstructured that restricts estimation to the correlations within the first # off-diagonals and sets the correlations outside this band to 0. index(*varname*) is required. # is an integer between 0 and $L - 1$, where L is the number of levels of index(). By default, # is $L - 1$; that is, all elements of the correlation matrix are estimated. When # is 0, the correlation matrix is assumed to be identity. group(*grpvar*) is required if there are no random effects in the model.

`unstructured, index(`*varname*`)` [`group(`*grpvar*`)`] assumes that within-group errors have distinct correlations. This is the most general correlation structure in that no structure is imposed on the correlation parameters. `index(`*varname*`)` is required. `group(`*grpvar*`)` is required if there are no random effects in the model.

rescorropts are `index(`*varname*`)`, `t(`*timevar*`)`, `by(`*byvar*`)`, and `group(`*grpvar*`)`.

`index(`*varname*`)` is used within the `rescorrelation()` option with *rescorr* `banded` or `unstructured`. *varname* is a nonnegative-integer–valued variable that identifies the observations within the lowest-level groups (for example, `obsid`). The groups may be unbalanced in that different groups may have different `index()` values, but you may not have repeated `index()` values within any particular group.

`t(`*timevar*`)` is used within the `rescorrelation()` option to specify a time variable for the `ar`, `ma`, `ctar1`, and `toeplitz` structures.

With *rescorr* `ar`, `ma`, and `toeplitz`, *timevar* is an integer-valued time variable used to order the observations within the lowest-level groups and to determine the lags between successive observations. Any nonconsecutive time values will be treated as gaps.

With *rescorr* `ctar1`, *timevar* is a real-valued time variable.

`by(`*byvar*`)` is used within the `rescorrelation()` option and specifies that a set of distinct within-group error correlation parameters be estimated for each category of *byvar*. *byvar* must be nonnegative-integer valued and constant within the lowest-level groups.

`group(`*grpvar*`)` is used to identify the lowest-level groups (panels) when modeling within-group error correlation structures. *grpvar* is a nonnegative-integer–valued group membership variable. This option lets you model within-group error correlation structures at the lowest level of your model hierarchy without having to include random effects at that level in your model. This is useful, for instance, when fitting nonlinear marginal or population-averaged models that model the dependence between error terms directly, without introducing random effects; see example 19. In the presence of random effects at other levels of hierarchy in your model, *grpvar* is assumed to be nested within those levels.

Time series

`tsorder(`*varname*`)` specifies the time variable that determines the time order for time-series operators used in expressions; see *Time-series operators*. When you use time-series operators with `menl`, you must either `tsset` your data prior to executing `menl` or specify option `tsorder()`. When you specify `tsorder()`, `menl` uses the time variable *varname* to create a new temporary variable that contains consecutive integers, which determine the sort order of observations within the lowest-level group. `menl` also creates and uses the appropriate panel variable based on the hierarchy of your model specification and the estimation sample; see example 17 and example 18.

`tsinit({`*name*`:}=<`*resubexpr*`>)` specifies an initial condition for the named expression *name* used with the one-period lag operator, `L.{`*name*`:}` or `L1.{`*name*`:}`, in the model specification. *name* can be the *depvar* or the name of a function of model parameters previously defined in, for instance, option `define()`. If you include the lagged predicted mean function `L.{`*depvar*`:}` or, equivalently, `L._yhat` in your model, you must specify its initial condition in `tsinit({`*depvar*`:}=...)`. The initial condition can be expressed as a random-effects substitutable expression, <*resubexpr*>. Option `tsinit()` may be repeated. Also see *Time-series operators*, example 17, and example 18.

tsmissing specifies that observations containing system missing values (.) in *depvar* be retained in the computation when a lagged named expression is used in the model specification. Extended missing values in *depvar* are excluded. Both missing and nonmissing observations are used to evaluate the predicted nonlinear mean function but only nonmissing observations are used to evaluate the likelihood. Observations containing missing values in variables used in the model other than the dependent variable are excluded. This option is often used when subjects have intermittent *depvar* measurements and the lagged predicted mean function, L.{*depvar*:} or L._yhat, is used in the model specification. Such models are common in pharmacokinetics; see example 17 and example 18.

Reporting

level(#); see [R] **Estimation options**.

variance, the default, displays the random-effects and within-group error parameter estimates as variances and covariances.

stddeviations displays the random-effects and within-group error parameter estimates as standard deviations and correlations.

noretable suppresses the random-effects table from the output.

nofetable suppresses the fixed-effects table from the output.

estmetric displays all parameter estimates in one table using the metric in which they are stored in e(b). Random-effects parameter estimates are stored as log standard-deviations and hyperbolic arctangents of correlations. Within-group error parameter estimates are stored as log standard-deviations and, when applicable, as hyperbolic arctangents of correlations. Note that fixed-effects estimates are always stored and displayed in the same metric.

nolegend suppresses the expression legend that appears before the fixed-effects estimation table when functions of parameters or named substitutable expressions are specified in the main equation or in the define() options.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

nostderr prevents menl from calculating standard errors for the estimated random-effects parameters, although standard errors are still provided for the fixed-effects parameters. Specifying this option will speed up computation times.

lrtest specifies to fit a reference nonlinear regression model and to use this model in calculating a likelihood-ratio test, comparing the nonlinear mixed-effects model with ordinary nonlinear regression.

notsshow prevents menl from showing the key ts variables; see [TS] **tsset**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

EM options

These options control the EM iterations that occur before estimation switches to the Lindstrom–Bates method. EM is used to obtain starting values.

emiterate(#) specifies the number of EM iterations to perform. The default is emiterate(25).

`emtolerance(#)` specifies the convergence tolerance for the EM algorithm. The default is `emtolerance(1e-10)`. EM iterations will be halted once the log (restricted) likelihood changes by a relative amount less than #. At that point, optimization switches to the Lindstrom–Bates method.

`emlog` specifies that the EM iteration log be shown. The EM iteration log is not displayed by default.

Maximization

menlmaxopts: `iterate(#)`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, `nonrtolerance`, `pnlsopts()`, `lmeopts()`, [`no`]`log`. The convergence is declared when either `tolerance()` or `ltolerance()` is satisfied; see *Stopping rules* for details.

menlmaxopts control the maximization process of the Lindstrom–Bates, the generalized nonlinear least-squares (GNLS), and the nonlinear least-squares (NLS) algorithms. The Lindstrom–Bates algorithm is the main optimization algorithm used for nonlinear models containing random effects. The GNLS algorithm is used for the models without random effects but with non-i.i.d. errors. The NLS algorithm is used for the models without random effects and with i.i.d. errors. The Lindstrom–Bates and GNLS algorithms are alternating algorithms—they alternate between two optimization steps and thus support options to control the overall optimization as well as the optimization of each step. The Lindstrom–Bates algorithm alternates between the penalized least-squares (PNLS) and the linear mixed-effects (LME) optimization steps. The GNLS algorithm alternates between the GNLS and ML or, if option `reml` is used, REML steps. Option `pnlsopts()` controls the PNLS and GNLS steps, and option `lmeopts()` controls the LME and ML/REML steps. The other *menlmaxopts* control the overall optimization of the alternating algorithms as well as the NLS optimization.

`iterate(#)` specifies the maximum number of iterations for the alternating algorithms and the NLS algorithm. One alternating iteration of the Lindstrom–Bates algorithm involves $\#_{\text{pnls}}$ PNLS iterations as specified in `pnlsopts()`'s `iterate()` suboption and $\#_{\text{lme}}$ LME iterations as specified in `lmeopts()`'s `iterate()` suboption. Similarly, one alternating iteration of the GNLS algorithm involves $\#_{\text{gnls}}$ GNLS iterations and $\#_{\text{ml}}$ ML/REML iterations. The default is the number set using `set maxiter`, which is 300 by default.

`tolerance(#)` specifies the tolerance for the parameter vector in the alternating algorithms and the NLS algorithm. When the relative change in the parameter vector from one (alternating) iteration to the next is less than or equal to `tolerance()`, the parameter convergence is satisfied. The default is `tolerance(1e-6)`.

`ltolerance(#)` specifies the tolerance for the linearization log likelihood of the Lindstrom–Bates algorithm and for the log likelihood of the GNLS and NLS algorithms. The linearization log likelihood is the log likelihood from the LME optimization step in the last iteration. When the relative change in the log likelihood from one (alternating) iteration to the next is less than or equal to `ltolerance()`, the log-likelihood convergence is satisfied. The default is `ltolerance(1e-7)`.

`nrtolerance(#)` and `nonrtolerance` control the tolerance for the scaled gradient.

`nrtolerance(#)` specifies the tolerance for the scaled gradient. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

`nonrtolerance` specifies that the default `nrtolerance()` criterion be turned off.

`nrtolerance(#)` and `nonrtolerance` are allowed only with the NLS algorithm.

`pnlsopts(`*pnlsopts*`)` controls the PNLS optimization step of the Lindstrom–Bates alternating algorithm and the GNLS optimization step of the GNLS alternating algorithm. *pnlsopts* include any of the following: `iterate(#)`, `ltolerance(#)`, `tolerance(#)`, `nrtolerance(#)`, and *maximize_options*. The convergence of this step within each alternating iteration is declared when `nrtolerance()` and one of `tolerance()` or `ltolerance()` are satisfied. This option is not allowed with the NLS algorithm.

`iterate(#)` specifies the maximum number of iterations for the PNLS and GNLS optimization steps of the alternating algorithms. The default is `iterate(5)`.

`ltolerance(#)` specifies the tolerance for the objective function in the PNLS and GNLS optimization steps. When the relative change in the objective function from one PNLS or GNLS iteration to the next is less than or equal to `ltolerance()`, the objective-function convergence is satisfied. The default is `ltolerance(1e-7)`.

`tolerance(#)` specifies the tolerance for the vector of fixed-effects parameters. When the relative change in the coefficient vector from one PNLS or GNLS iteration to the next is less than or equal to `tolerance()`, the parameter convergence criterion is satisfied. The default is `tolerance(1e-6)`.

`nrtolerance(#)` specifies the tolerance for the scaled gradient in the PNLS and GNLS optimization steps. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

maximize_options are [`no`]`log`, `trace`, `showtolerance`, `nonrtolerance`; see [R] **Maximize**.

`lmeopts(`*lmeopts*`)` controls the LME optimization step of the Lindstrom–Bates alternating algorithm and the ML/REML optimization step of the GNLS alternating algorithm. *lmeopts* include any of the following: `iterate(#)`, `ltolerance(#)`, `tolerance(#)`, `nrtolerance(#)`, and *maximize_options*. The convergence of this step within each alternating iteration is declared when `nrtolerance()` and one of `tolerance()` or `ltolerance()` are satisfied. This option is not allowed with the NLS algorithm.

`iterate(#)` specifies the maximum number of iterations for the LME and ML/REML optimization steps of the alternating algorithms. The default is `iterate(5)`.

`ltolerance(#)` specifies the tolerance for the log likelihood in the LME and ML/REML optimization steps. When the relative change in the log likelihood from one LME or ML/REML iteration to the next is less than or equal to `ltolerance()`, the log-likelihood convergence is satisfied. The default is `ltolerance(1e-7)`.

`tolerance(#)` specifies the tolerance for the random-effects and within-group error covariance parameters. When the relative change in the vector of parameters from one LME or ML/REML iteration to the next is less than or equal to `tolerance()`, the convergence criterion for covariance parameters is satisfied. The default is `tolerance(1e-6)`.

`nrtolerance(#)` specifies the tolerance for the scaled gradient in the LME and ML/REML optimization steps. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

maximize_options are [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `nonrtolerance`; see [R] **Maximize**.

[`no`]`log`; see [R] **Maximize**.

The following option is available with menl but is not shown in the dialog box:

coeflegend; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Random-effects substitutable expressions
Substitutable expressions
Linear combinations
Linear forms versus linear combinations
Random effects
Multilevel specifications
Time-series operators
Summary
Specifying initial values
Two-level models
Testing variance components
Random-effects covariance structures
Heteroskedastic within-group errors
Restricted maximum likelihood
Pharmacokinetic modeling
Single-dose pharmacokinetic modeling
Multiple-dose pharmacokinetic modeling
Nonlinear marginal models
Three-level models
Obtaining initial values
Linearization approach to finding initial values
Graphical approach to finding initial values
Smart regressions approach to finding initial values
Examples of specifying initial values

Introduction

Nonlinear mixed-effects (NLME) models are models containing both fixed effects and random effects where some of, or all, the fixed and random effects enter the model nonlinearly. They can be viewed as a generalization of linear mixed-effects (LME) models (see [ME] **mixed**), in which the conditional mean of the outcome given the random effects is a nonlinear function of the coefficients and random effects. Alternatively, they can be considered as an extension of nonlinear regression models for independent data (see [R] **nl**), in which coefficients may incorporate random effects, allowing them to vary across different levels of hierarchy and thus inducing correlation within observations at the same level.

Why use NLME models? Can't we use higher-order polynomial LME models or generalized linear mixed-effects (GLME) models instead?

In principle, any smooth nonlinear function can be approximated by a higher-order polynomial. One may argue that we can use an LME (see [ME] **mixed**) polynomial model and increase the order of the polynomial until we get an accurate approximation of the desired nonlinear model. There are three problems with this approach. First, parameters in NLME models often have natural physical interpretations such as half-life and limiting growth. This is not the case in LME polynomial models. For example, what is the physical interpretation of the coefficient of $\texttt{time}^4$? Second, NLME models typically use fewer parameters than the corresponding LME polynomial model, which provides a more parsimonious summarization of the data. Third, NLME models usually provide better predictions outside the range of the observed data than predictions based on LME higher-order polynomial models.

GLME models (see [ME] **meglm**) are also nonlinear, but in the restricted sense that the conditional mean response given random effects is a nonlinear function of the linear predictor that contains both fixed and random effects, and only indirectly nonlinear in fixed and random effects themselves. That is, the nonlinear function must be an invertible function of the linear predictor. However, many estimation methods for GLME and NLME models are similar because random effects enter both models nonlinearly.

Population pharmacokinetics, bioassays, and studies of biological and agricultural growth processes are just a few areas that use NLME models to analyze multilevel data such as longitudinal or repeated-measures data. Comprehensive treatments of both methodology and history of NLME models may be found in Davidian and Giltinan (1995), Vonesh and Chinchilli (1997), Demidenko (2013), and Pinheiro and Bates (2000). Davidian and Giltinan (2003) provide an excellent summary.

Consider a sample of M subjects from a population of interest, where n_j measurements, $y_{1j}, \ldots, y_{n_j j}$, are observed on subject j at times $t_{1j}, \ldots, t_{n_j j}$. By "subject", we mean any distinct experimental unit, individual, panel, or cluster with two or more correlated observations. The basic nonlinear two-level model can be written as follows (in our terminology, a one-level NLME is just a nonlinear regression model for independent data),

$$y_{ij} = \mu\left(\mathbf{x}'_{ij},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) + \epsilon_{ij} \qquad i = 1, \ldots, n_j;\ j = 1, \ldots, M \tag{1}$$

where $\mu(\cdot)$ is a real-valued function that depends on a $p \times 1$ vector of fixed effects $\boldsymbol{\beta}$, a $q \times 1$ vector of random effects $\mathbf{u}_j$, which are distributed as multivariate normal with mean $\mathbf{0}$ and variance–covariance matrix $\boldsymbol{\Sigma}$, and a covariate vector $\mathbf{x}_{ij}$ that contains both within-subject covariates $\mathbf{x}^w_{ij}$ and between-subject covariates $\mathbf{x}^b_j$. The $n_j \times 1$ vector of errors $\boldsymbol{\epsilon}_j = \left(\epsilon_{1j}, \ldots, \epsilon_{n_j j}\right)'$ is assumed to be multivariate normal with mean $\mathbf{0}$ and variance–covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$, where depending on $\boldsymbol{\Lambda}_j$, σ^2 is either a within-group error variance σ^2_ϵ or a squared scale parameter σ^2.

Parameters of NLME models often have scientifically meaningful interpretations, and research questions are formed based on them. To allow parameters to reflect phenomena of interest, (1) can be equivalently formulated as a two-stage hierarchical model as follows:

$$\begin{aligned} &\text{Stage 1: Individual-level model } y_{ij} = m\left(\mathbf{x}^w_{ij},\, \boldsymbol{\phi}_j\right) + \epsilon_{ij} && i = 1, \ldots, n_j \\ &\text{Stage 2: Group-level model } \boldsymbol{\phi}_j = \boldsymbol{d}\left(\mathbf{x}^b_j,\, \boldsymbol{\beta},\, \mathbf{u}_j\right) && j = 1, \ldots, M \end{aligned} \tag{2}$$

In stage 1, we model the response by using a function $m(\cdot)$, which describes within-subject behavior. This function depends on subject-specific parameters $\boldsymbol{\phi}_j$'s, which have a natural physical interpretation, and a vector of within-subject covariates $\mathbf{x}^w_{ij}$. In stage 2, we use a known vector-valued function $\boldsymbol{d}(\cdot)$ to model between-subject behavior, that is, to model $\boldsymbol{\phi}_j$'s and to explain how they vary across subjects. The $\boldsymbol{d}(\cdot)$ function incorporates random effects and, optionally, a vector of between-subject covariates $\mathbf{x}^b_j$. The general idea is to specify a common functional form for each subject in stage 1 and then allow some parameters to vary randomly across subjects in stage 2.

To further illustrate (1) and (2), we consider the soybean plants data (Davidian and Giltinan 1995), in which we model the average leaf weight per soybean plant, y_{ij}, in plot j at t_{ij} days after planting. Let's first use (1):

$$\begin{aligned} y_{ij} &= \mu\left(\mathbf{x}'_{ij},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) + \epsilon_{ij} \\ &= \frac{\beta_1 + u_{1j}}{1 + \exp\left[-\left\{t_{ij} - \left(\beta_2 + u_{2j}\right)\right\} / \left(\beta_3 + u_{3j}\right)\right]} + \epsilon_{ij} \end{aligned}$$

Here $\boldsymbol{\beta} = (\beta_1, \beta_2, \beta_3)'$, $\mathbf{u}_j = \left(u_{1j}, u_{2j}, u_{3j}\right)'$, and $\mathbf{x}_{ij}$ is simply t_{ij}.

Equivalently, we can use (2) to define our model,

$$\begin{aligned}
\text{Stage 1: } y_{ij} &= m\left(\mathbf{x}^{w}_{ij},\, \boldsymbol{\phi}_j\right) + \epsilon_{ij} \\
&= \frac{\phi_{1j}}{1 + \exp\left\{-\left(t_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij} \\
\text{Stage 2: } \phi_{1j} &= \beta_1 + u_{1j} \\
\phi_{2j} &= \beta_2 + u_{2j} \\
\phi_{3j} &= \beta_3 + u_{3j}
\end{aligned}$$

where $\mathbf{x}^{w}_{ij} = t_{ij}$, $\boldsymbol{\phi}_j = \left(\phi_{1j}, \phi_{2j}, \phi_{3j}\right)' = \boldsymbol{d}\left(\mathbf{x}^{b}_{j},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) = \boldsymbol{\beta} + \mathbf{u}_j$. A key advantage of (2) is the interpretability. $\boldsymbol{\phi}_j$'s are parameters that characterize features of the trajectory. For example, ϕ_{1j} can be interpreted as the asymptotic average leaf weight per soybean plant in plot j when $t_{ij} \to \infty$ and ϕ_{2j} as the time at which half of ϕ_{1j} is reached; that is, if we set $t_{ij} = \phi_{2j}$, then $E(y_{ij}) = \phi_{1j}/2$. `menl` provides both representations.

The random effects $\mathbf{u}_j$ are not directly estimated (although they may be predicted) but instead are characterized by the elements of $\boldsymbol{\Sigma}$, known as variance components, which are estimated together with the parameters of the within-group error variance–covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$. Correlation among repeated measures is induced either indirectly through the subject-specific random effects $\mathbf{u}_j$ or directly through specification of the within-subject covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$. Several covariance structures are available for $\boldsymbol{\Sigma}$, similar to those allowed in `mixed`. In contrast to `mixed`, `menl` provides more flexible modeling of the within-subject variance and correlation structures.

`menl` uses the following decomposition of the $\boldsymbol{\Lambda}_j$ matrix,

$$\boldsymbol{\Lambda}_j = \mathbf{S}_j\mathbf{C}_j\mathbf{S}_j \tag{3}$$

where $\mathbf{S}_j$ is diagonal with positive elements such that $\text{Var}\left(\epsilon_{ij}\right) = \sigma^2[\mathbf{S}_j]^2_{ii}$ and $\mathbf{C}_j$ is a correlation matrix such that $\text{corr}\left(\epsilon_{ij},\, \epsilon_{kj}\right) = [\mathbf{C}_j]_{ik}$; $[A]_{ij}$ denotes the ijth element of matrix A. Decomposition (3) of $\boldsymbol{\Lambda}_j$ allows us to separately model the variance structure (heteroskedasticity) and the correlation structure by using disjoint sets of parameters for $\mathbf{C}_j$ and $\mathbf{S}_j$. This is different from how `mixed` handles within-subject correlation, where heteroskedasticity and correlation are determined by the type of the chosen residual covariance structure. For convenience, `menl` accommodates the behavior of the `mixed` command for specifying residual covariance structures via the `rescovariance()` option. The more flexible modeling of the residual structures according to (3) is available via the `resvariance()` and `rescorrelation()` options.

For LME models, because the random effects $\mathbf{u}_j$'s are unobserved, inference about $\boldsymbol{\beta}$ and the covariance parameters are based on the marginal likelihood obtained after integrating out the random effects. Unlike LME models, no closed-form solution is available because the random effects enter the model nonlinearly, making the integration analytically intractable in all but the simplest situations. There are two principal methods proposed in the literature for fitting NLME models. One is to use an adaptive Gauss–Hermite (AGH) quadrature to approximate the integral that appears in the expression of the marginal likelihood. The other one is to use the linearization method of Lindstrom and Bates (1990), also known as a conditional first-order linearization method, which is based on a first-order Taylor-series approximation of the mean function and essentially linearizes the mean function with respect to fixed and random effects. With the AGH method, the level of accuracy increases as the number of quadrature points increases but at the expense of increasing computational burden. The linearization method is computationally efficient because it avoids the intractable integration, but the approximation cannot be made arbitrarily

accurate. Despite its potential limiting accuracy, the linearization method has proven the most popular in practice (Fitzmaurice et al. 2009, sec. 5.4.8). The linearization method of Lindstrom and Bates (1990), with extensions from Pinheiro and Bates (1995), is the method of estimation in `menl`.

The linearization method uses a first-order Taylor-series expansion of the specified nonlinear mean function to approximate it with a linear function of fixed and random effects. Thus an NLME model is approximated by an LME model, in which the fixed-effects and random-effects design matrices involve derivatives of the nonlinear mean function with respect to fixed effects (coefficients) and random effects, respectively. As such, inference after the linearization method uses the computational machinery of the LME models. For example, estimates of random effects are computed as best linear unbiased predictors (BLUPs) of random effects from the approximating LME model. The accuracy of the inferential results will depend on the accuracy of the linearization method in approximating the original NLME model. In general, asymptotic inference for the NLME models based on the linearization method is only "approximately asymptotic", making it less accurate than the corresponding asymptotic inference for true LME models. In practice, however, the linearization method was found to perform well in many situations (for example, Pinheiro and Bates [1995]; Wolfinger and Lin [1997]; Plan et al. [2012]; and Harring and Liu [2016]).

Both ML and REML estimation are supported by `menl`. The ML estimates are based on the usual application of likelihood theory, given the distributional assumptions of the model. In small samples, ML estimation generally leads to small-sample bias in the estimated variance components. The REML method (Thompson 1962) reduces this bias by forming a set of linear contrasts of the response that do not depend on the fixed effects $\boldsymbol{\beta}$ but instead depend only on the variance components to be estimated. The likelihood is then formed based on the distribution of the linear contrasts, and standard ML methods are applied.

The next section describes how to specify nonlinear expressions containing random effects in `menl`.

Random-effects substitutable expressions

You define the nonlinear model to be fit by `menl` by using a random-effects substitutable expression, a substitutable expression that contains random effects. For example, `exp({b}+{U[id]})`, `{b1}/({b2}+{b3}*x+{U[id]})`, and `({b1}+{U1[id]})/(1+{b2}*x+{c.x#U2[id]})` are a few examples of such expressions. We describe them in more detail below.

Substitutable expressions

Let's first consider substitutable expressions without random effects. Substitutable expressions are just like any other mathematical expressions involving scalars and variables, such as those you would use with Stata's `generate` command, except that the parameters to be estimated are bound in braces. See **[U] 13.2 Operators** and **[U] 13.3 Functions** for more information on expressions.

For teaching purposes, we will start with simpler substitutable expressions that do not contain random effects. Suppose that we wish to fit the model

$$y_{ij} = \alpha\left(1 - e^{-(\beta_0+\beta_1 x_{1ij}+\beta_2 x_{2ij})}\right) + \epsilon_{ij}$$

where α, β_0, β_1, and β_2 are the parameters to be estimated and ϵ_{ij} is an error term. We could simply type

```
. menl y = {a}*(1 - exp(-({b0}+{b1}*x1+{b2}*x2)))
```

Because `a`, `b0`, `b1`, and `b2` are enclosed in braces, `menl` knows that they are parameters in the model.

You can group several parameters together by assigning a group name (or equation name) to them. Parameters with the same group name, `lc` in the example below, will be grouped together in the output table:

```
. menl y = {a}*(1 - exp(-({lc:b0}+{lc:b1}*x1+{lc:b2}*x2)))
```

That is, parameters `b0`, `b1`, and `b2` will appear together in the output table in the equation labeled `lc`. Parameters without equation names will appear at the bottom of the output table.

Sometimes, it may be convenient to define subexpressions within the main expression. This can be done inside the expression itself or by using the `define()` option. For example,

```
. menl y = {a}*(1 - exp(-{xb:})), define(xb: {lc:b0}+{lc:b1}*x1+{lc:b2}*x2)
```

defines the linear predictor of the exponent in the `define()` option with label `xb` and then refers to it inside the exponent as `{xb:}`. You can define as many subexpressions as you like by using the `define()` option repeatedly. Defining subexpressions is also useful for later predictions; see, for instance, example 13.

The above is equivalent to

```
. menl y = {a}*(1 - exp(-{xb: {lc:b0}+{lc:b1}*x1+{lc:b2}*x2}))
```

Parameters `{a}`, `{lc:b0}`, `{lc:b1}`, and `{lc:b2}` are what we call "free parameters", meaning that they are not defined by a linear form, which we describe in the next section. Free parameters are displayed with a forward slash in front of their names or their group names.

The general syntax for a free parameter is

{[*eqname*:] *name*}

Linear combinations

Nonlinear functions will often contain linear combinations of variables. Recall our nonlinear function from *Substitutable expressions*:

$$y_{ij} = \alpha\left(1 - e^{-(\beta_0+\beta_1 x_{1ij}+\beta_2 x_{2ij})}\right) + \epsilon_{ij}$$

Instead of explicitly specifying the linear combination that appears in the exponent, as we did in the previous section, we can use `menl`'s shorthand notation

```
. menl y = {a}*(1 - exp(-({lc: x1 x2})))
```

By specifying `{lc:x1 x2}`, you are telling `menl` that you are declaring a linear combination named `lc` that is a function of two variables, `x1` and `x2`. `menl` will create three parameters, named `{lc:_cons}`, `{lc:x1}`, and `{lc:x2}`.

Although both specifications produce the same results, the shorthand specification is more convenient.

The general syntax for defining a linear combination is

{ *eqname*: *varspec*[, xb noconstant]}

where *varspec* includes a list of variables (*varlist*), a list of random-effects terms, or both.

The xb option is used to distinguish between the linear combination that contains one variable and a free parameter that has the same name as the variable and the same group name as the linear combination. For example, {lc: x1, xb} is equivalent to {lc:_cons} + {lc:x1}*x1, whereas {lc:x1} refers to either a free parameter x1 with a group name lc or the coefficient of the x1 variable, if {lc:} has been previously defined in the expression as a linear combination that involves variable x1; see examples below. Thus the xb option indicates that the specification is a linear combination rather than a single parameter to be estimated.

When you define a linear combination, a constant term is included by default (a mathematician would argue that "affine combination" is the correct terminology!). The noconstant option suppresses the constant.

Having defined a linear combination such as {lc:x1 x2}, you can refer to its individual coefficients by using {lc:x1} and {lc:x2} or, more generally, {*eqname*:*varname*}. For example, suppose that we want to fit the following model, where the coefficient of x1, β_1, appears in two places in the expression:

$$y_{ij} = \frac{1}{\left(1+\beta_1 x_{1ij}+\beta_2 x_{2ij}+\beta_3 x_{3ij}\right)} \exp\left\{-\left(\alpha_0+\alpha_1 z_{ij}\right)/\left(1+\beta_1 x_{4ij}\right)\right\}+\epsilon_{ij}$$

We use {lc1: x1 x2 x3, noconstant} to specify the first linear combination, which appears in the denominator outside the exponentiated expression, and then use {lc1:x1} to refer to β_1 in the denominator inside the exponentiated expression. We also use the xb option, when we specify the second linear combination that contains only one covariate z. Below is the full specification:

```
. menl y = 1/(1+{lc1: x1 x2 x3, noconstant})*exp(-{lc2: z, xb}/(1+{lc1:x1}*x4))
```

You may also refer to a "subset" of a previously defined linear combination. For example, let's modify our previous expression by substituting $\beta_1 x_{4ij}$ in the denominator in the exponent with the subset $\beta_1 x_{1ij}+\beta_3 x_{3ij}$ of the first linear combination:

$$y_{ij} = \frac{1}{\left(1+\beta_1 x_{1ij}+\beta_2 x_{2ij}+\beta_3 x_{3ij}\right)} \exp\left\{-\left(\alpha_0+\alpha_1 z_{ij}\right)/\left(1+\beta_1 x_{1ij}+\beta_3 x_{3ij}\right)\right\}+\epsilon_{ij}$$

The coefficients for variables x1 and x3 are the same in the denominators inside and outside the exponent. We fit this model by typing

```
. menl y = 1/(1+{lc1: x1 x2 x3, nocons})*     ///
            exp(-{lc2: z, xb}/(1+{lc1: x1 x3, nocons}))
```

We used the same equation name, lc1, to constrain the coefficients to be the same between the two linear-combination specifications. If we used a different equation name, say, lc3, in the last linear combination, we would have specified $\beta_4 x_{1ij}+\beta_5 x_{3ij}$ instead of $\beta_1 x_{1ij}+\beta_3 x_{3ij}$ and estimated two extra parameters, β_4 named {lc3:x1} and β_5 named {lc3:x3}.

To refer to the entire linear combination that was already defined, you can simply refer to its name. For example, if both denominators included the same linear combination, $\beta_1 x_{1ij}+\beta_2 x_{2ij}+\beta_3 x_{3ij}$, the corresponding menl specification would be

```
. menl y = 1/(1+{lc1: x1 x2 x3, nocons})*exp(-{lc2: z, xb}/(1+{lc1:}))
```

Just like subexpressions, linear combinations can be defined in the define() option. For example, the above is equivalent to

```
. menl y = 1/(1+{lc1:})*exp(-{lc2:}/(1+{lc1:})), define(lc1: x1 x2 x3, nocons) ///
                                                 define(lc2: z, xb)
```

Linear forms versus linear combinations

As we mentioned in *Linear combinations*, the linear-combination specification is syntactically convenient. It can also be more computationally efficient when a linear combination is a linear form.

A linear combination is what we call a linear form as long as you do not refer to its coefficients or any subset of the linear combination anywhere in the expression. Linear forms are beneficial for some nonlinear commands such as `nl` because they make derivative computation faster and more accurate. Although `menl` does not fully utilize the linear-form specification in its computations, it is still important to learn to distinguish between linear forms and linear combinations.

For example, in *Linear combinations*, the first linear combination `{lc:}`, the linear combination `{lc2:}`, and the linear combination `{lc1:}` in the last example are all linear forms. The linear combination `{lc1:}` in the examples where we referred to `{lc1:x1}` and `{lc1:x1 x3}` is not a linear form.

In contrast to free parameters, parameters of a linear form are displayed without forward slashes in the output. Rather, they are displayed as parameters within an equation whose name is the linear combination name. Parameters of linear combinations that are not linear forms are considered free parameters.

Random effects

So far, we have restricted our discussion to substitutable expressions that do not contain random effects. Examples of random effects specified within the `menl` syntax are `{U1[id]}`, `{U2[id1>id2]}`, `{c.x1#U3[id]}`, and `{2.f1#U4[id]}`. These represent a random intercept at the `id` level, a random intercept at the `id2`-within-`id1` level, a random slope for the continuous variable `x1`, and a random slope associated with the second level of the factor variable `f1`, respectively.

The general syntax for specifying random effects, *respec*, is provided below.

respec	Description
{*rename*[*levelspec*]}	Random intercepts *rename* at hierarchy *levelspec*
{c.*varname*#*rename*[*levelspec*]}	Random coefficients *rename* for continuous variable *varname*
{#.*fvvarname*#*rename*[*levelspec*]}	Random coefficients *rename* for the #th level of factor variable *fvvarname*

rename is a random-effects name. It is a Stata name that starts with a capital letter. *levelspec* defines the level of hierarchy and is described below.

levelspec	Description
levelvar	variable identifying the group structure for the random effect at that level
lv2 > *lv1*	two-level nesting: levels of variable *lv1* are nested within *lv2*
lv3 > *lv2* > *lv1*	three-level nesting: levels of variable *lv1* are nested within *lv2*, which is nested within *lv3*
. . . > *lv3* > *lv2* > *lv1*	higher-level nesting

You can equivalently specify levels in the opposite order, from the lowest level to the highest; for example, *lv1* < *lv2* < *lv3*, but they will be displayed in the canonical order, from the highest level to the lowest.

Random effects can be specified within a linear-combination specification such as `{lc_u: x1 x2 U1[id1] U2[id2>id1]}`. In this case, the curly braces around each random effect are not needed.

Let us illustrate several random-effects specifications with `menl`. In this section, we concentrate on two-level nonlinear models; see *Multilevel specifications* for higher-level models.

Suppose that we want to fit the following model:

$$y_{ij} = \frac{\alpha z_{ij} + u_{0j}}{1 + \exp\{-(\beta_0 + \beta_1 x_{1ij})\}} + \epsilon_{ij}$$

Compared with models we considered in previous sections, this model includes random effects or, specifically, random intercepts u_{0j}. Suppose that these random intercepts correspond to the levels of the `id` variable. Then, we can include them in our model by using `{U0[id]}`, where `U0` will be their name.

```
. menl y = ({a}*z+{U0[id]})/(1+exp(-({b0}+{b1}*x1)))
```

A more efficient specification is to use the linear-combination notation:

```
. menl y = {lc1: z U0[id], nocons}/(1+exp(-{lc2: x1, xb}))
```

The curly braces around `U0[id]` are removed when it is specified within a linear-combination specification.

If you need to refer to the random-effects term again in the expression, you can simply use its name. For example, suppose that our model includes the same random intercepts in both the numerator and the denominator.

$$y_{ij} = \frac{\alpha z_{ij} + u_{0j}}{1 + \exp\{-(\beta_0 + \beta_1 x_{1ij} + u_{0j})\}} + \epsilon_{ij}$$

We include random intercepts u_{0j}'s in the second linear combination by simply referring to their name, `U0`:

```
. menl y = {lc1: z U0[id], nocons}/(1+exp(-{lc2: x1 U0}))
```

If instead of u_{0j}'s, we had a different set of random intercepts, v_{0j}'s, in the denominator, we would need to specify a new set of random intercepts, say, `V0[id]`, with `menl`:

```
. menl y = {lc1: z U0[id], nocons}/(1+exp(-{lc2: x1 V0[id]}))
```

The shorthand notation for referring to random effects only by name, that is, without the brackets and the *levelspec*, is also useful when specifying the `covariance()` option, especially for multilevel random effects with long-level specifications; see *Multilevel specifications*.

Let's now see how to include random slopes. Consider the following extension of the *first*, simpler model in this subsection:

$$y_{ij} = \frac{\alpha z_{ij} + u_{0j} + u_{1j} z_{ij}}{1 + \exp\{-(\beta_0 + \beta_1 x_{1ij})\}} + \epsilon_{ij}$$

Here u_{1j} is a random slope for a continuous variable `z` and is specified as `{c.z#U1[id]}` directly or without curly braces within a linear-combination specification.

```
. menl y = {lc1: z U0[id] c.z#U1[id], nocons}/(1+exp(-{lc2: x1, xb}))
```

We can also include random slopes for factor variables. To demonstrate this, let's consider a different nonlinear model for variety. Consider the model below, where binary variables x_{1ij} and x_{2ij} correspond to the factor levels 1 and 2 of a factor variable x that takes on values 0, 1, and 2, with 0 being the base level.

$$y_{ij} = \alpha_0 + \alpha_1 z_{1ij} - \sqrt{w_{ij}^2 + \exp\left(\beta_0 + \beta_1 x_{1ij} + \beta_2 x_{2ij} + u_{0j} + u_{1j} x_{1ij} + u_{2j} x_{2ij}\right)} + \epsilon_{ij}$$

There are three random-effects terms in this model: random intercepts u_{0j}, random slopes u_{1j} for x_{1ij} (level 1 of x), and random slopes u_{2j} for x_{2ij} (level 2 of x). In Stata, for a factor variable `x`, we can use the factor-variable notation ([U] **11.4.3 Factor variables**) to refer to its levels, `1.x` for level 1 and `2.x` for level 2. So, to include the three random-effects terms in `menl`, we will use `U0[id]`, `1.x#U1[id]`, and `2.x#U2[id]`, respectively.

```
. menl y = {lc1: z1, xb} - sqrt(c.w#c.w +  ///
        exp({lc2: i.x U0[id] 1.x#U1[id] 2.x#U2[id]}))
```

In the above specification, we used the factor-variable notations `i.x` to include fixed effects for all levels of `x`, except the base level, and `c.w#c.w` to include a square of `w`; see [U] **11.4.3 Factor variables** for details. The factor-variable specification `i.` or any other specification that refers to multiple levels of a factor variable is not allowed when specifying random coefficients, because each level will typically require a different set of random effects. For example, if we had specified `i.x#U[id]` in the above example, we would have received an error.

Multilevel specifications

In *Random effects*, we focused on specifying substitutable expressions containing random effects for two-level nonlinear mixed-effects models. Here we will consider higher-level models.

Suppose that we want to fit the following three-level nonlinear mixed-effects model,

$$y_{ijk} = \beta_0 + u_{0k}^{(3)} + u_{0jk}^{(2)} + \cos\left\{\left(\beta_1 + u_{1k}^{(3)}\right) x_{1ijk}\right\} + \epsilon_{ijk}$$

where first-level observations, indexed by i, are nested within second-level groups, indexed by j, which are nested within third-level groups, indexed by k.

There are three random-effects terms in this model: random intercepts, $u_{0k}^{(3)}$, and random slopes for x_1, $u_{1k}^{(3)}$, at the third level (`idk`) and random intercepts $u_{0jk}^{(2)}$ at the second level (`idj`-nested-within-`idk`). We specify random intercepts and random slopes for x_1 at the highest hierarchical level just like we did in *Random effects* for two-level models. Specifically, we can use `U0[idk]` and `c.x1#U1[idk]`, respectively. To specify random intercepts $u_{0jk}^{(2)}$ at the `idj`-nested-within-`idk` level, we need to use one of the *levelspec* specifications for two nested levels. For example, we can use `UU0[idk>idj]`. Below is the full specification:

```
. menl y = {lc1: U0[idk] UU0[idk>idj]} + cos({lc2: x1 c.x1#U1[idk], noconstant})
```

We can also include a random slope of the `x1` variable at the `idj`-within-`idk` level in the cosine function by specifying `c.x1#UU1[idk>idj]` inside the `cos()` function.

```
. menl y = {lc1: U0[idk] UU0[idk>idj]} +   ///
        cos({lc2: x1 c.x1#U1[idk] c.x1#UU1[idk>idj], noconstant})
```

We can shorten the above specification by writing `c.x1#U1[idk] c.x1#UU1[idk>idj]` more compactly as `c.x1#(U1[idk] UU1[idk>idj])`,

```
. menl y = {lc1: U0[idk] UU0[idk>idj]} +   ///
        cos({lc2: x1 c.x1#(U1[idk] UU1[idk>idj]), noconstant})
```

Similarly, if we had a four-level model with, say, a random intercept at the idj-within-idk-within-idl level, we could specify it as W[idl>idk>idj]; see *levelspec* for other specifications.

Time-series operators

You can use time-series operators in the specification of your nonlinear model (see **[U] 11.4.4 Time-series varlists**) but with some exceptions described next. You can use time-series operators in the main nonlinear specification <*menlexpr*> or any random-effects substitutable expression <*resubexpr*>. The supported time-series operators include L. and L#., F. and F#., and D. and D#.. You cannot combine time-series operators or use them with a list of variables. Also, you cannot combine time-series operators with factor variables.

You can also include the lagged predicted mean function and lagged functions of model parameters in your expressions. For brevity, we will refer to both types of lagged functions as lagged named expressions. Lagged named expressions are useful, for instance, for fitting certain pharmacokinetic models; see example 17 and example 18.

To include the lagged predicted mean function, you can use the specification L.{*depvar*:} or, equivalently, L._yhat. (Do not confuse this with the lagged dependent variable specification L.*depvar*.) You can specify the lagged predicted mean function only in the main nonlinear specification *menlexpr*. To include a lagged function of model parameters, you can use the specification L.{*name*:}, where *name* is the name of the previously defined function of model parameters. Such functions are typically defined in the define() options. Only the one-period lag operator, L. or L1., is supported with named expressions.

To use time-series operators, you must either tsset your data prior to executing menl or specify the tsorder() option with menl. You must specify time and panel variables with tsset. When you use the tsorder(*varname*) option, menl uses the time variable *varname* to determine the ordering for time-series operators. menl creates a new temporary time variable that takes on values 1, 2, ... in each panel for the estimation sample. menl also creates the appropriate panel variable and uses the newly generated variables with tsset. For two-level models, menl uses the specified level variable as the panel variable. With more than two levels, menl creates the panel variable as a variable that takes on values 1, 2, ... for the groups formed by all level variables in the estimation sample. The generated panel and time variables are labeled as <panel> and <time> in the output of tsset as displayed by menl.

When you use time-series operators with variables in the dataset, some of the observations are used to initialize the series for those variables. For example, if you include a lagged variable $varname_{t-1}$ (L.*varname*) in your model, the value of *varname* in the first observation in each panel is used to initialize the series; see [TS] **tsset**. But what happens when you include a lagged named expression for which there is no existing variable in the dataset? If your named expression is a function of existing variables, the values of those variables in the first observation (in each panel) will be used to compute an initial value for the lagged named expression. For some models, a named expression can depend on its own lag; see example 17 and example 18. In this case, you must specify the initial condition for it in the tsinit() option. Note that you will always need to specify the tsinit() option for the lagged predicted mean function. The tsinit() option may be repeated and may contain functions of variables and model parameters. When you specify the tsinit() option, menl uses its value (or values in the first observation of each panel) to initialize the corresponding lagged named expression. Just like with regular time-series variables, the first observation in each panel will be excluded from the estimation sample whenever you use lagged named expressions in the model.

Summary

To summarize, here are a few rules to keep in mind when defining substitutable expressions.

1. Model parameters and random effects are bound in braces if specified directly in the expression: {b0}, {U0[id]}, etc.
2. Model parameters can be assigned group names: {slopes:x1}, {slopes:x2}, etc.
3. Random-effects names must start with a capital letter as in {U0[id]}, {c.x#U1[id]}, {V0[id2>id1]}, {1.z#V1[id2>id1]}, etc.
4. The factor-variable specification i., as in {i.z#V1[id2>id1]}, or any other specification that refers to multiple levels of a factor variable, as in {i(1/4).z#V1[id2>id1]}, is not allowed when specifying random coefficients.
5. Linear combinations of variables can be included using the specification

   {*eqname*:*varlist*[, xb noconstant]}

   For example, {price: mpg weight i.rep78} and {lc: x1 x2, noconstant}.
6. Random effects can be specified within a linear combination, in which case they should be included without curly braces, for example, {lc_u: x1 x2 U[id]}.
7. To specify a linear combination that contains only one variable, use the xb option, for example, {lc: x1, xb}.
8. To refer to the previously defined linear combination again in the expression, simply use its name {*eqname*:}, for example, {lc:} and {lc_u:}.
9. You can refer to individual parameters of the linear combination by using {*eqname*:_cons} and {*eqname*:*varname*}, for example, {price:_cons} and {price:weight}.
10. You can refer to a "subset" of the previously defined linear combination by using {*eqname*:*subset*}, where *subset* is a subset of the variables from *varlist* used to define *eqname*, as in {price: mpg weight}. To refer to the subset containing only one variable, use the xb option, as in {price: weight, xb}. If a linear combination contains only one random-effects term, the xb option is implied.
11. To refer to the previously defined random effects again in the expression or in the covariance() option, simply use their names, such as {U0} and {U1}.
12. You can define subexpressions, including linear combinations, inside the main expression or in the define() option, which can be repeated. For example,

```
. menl y = {numer:}/{denom:}, define(numer: z U0[id]) ///
      define(denom:1+exp(-{lc: x1, xb}))
```

13. Specify linear forms whenever possible for faster and more accurate computation of derivatives; see *Linear forms versus linear combinations*.
14. Model parameters that are not defined by linear forms are considered free parameters. They are included in the output with a forward slash in front of their names or group names and displayed after linear forms in the estimation table.

Specifying initial values

By default, menl uses the EM algorithm to obtain initial values, but you may often need to specify your own. You specify your own initial values in the initial() option. For example, specifying the initial(a 1.1 b -2) option with menl initializes parameter {a} to 1.1 and parameter {b} to −2.

When you specify your own initial values, they are used for initialization, and the EM algorithm is not performed. When you specify initial values for only a subset of model parameters, the remaining parameters are initialized with some predetermined values such as zeros for fixed-effects parameters and correlations and ones for variances. You can specify the iterate(0) option to see the initial values that will be used by menl in the optimization.

Often, you may have good initial values for fixed-effects parameters but not for random-effects parameters. In this case, you can specify initial()'s fixed suboption to supply your own fixed-effects parameters, but use the EM algorithm to obtain initial values for the random-effects parameters.

There are three ways in which you can use the initial(*initial_values*) option: you can specify a vector of values, a list of values, or values for individual parameters and groups of parameters.

Specifically, *initial_values* is one of the following:

vectorname [, skip copy fixed]

[#] [...], copy

paramlist[=]# [*paramlist*[=]# [...]] [, fixed]

skip specifies that any parameters found in the specified initialization vector, *vectorname*, that are not also found in the model be ignored. The default action is to issue an error message.

copy specifies that the initial values be copied into the initialization vector without checking for valid column names. copy must be specified when initial values are supplied as a list of numbers.

fixed specifies that initial estimates are being supplied for the fixed effects only and that menl should still perform the EM algorithm to refine initial values for variance components. The specified initial values are used for fixed-effects parameters during the EM algorithm. If you omit fixed, menl presumes that you are specifying starting values for all parameters in e(b), and the EM algorithm will not be performed.

Examples of *paramlist* are param, {param}, {param1} {param2}, {param1 param2}, {grp:param1} {grp:param2} {grp:param3}, {grp:param1 param2}, and {grp:}.

Let's describe each specification in more detail. You can specify the name of a vector containing the initial values, say, initial(b0). Vector b0 should be properly labeled with labels found in column names of e(b), unless you specify the copy option. A properly labeled vector can have fewer elements than e(b) or, if skip is specified, even more elements. A vector without labels must be of the same dimension as e(b).

Alternatively, you can supply a list of numbers to initial(), in which case copy must be specified. The list of numbers should be of length equal to the dimension of e(b). For example, if e(b) has four parameters and you type initial(1.1 0 3 -2, copy), then the four coefficients in e(b) will be initialized to 1.1, 0, 3, and −2, respectively. If instead you specify, for example, only three initial values in your list, an error will be issued.

Finally, you can initialize parameters by referring to their names. You can specify a parameter name, its initial value, another parameter name, its initial value, and so on, for example, initial(a 1.1 b -2). You can also assign the same initial value to a group of parameters. For example, initial({a b c}

1) will initialize parameters {a}, {b}, and {c} to 1 and initial({lc:x1 x2 _cons} 0) will initialize {lc:x1}, {lc:x2}, and {lc:_cons} to 0. You can assign the same initial value to all parameters with the same group name. For example, we can shorten the previous specification to initial({lc:} 0).

Depending on the situation, it may also be beneficial to specify initial values for the NLS algorithm used by menl to obtain starting values for the EM algorithm. These initial values can be specified in the parameter definition such as {a=0.5}, in which case the NLS algorithm used during the initialization will use 0.5 as the starting value for parameter a instead of the default 0. Such initialization is particularly useful for parameters used in the denominators for which zero values may lead to an undefined value of the mean function.

See *Examples of specifying initial values* and *Obtaining initial values* for examples.

Two-level models

The sole purpose of this section and its examples is to highlight the syntax of menl and make you familiar with how to specify substitutable expressions in menl and with its output. Also see an introductory example in *Nonlinear models* in [ME] **me**.

We will use the data from the Longitudinal Study of Unicorn Health in Zootopia, which contain the brain weight (weight) of 30 newborn male unicorns and 30 newborn female unicorns. Measurements were collected at 13 occasions every 2 months over the first 2 years after birth (time). Based on previous studies, a model for unicorn brain shrinkage is believed to be

$$\texttt{weight}_{ij} = \beta_1 + (\beta_2 - \beta_1)\,\exp\left(-\beta_3 \texttt{time}_{ij}\right) + \epsilon_{ij} \quad i = 1, 2, \ldots, 13;\ j = 1, 2, \ldots, 60$$

Parameter β_1 represents the average brain weight of unicorns as $\texttt{time}_{ij}$ increases to infinity. Parameter β_2 is the average brain weight at birth (at $\texttt{time}_{ij} = 0$), and β_3 is a scale parameter that determines the rate at which the average brain weight of unicorns approaches the asymptotic weight β_1 (decay rate). This model can be fit with the nl command; see [R] **nl**.

We will start with a simple two-level model in which we allow the asymptote parameter β_1 to vary between unicorns by replacing β_1 in the above equation with $\beta_1 + u_{0j}$,

$$\texttt{weight}_{ij} = \beta_1 + u_{0j} + \left(\beta_2 - \beta_1 - u_{0j}\right)\,\exp\left(-\beta_3 \texttt{time}_{ij}\right) + \epsilon_{ij} \tag{4}$$

where β_1, β_2, and β_3 are fixed-effects parameters to be estimated and u_{0j} is a random intercept at the unicorn, id, level that follows a normal distribution with mean 0 and variance σ_u^2.

Equivalently, the model defined by (4) can be written as a two-stage model,

$$\texttt{weight}_{ij} = \phi_{1j} + \left(\phi_{2j} - \phi_{1j}\right)\,\exp\left(-\phi_{3j} \texttt{time}_{ij}\right) + \epsilon_{ij} \tag{5}$$

with the following stage 2 specification:

$$\begin{aligned} \phi_{1j} &= \beta_1 + u_{0j} \\ \phi_{2j} &= \beta_2 \\ \phi_{3j} &= \beta_3 \end{aligned} \tag{6}$$

Parameters ϕ_{1j}, ϕ_{2j}, and ϕ_{3j} now describe the behavior of the jth unicorn. For example, ϕ_{1j} represents the brain weight of the jth unicorn as $\texttt{time}_{ij}$ increases to infinity.

▷ Example 1: Simple two-level model

Let's use `menl` to first fit a single-equation model defined by (4), described above.

```
. use https://www.stata-press.com/data/r19/unicorn
(Brain shrinkage of unicorns in the land of Zootopia)
. menl weight = {b1}+{U0[id]}+({b2}-{b1}-{U0[id]})*exp(-{b3}*time)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood =  -56.97576
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
Linearization log likelihood =  -56.97576
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b1	4.707954	.1414511	33.28	0.000	4.430715	4.985193
/b2	8.089432	.0260845	310.12	0.000	8.038307	8.140556
/b3	4.13201	.0697547	59.24	0.000	3.995293	4.268726

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(U0)	1.189809	.2180036	.8308359	1.703881
var(Residual)	.0439199	.0023148	.0396095	.0486995

Notes:

1. The response variable `weight` is specified on the left-hand side of the equality sign, and parameters to be estimated are enclosed in curly braces `{b1}`, `{b2}`, and `{b3}` on the right-hand side.
2. By typing `{U0[id]}`, we specified a random intercept at the level identified by the group variable `id`, that is, the unicorn level (level two).
3. The estimation log consists of three parts:
   a. A set of EM iterations used to refine starting values. By default, the iterations themselves are not displayed, but you can display them by using the `emlog` option. NLME models may often have multiple solutions and converge to a local maximum. It is thus important to try different initial values to investigate the existence of multiple solutions and the convergence to a global maximum; see *Obtaining initial values*.
   b. A set of iterations displaying the value of the linearization log likelihood from the Lindstrom–Bates algorithm or alternating algorithm. The term "linearization" reflects the fact that the reported log likelihood corresponds to the linear mixed-effects model obtained after linearization of the specified nonlinear mean function with respect to fixed and random effects. See *Inference based on linearization* and *Stopping rules* for details about the algorithm.

c. The message "Computing standard errors". This is just to inform you that `menl` has finished its iterative maximization and is now reparameterizing the variance components (see *Methods and formulas*) to their natural metric and computing their standard errors. If you are interested only in the fixed-effects estimates, you can use the `nostderr` option to bypass this step.

4. The output title, "Mixed-effects ML nonlinear regression", informs us that our model was fit using ML, the default. For REML estimates, use the `reml` option.

5. The header information is similar to that of the `mixed` command, but unlike `mixed`, `menl` in general does not report a model χ^2 statistic in the header because a test of the joint significance of all fixed-effects parameters (except the constant term) may not be relevant in a nonlinear model.

6. The first estimation table reports the fixed effects. We estimate $\beta_1 = 4.71$, $\beta_2 = 8.09$, and $\beta_3 = 4.13$. Although z tests against zeros are reported automatically for all fixed-effects parameters, as part of standard estimation output, they may not always be of interest or even appropriate for parameters of nonlinear models. You can use the `test` command ([R] **test**) to test hypotheses of interest or reparameterize your model so that the tests of parameters against zeros are meaningful.

7. The second estimation table shows the estimated variance components. The first section of the table is labeled `id: Identity`. In general, this means that our model includes random effects at the `id` (unicorn) level and that their variance–covariance matrix, $\mathbf{\Sigma}$, is the identity matrix (all random effects have the same variance). In our example, because we have only one random effect, u_{0j}, the random-effect covariance structure is irrelevant, and the variance of the random intercept, σ_u^2, labeled as `var(U0)` in the output, is estimated as 1.19 with standard error 0.22.

8. The row labeled `var(Residual)` displays the estimated overall error variance or variance of the error term; that is, $\widehat{\text{Var}}\left(\epsilon_{ij}\right) = \hat{\sigma}_\epsilon^2 = 0.044$.

◁

▷ Example 2: Two-level model as a two-stage model, using the define() option

The model from example 1 can also be specified as a two-stage model, as defined by (5) and (6), by using the `define()` option. The `define()` option is particularly useful when you have a complicated nonlinear expression, and you would like to break it down into smaller pieces. Parameters of interest that are functions of other parameters can be defined using the `define()` option. This will make it easier to predict them for each subject after estimation; see [ME] **menl postestimation**.

Below we specify the asymptote parameter, ϕ_{1j}, by using `define()`. The colon (`:`) in `{phi1:}` instructs `menl` that `phi1` will be specified as a substitutable expression within the `define()` option. Parameters `{phi2}` and `{phi3}` are simple free parameters and thus do not need to be specified in `define()`.

```
. menl weight = {phi1:}+({phi2}-{phi1:})*exp(-{phi3}*time),
> define(phi1: {b1}+{U0[id]})
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood =  -56.97576
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
Linearization log likelihood =  -56.97576
        phi1: {b1}+{U0[id]}
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b1	4.707954	.1414511	33.28	0.000	4.430715	4.985193
/phi2	8.089432	.0260845	310.12	0.000	8.038307	8.140556
/phi3	4.13201	.0697547	59.24	0.000	3.995293	4.268726

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(U0)	1.189809	.2179903	.830854	1.703844
var(Residual)	.0439199	.0023148	.0396095	.0486995

The results are identical to those obtained in example 1, but the estimation table now has a legend that lists the expression `phi1` defined in the model. We can suppress this legend by specifying the `nolegend` option.

We could have defined `phi1` directly in the main expression instead of in the `define()` option,

```
. menl weight = {phi1:{b1}+{U0[id]}}+({phi2}-{phi1:})*exp(-{phi3}*time)
(output omitted)
```

but by using the `define()` option, we simplified the main expression.

◁

▷ Example 3: Two-level model containing covariates

Reducing brain weight loss has been an active research area in Zootopia for the past two decades, and scientists believe that consuming rainbow cupcakes right after birth may help slow down brain shrinkage. Recall that the scale parameter ϕ_{3j} determines the rate at which the brain weight of the jth unicorn decreases to its asymptotic value ϕ_{1j}. Hence, covariate `cupcake`, which represents the number of rainbow cupcakes consumed right after birth, is added to the equation of ϕ_{3j}. Also, we would like to investigate whether the asymptote, ϕ_{1j}, is gender specific, so we include the factor variable `female` in the equation for ϕ_{1j}. $\texttt{female}_j$ is 1 if the jth unicorn is a female and 0 otherwise.

The stage 2 specification of the model defined by (5) becomes

$$\begin{aligned}\phi_{1j} &= \beta_{10} + \beta_{11}\texttt{female}_j + u_{0j}\\ \phi_{2j} &= \beta_2\\ \phi_{3j} &= \beta_{30} + \beta_{31}\texttt{cupcake}_j\end{aligned} \tag{7}$$

The `define()` option can be repeated, so we specify a separate `define()` option for ϕ_{1j}, ϕ_{2j}, and ϕ_{3j}. We could have left ϕ_{2j} as a free parameter `{phi2}` in our specification, but we wanted to closely match the stage 2 specification (7).

```
. menl weight = {phi1:}+({phi2:}-{phi1:})*exp(-{phi3:}*time),
> define(phi1: {b10}+{b11}*1.female+{U0[id]})
> define(phi2: {b2})
> define(phi3: {b30}+{b31}*cupcake)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -29.014988
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
Linearization log likelihood = -29.014988
        phi1: {b10}+{b11}*1.female+{U0[id]}
        phi2: {b2}
        phi3: {b30}+{b31}*cupcake
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b10	4.072752	.1627414	25.03	0.000	3.753785	4.39172
/b11	1.264407	.2299723	5.50	0.000	.8136694	1.715144
/b2	8.088102	.0255465	316.60	0.000	8.038032	8.138172
/b30	4.706926	.1325714	35.50	0.000	4.44709	4.966761
/b31	-.2007309	.0356814	-5.63	0.000	-.2706651	-.1307966

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(U0)	.7840578	.1438924	.5471838	1.123474
var(Residual)	.0420763	.0022176	.0379468	.0466551

In the table legend, `/b10` and `/b11` correspond, respectively, to the constant term and coefficient of `1.female` in the equation for ϕ_{1j}. `/b2` is ϕ_{2j}, and `/b30` and `/b31` correspond, respectively, to the constant term and coefficient for `cupcake` in the equation for ϕ_{3j}.

Based on our results, consuming rainbow cupcakes after birth indeed slows down brain shrinkage: `/b31` is roughly -0.2 with a 95% CI of $[-0.271, -0.131]$.

◁

▷ Example 4: Specifying linear combinations

A more convenient way to specify the model in example 3 is to use linear-combination specifications; see *Random-effects substitutable expressions*.

For example, `define(phi1: {b10}+{b11}*1.female+{U0[id]})` can be replaced with `define(phi1: i.female U0[id])`. `menl` knows that we are defining ϕ_{1j} as a linear combination of `i.female` and `U0[id]` and thus will create a constant term and a coefficient for each level of factor variable `female` and will use a coefficient of 1 for any random effect. Because `female` has only two levels, `menl` will create two coefficients for `0b.female` and `1.female`, respectively, but will constrain the coefficient of the base level, level 0, to be 0.

We now fit our model by using linear-combination specifications within the `define()` options. We explain the use of the second and third `define()` specifications following estimation.

```
. menl weight = {phi1:}+({phi2:}-{phi1:})*exp(-{phi3:}*time),
> define(phi1: i.female U0[id])
> define(phi2: _cons, xb)
> define(phi3: cupcake, xb)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -29.014988

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60
                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13
                                                Wald chi2(2)      =      61.78
Linearization log likelihood = -29.014988       Prob > chi2       =     0.0000

       phi1: i.female U0[id]
       phi3: cupcake, xb
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
female						
female	1.264407	.2299723	5.50	0.000	.8136695	1.715144
_cons	4.072752	.1627414	25.03	0.000	3.753785	4.39172
phi2						
_cons	8.088102	.0255465	316.60	0.000	8.038032	8.138172
phi3						
cupcake	-.2007309	.0356814	-5.63	0.000	-.2706651	-.1307966
_cons	4.706926	.1325714	35.50	0.000	4.44709	4.966761

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(U0)	.7840578	.1438918	.5471847	1.123472
var(Residual)	.0420763	.0022176	.0379468	.0466551

By using linear-combination specifications within the `define()` options, we improved the readability of the coefficient table. For example, it is now clear that `_cons` in the equation labeled `phi3` corresponds to the constant term in the equation for ϕ_{3j}. This term was labeled `/b30` previously.

Notes:

1. The `define()` option interprets its specification as a random-effects substitutable expression, so you do not need to specify the curly braces (`{}`) around the specification.
2. All rules for random-effects substitutable expressions apply to the specifications within `define()`.
3. Following one of the rules for random-effects substitutable expressions, we used the `xb` option within `define()`s for `phi2` and `phi3`, because their linear combinations contained only one term: `_cons` for `phi2` and `cupcake` for `phi3`.
4. Specification `define(phi2: _cons, xb)` is the same as `define(phi2:, xb)`. In other words, `_cons` is implied with any linear combination, unless the `noconstant` option is specified. We chose to include `_cons` directly for clarity.
5. We could have used a free parameter `{phi2}` instead of the linear combination `{phi2: _cons, xb}`, but we wanted to preserve the order in which `phi1`, `phi2`, and `phi3` appear in the estimation table. See example 5, where we specify ϕ_{2j} as a free parameter `{phi2}`.
6. In the presence of linear combinations, `menl` reports a joint test of significance of all coefficients (except the constant term) across all linear combinations.
7. Linear combinations containing only a constant such as `{phi2:}` are not listed in the table expression legend for brevity.

◁

▷ Example 5: Including random coefficients

In previous examples, we only illustrated how to specify random intercepts such as `{U0[id]}`, and it is bad karma to end a unicorn story without showing how to specify random coefficients or random slopes.

Continuing with our model as defined by (5) and (7), let's suppose that the equation for the brain-weight scale parameter, ϕ_{3j}, is as follows:

$$\phi_{3j} = \beta_{30} + (\beta_{31} + u_{1j})\texttt{cupcake}_j$$

We incorporated a unicorn-specific random slope for variable `cupcake`. The random slope, u_{1j}, for a continuous variable `cupcake` can be specified in `menl` as `c.cupcake#U1[id]`, and by default, `menl` assumes that it is independent of the random intercept, u_{0j}. (See example 9 for specifying other random-effects covariance structures.)

```
. menl weight = {phi1:}+({phi2}-{phi1:})*exp(-{phi3:}*time),
> define(phi1: i.female U0[id])
> define(phi3: cupcake c.cupcake#U1[id])

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood =  165.41751
Iteration 2:  Linearization log likelihood =  165.42008
Iteration 3:  Linearization log likelihood =  165.42011
Iteration 4:  Linearization log likelihood =   165.4201

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        780
Group variable: id                              Number of groups  =         60

                                                Obs per group:
                                                              min =         13
                                                              avg =       13.0
                                                              max =         13

                                                Wald chi2(2)      =      46.70
Linearization log likelihood =   165.4201       Prob > chi2       =     0.0000

        phi1: i.female U0[id]
        phi3: cupcake c.cupcake#U1[id]
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
female						
female	1.320623	.2215707	5.96	0.000	.8863522	1.754894
_cons	4.006823	.1568268	25.55	0.000	3.699448	4.314198
phi3						
cupcake	-.219661	.0659984	-3.33	0.001	-.3490155	-.0903066
_cons	4.771466	.1128421	42.28	0.000	4.5503	4.992633
/phi2	8.087655	.0179406	450.80	0.000	8.052492	8.122818

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Independent				
var(U0)	.727464	.1337152	.507402	1.042968
var(U1)	.1258914	.0309569	.0777471	.2038487
var(Residual)	.0208202	.0011403	.018701	.0231795

In addition to the overall error variance and the random-intercept variance, we now have a random-slope variance, which is labeled `var(U1)` in the output. In this example, we also specified parameter ϕ_{2j} as a free parameter `{phi2}` instead of a linear combination as in example 4. As we mentioned in *Summary*, free parameters are displayed after linear combinations, so `phi2` is listed last in the estimation table.

Previous studies of unicorns considered a model that also incorporated gender-specific variation between unicorns in asymptotic weight ϕ_{1j},

$$\phi_{1j} = \beta_{10} + u_{0j} + (\beta_{11} + u_{2j})\texttt{female}_j$$

but found no statistical evidence of such variation.

If we wanted to verify this for our data, we could have fit the following model:

```
. menl weight = {phi1:}+({phi2}-{phi1:})*exp(-{phi3:}*time), ///
    define(phi1: i.female U0[id] 1.female#U2[id])           ///
    define(phi3: cupcake c.cupcake#U1[id])
```

Compared with our previous specification, we included a new term in the `define()` option for `phi1`—a random slope for level 1 of the factor variable `female`, `1.female#U2[id]`. To include random slopes for a factor variable, we must specify random effects for each level, except the base level, of the factor variable. The specification `i.`*fvvarname* for referring to all levels of a factor variable is not allowed in the context of random effects, because a different set of random effects must be defined for each level. For example, if we specified `i.female#U2[id]` in our example, we would have received an error.

◁

To summarize:

1. Use {*name*} to define free parameters such as `{b1}`.
2. Use, for example, `{U0[id]}` to define random intercepts at the `id` level, `{c.`*varname*`#U1[id]}` to define random slopes for continuous variable *varname* at the `id` level, and `{#.`*fvvarname*`#U1[id]}` for each level #, except the base level, of variable *fvvarname* to include random slopes for factor variable *fvvarname*. The specification `{i.`*fvvarname*`#U1[id]}` is not allowed.
3. Use linear-combination specifications whenever possible. Do not use {} around random effects when they are specified within a linear combination.
4. Use multiple `define()` options to specify parameters of interest that are functions of other parameters, and use linear-combination specifications within `define()` whenever possible.
5. Use the `xb` option within a linear combination or within `define()` whenever you specify one variable such as `define(phi1: cupcake, xb)`, one random effect such as `{phi2: U0[id], xb}`, or a constant-only linear combination such as `{phi2: _cons, xb}` or `{phi2: , xb}`. When you specify the `xb` option, the above specifications are interpreted by `menl`, respectively, as a linear combination `{phi1:_cons}+{phi1:cupcake}*cupcake`, a linear combination `{phi:_cons}+{U0[id]}`, and a constant term `{phi2:_cons}`.
6. Unicorns do exist in our world, they are just gray, fat, and called rhinos.

Testing variance components

Consider data on the intensity of 23 large earthquakes in western North America between 1940 and 1980 analyzed originally by Joyner and Boore (1981) and then also by Davidian and Giltinan (1995, sec. 11.4). The objective of the study was to model the maximum horizontal acceleration (in g units), `accel`, taken at the ith measuring station for the jth earthquake, as a function of the magnitude of the quake on the Richter scale, `richter`, and the distance (in km) of the measuring station from the quake epicenter, `distance`. We are also interested in the possible effect of the soil type `soil`, soil versus rock, at a given measuring station on acceleration. The results of this study are useful to understand the nature of the damage caused by a particular earthquake and to determine the location for sensitive installations such as nuclear facilities.

Let's consider one of the models studied by Davidian and Giltinan (1995) for these data,

$$\log_{10}(\texttt{accel}_{ij}) = \phi_{1j} - \log_{10}\sqrt{\texttt{distance}_{ij}^2 + \exp(\phi_{2j})} - \phi_{3ij}\sqrt{\texttt{distance}_{ij}^2 + \exp(\phi_{2j})} + \epsilon_{ij} \quad (8)$$

where

$$\begin{aligned}\phi_{1j} &= \beta_0 + \beta_1 \texttt{richter}_j + u_{1j}\\ \phi_{2j} &= \beta_2\\ \phi_{3i} &= \beta_3 + u_{3j}\end{aligned} \tag{9}$$

and

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{3j} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}), \text{ diagonal } \mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{u_1} & 0 \\ 0 & \sigma^2_{u_3} \end{bmatrix}, \text{ and } \epsilon_{ij} \sim N(0, \sigma^2_\epsilon) \tag{10}$$

▷ Example 6: Fitting an NLME model for the earthquake data

Let's fit the model defined by (8), (9), and (10) by using `menl`.

```
. use https://www.stata-press.com/data/r19/earthquake
(Earthquake intensity (Joyner and Boore, 1981))
. menl laccel = {phi1:}-log10(sqrt(c.distance#c.distance+exp({phi2})))
> -{phi3:}*sqrt(c.distance#c.distance+exp({phi2})),
> define(phi1: richter U1[quake]) define(phi3: U3[quake], xb)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood =  2.4115811
Iteration 2:  Linearization log likelihood =  2.4075141
Iteration 3:  Linearization log likelihood =   2.407347
Iteration 4:  Linearization log likelihood =  2.4073424
Iteration 5:  Linearization log likelihood =  2.4073412
Iteration 6:  Linearization log likelihood =  2.4073411
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        182
Group variable: quake                           Number of groups  =         23
                                                Obs per group:
                                                              min =          1
                                                              avg =        7.9
                                                              max =         38
                                                Wald chi2(1)      =      26.26
Linearization log likelihood =  2.4073411       Prob > chi2       =     0.0000
        phi1: richter U1[quake]
        phi3: U3[quake], xb
```

laccel	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
richter	.231002	.0450804	5.12	0.000	.1426461	.319358
_cons	-.8836537	.2826255	-3.13	0.002	-1.437589	-.329718
phi3						
_cons	.004575	.0014192	3.22	0.001	.0017935	.0073566
/phi2	4.063075	.4023386	10.10	0.000	3.274506	4.851644

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
quake: Independent				
var(U1)	.0056676	.0073404	.0004477	.0717519
var(U3)	.000013	8.42e-06	3.66e-06	.0000463
var(Residual)	.0461647	.0054421	.0366409	.0581639

We also store our estimates for later use:

```
. estimates store E1
```

By default, `menl` assumes that the random effects u_{1j} and u_{3j} are independent, so there is no need to specify the `covariance()` option in this case. In other words, omitting the `covariance()` option is equivalent to specifying `covariance(U1 U3, independent)`.

◁

▷ Example 7: Likelihood-ratio test for variance components

Davidian and Giltinan (1995) did not include any random effects in the model for the ϕ_{2j} parameters. Let's check whether the random effects are needed in the equations for ϕ_{1j} and ϕ_{3j} parameters in (9).

One simple way to assess whether a random effect associated with a certain ϕ_j can be omitted, is to examine its coefficient of variation (CV), the ratio of the standard deviation to the mean. Let's compute the CV for ϕ_{3j}. For convenience, let's redisplay the results from example 6 as standard deviations for variance components.

```
. menl, stddeviations
Mixed-effects ML nonlinear regression          Number of obs     =        182
Group variable: quake                          Number of groups  =         23
                                               Obs per group:
                                                             min =          1
                                                             avg =        7.9
                                                             max =         38
                                               Wald chi2(1)      =      26.26
Linearization log likelihood =  2.4073411      Prob > chi2       =     0.0000
        phi1: richter U1[quake]
        phi3: U3[quake], xb
```

laccel	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
richter	.231002	.0450804	5.12	0.000	.1426461	.319358
_cons	-.8836537	.2826255	-3.13	0.002	-1.437589	-.329718
phi3						
_cons	.004575	.0014192	3.22	0.001	.0017935	.0073566
/phi2	4.063075	.4023386	10.10	0.000	3.274506	4.851644

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
quake: Independent				
sd(U1)	.0752832	.0487517	.0211582	.2678655
sd(U3)	.0036085	.0011673	.0019142	.0068026
sd(Residual)	.2148596	.0126644	.1914181	.241172

The `stddeviations` option specifies that `menl` display random-effects and error standard deviations instead of variances. It will also display correlations instead of covariances whenever they are in the model. Because random-effects variances for these data are very small, we will use this option in all subsequent examples to display results in the standard deviation metric.

The interquake random variation in the ϕ_{3j} values about their mean is CV = `sd(U3)/{phi3:_cons}` $= 0.0036/0.0046 \approx 78\%$, and it appears reasonable to keep it in the model. You can perform a formal likelihood-ratio (LR) test of H_0: $\sigma^2_{u_3} = 0$ to verify this, as we show below for the test of H_0: $\sigma^2_{u_1} = 0$.

Let's check whether we need random intercept u_{1j} to model ϕ_{1j}. Computing CV in this case to get an initial assessment is not simple because the mean of ϕ_{1j} depends on the jth quake through variable `richter`. Given the same main equation (8), we will use the LR test to compare the restricted model, with u_{1j} excluded, which is defined by (11) and (12) below, with the full model defined by (9) and (10).

The stage 2 specification of the restricted model is

$$\begin{aligned} \phi_{1j} &= \beta_0 + \beta_1 \texttt{richter}_j \\ \phi_{2j} &= \beta_2 \\ \phi_{3ij} &= \beta_3 + u_{3j} \end{aligned} \tag{11}$$

where

$$u_{3j} \sim N(0, \sigma^2_{u_3}) \quad \text{and} \quad \epsilon_{ij} \sim N(0, \sigma^2_{\epsilon}) \tag{12}$$

We now fit the restricted model:

```
. menl laccel = {phi1:}-log10(sqrt(c.distance#c.distance+exp({phi2})))
> -{phi3:}*sqrt(c.distance#c.distance+exp({phi2})),
> define(phi1: richter, xb) define(phi3: U3[quake], xb)
> stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood =  2.1262862
Iteration 2:  Linearization log likelihood =   2.126043
Iteration 3:  Linearization log likelihood =  2.1260328
Iteration 4:  Linearization log likelihood =    2.12603
Iteration 5:  Linearization log likelihood =  2.1260297

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        182
Group variable: quake                           Number of groups  =         23
                                                Obs per group:
                                                              min =          1
                                                              avg =        7.9
                                                              max =         38
                                                Wald chi2(1)      =      32.22
Linearization log likelihood =  2.1260297       Prob > chi2       =     0.0000

        phi1: richter, xb
        phi3: U3[quake], xb
```

laccel	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
richter	.2208878	.0389144	5.68	0.000	.1446169	.2971586
_cons	-.7863293	.2503442	-3.14	0.002	-1.276995	-.2956637
phi3						
_cons	.0054348	.0015661	3.47	0.001	.0023653	.0085044
/phi2	4.228431	.3702251	11.42	0.000	3.502803	4.954059

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
quake: Identity				
sd(U3)	.0042144	.0011309	.0024907	.0071309
sd(Residual)	.2170084	.0122821	.1942231	.2424668

```
. estimates store E2
```

Next, we use `lrtest` to perform an LR test of the hypothesis:

$$H_0\colon \sigma^2_{u_1} = 0 \qquad \text{versus} \qquad H_1\colon \sigma^2_{u_1} \neq 0$$

```
. lrtest E1 E2, stats
Likelihood-ratio test
Assumption: E2 nested within E1
 LR chi2(1) =   0.56
Prob > chi2 = 0.4532
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
E2	182	.	2.12603	6	7.747941	26.97198
E1	182	.	2.407341	7	9.185318	31.61336

```
Note: BIC uses N = number of observations. See [R] IC note.
```

Because testing of H_0: $\sigma^2_{u_1} = 0$ is on the boundary of the parameter space, `lrtest` reports a note that the provided LR test is conservative; that is, the actual p-value is smaller than the one reported. For a test of H_0: $\sigma^2_{u_1} = 0$ in a two-level model, the true asymptotic distribution is not $\chi^2(1)$ but a mixture of $\chi^2(0)$ and $\chi^2(1)$ with equal weights, $0.5\chi^2(0) + 0.5\chi^2(1)$; thus the p-value is actually $0.4532/2 = 0.2266$ (see Rabe-Hesketh and Skrondal 2022, sec 8.8). We do not have sufficient evidence to reject the null hypothesis, so we can omit random effect u_{1j} from the full model. AIC and BIC also favor a simpler, reduced model.

◁

▷ Example 8: Including within-subject covariates

One of the questions of interest in the earthquake study was the potential effect of the soil type on acceleration. Variable `soil` is a within-subject covariate because the values $\texttt{soil}_{ij}$ may vary within a subject (earthquake). We include variable `soil` in the equation for ϕ_{3ij} in (11),

$$\begin{aligned}\phi_{1j} &= \beta_0 + \beta_1 \texttt{richter}_j \\ \phi_{2j} &= \beta_2 \\ \phi_{3ij} &= \beta_3 + \beta_4 \texttt{soil}_{ij} + u_{3j}\end{aligned}$$

and fit the corresponding model:

```
. menl laccel = {phi1:}-log10(sqrt(c.distance#c.distance+exp({phi2})))
> -{phi3:}*sqrt(c.distance#c.distance+exp({phi2})),
> define(phi1: richter, xb) define(phi3: i.soil U3[quake]) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood =  3.5634779
Iteration 2:  Linearization log likelihood =  3.5632472
Iteration 3:  Linearization log likelihood =  3.5632339
Iteration 4:  Linearization log likelihood =  3.5632304
Iteration 5:  Linearization log likelihood =  3.5632298

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        182
Group variable: quake                           Number of groups  =         23
                                                Obs per group:
                                                              min =          1
                                                              avg =        7.9
                                                              max =         38
                                                Wald chi2(2)      =      34.20
Linearization log likelihood =  3.5632298       Prob > chi2       =     0.0000

        phi1: richter, xb
        phi3: i.soil U3[quake]
```

laccel	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
richter	.2275944	.0395549	5.75	0.000	.1500683	.3051206
_cons	-.8079826	.2548833	-3.17	0.002	-1.307545	-.3084205
phi3						
soil						
soil	-.0011041	.0006441	-1.71	0.087	-.0023665	.0001583
_cons	.0067347	.0017416	3.87	0.000	.0033213	.0101481
/phi2	4.3212	.3653809	11.83	0.000	3.605067	5.037334

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
quake: Identity				
sd(U3)	.0043088	.0011285	.0025788	.0071992
sd(Residual)	.2147101	.0121424	.1921829	.2398779

The estimated coefficient for the soil type is -0.0011 with a 95% CI of $[-0.0024, 0.0002]$. The knowledge of the soil type at a particular site does not appear to add explanatory power to our model.

◁

Random-effects covariance structures

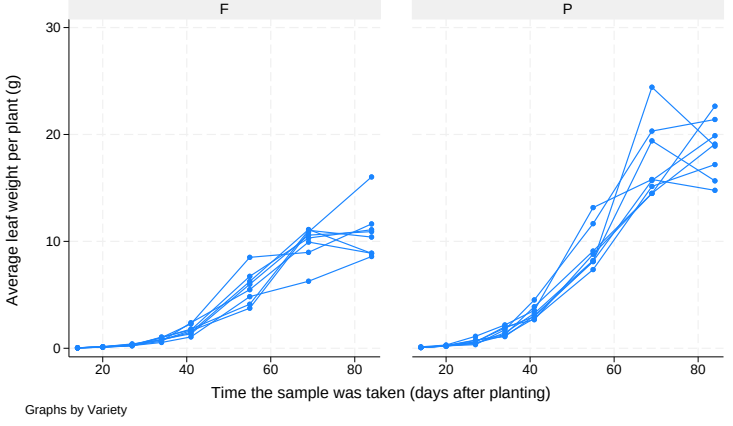
`menl` supports various covariance structures to model the random-effects covariance matrix. They are specified using the `covariance()` option. The `covariance()` option may be repeated. This is necessary to accommodate multilevel NLME models, where you may need to specify different covariance matrices for the random effects at different levels. Repeating this option may also be useful if you want to specify a block-diagonal covariance structure. See example 23 for details.

▷ Example 9: Two-level model with correlated random effects

Davidian and Giltinan (1995, sec. 1.1.3 and 11.2) discuss a study of soybean plants that started in 1988 and spanned over three growing seasons, `year`. The central objective of the study was to compare the growth patterns of two genotypes of soybean plants, `variety`: a commercial variety of soybean, denoted by F, and an experimental variety, denoted by P. In each season, eight plots were planted using F variety and eight using P variety. To assess growth, researchers sampled each plot 8 to 10 times ($8 \leq n_j \leq 10$) at approximately weekly intervals, `time`. At each sampling time, six plants were taken from each plot at random. Leaves from the plants were weighed, and the resulting total weight was divided by six to yield a measure of the average leaf weight per plant (in g) for the plot for that week, `weight`. Plots are identified by the `plot` variable.

Let's plot the data first.

```
. use https://www.stata-press.com/data/r19/soybean
(Growth of soybean plants (Davidian and Giltinan, 1995))
. twoway connected weight time if year==2, connect(L) by(variety)
```



The graph shows the average leaf weights per plant over time for the eight plots with plants of each genotype in the 1989 growing season. Longitudinal growth measures for each plot are connected with solid lines. Apart from some intraplot variation, the growth profile of each plot follows roughly an S shape, according to which growth begins slowly, then shows a linear trend during the middle of the growing season, and then "levels off" at the end. Such pattern is typical for many growth studies.

The main goal of the study was to compare growth patterns over the growing season for the two soybean genotypes. Because the three growing seasons differed markedly in terms of precipitation—1988 was unusually dry, 1989 was wet, and 1990 was normal—contrasting these growth patterns across years was also of interest. The results of this study are useful, for example, for harvesting purposes.

A popular model for individual profiles that resemble an S shape is the logistic growth model:

$$\texttt{weight}_{ij} = \frac{\phi_{1j}}{1 + \exp\left\{-\left(\texttt{time}_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij} \tag{13}$$

ϕ_{1j} is the asymptotic average leaf weight per soybean plant in plot j as $\texttt{time}_{ij} \to \infty$. ϕ_{2j} is the time at which half of ϕ_{1j} is reached; that is, if $\texttt{time}_{ij} = \phi_{2j}$, then $E(\texttt{weight}_{ij}) = 0.5\phi_{1j}$. ϕ_{1j} and ϕ_{2j} will henceforth be referred to as "the limiting growth" and "half-life", respectively. ϕ_{3j} is a scale parameter,

and it represents the number of days it takes for average leaf weight to grow from 50% (half-life) to about 73% of its limiting growth. That is, if we set $\texttt{time}_{ij} = t_{0.73} = \phi_{2j} + \phi_{3j}$, the right-hand side of (13), ignoring the error term, reduces to $\phi_{1j}/\{1 + \exp(-1)\} = 0.73\phi_{1j}$, and then $\phi_{3j} = t_{0.73} - \phi_{2j}$.

We will start with a simple stage 2 specification that does not contain any covariates. Also, because the number of soybean plots, 48, is large compared with the number of random effects, 3, we consider a general positive-definite, unstructured, random-effects covariance matrix:

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \tag{14}$$

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}), \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} \end{bmatrix}, \epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$$

To specify this covariance structure in `menl`, we specify `unstructured` in the `covariance()` option. The `covariance()` option also requires that we list the names of random effects to be correlated.

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb) define(phi3: U3[plot], xb)
> covariance(U1 U2 U3, unstructured)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -739.90142
Iteration 2:  Linearization log likelihood = -739.84929
  (iteration log omitted)
Iteration 39: Linearization log likelihood = -739.83452
Iteration 40: Linearization log likelihood = -739.83445

Computing standard errors:

Mixed-effects ML nonlinear regression          Number of obs     =        412
Group variable: plot                           Number of groups  =         48
                                               Obs per group:
                                                             min =          8
                                                             avg =        8.6
                                                             max =         10

Linearization log likelihood = -739.83445

        phi1: U1[plot], xb
        phi2: U2[plot], xb
        phi3: U3[plot], xb
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	19.25314	.8031811	23.97	0.000	17.67893	20.82734
phi2						
_cons	55.01999	.7272491	75.65	0.000	53.59461	56.44537
phi3						
_cons	8.403468	.3152551	26.66	0.000	7.78558	9.021357

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
plot: Unstructured				
var(U1)	27.05081	6.776514	16.55561	44.19929
var(U2)	17.61605	5.317897	9.748768	31.83227
var(U3)	1.972036	.9849817	.7409027	5.2489
cov(U1,U2)	15.73304	5.413362	5.123046	26.34304
cov(U1,U3)	5.193819	2.165585	.9493497	9.438288
cov(U2,U3)	5.649306	2.049458	1.632442	9.66617
var(Residual)	1.262237	.1111685	1.062119	1.500059

The expected limiting growth or expected maximum average weight, $\beta_1 = E\left(\phi_{1j}\right)$, of soybean leaves is estimated to be around 19.25 grams. The expected half-life or the time at which the leaves reach half of their expected maximum average weight, $\beta_2 = E\left(\phi_{2j}\right)$, is estimated to be around 55 days after planting. The expected time needed for the average leaf weight per plant to grow from 50% to 73% of the limiting growth, $\beta_3 = E\left(\phi_{3j}\right)$, is about 8.4 days.

The estimates of the six random-effects variance–covariance parameters σ_{11}, σ_{22}, σ_{33}, σ_{12}, σ_{13}, and σ_{23} are displayed in the upper part of the random-effects parameters table. There is a plot-to-plot variation in the estimates of all three parameters of interest: β_1, β_2, and β_3. Also, the plot-specific effects associated with the parameters of interest are positively correlated. For example, based on the estimate of 5.19 of `cov(U1,U3)`, plants with larger maximum weights tend to grow faster.

We store our estimates for later use:

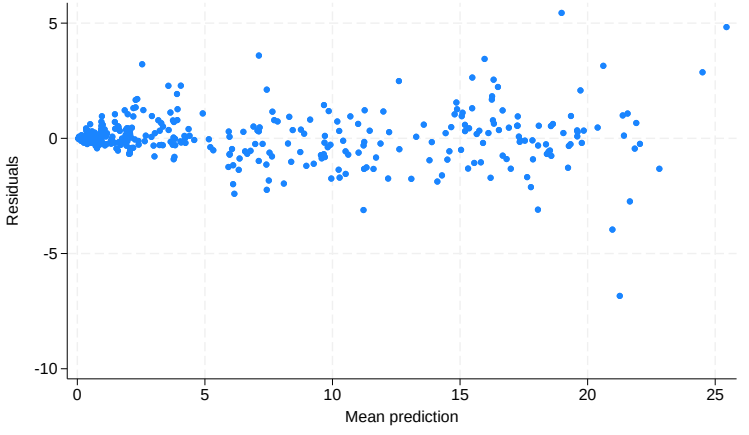
```
. estimates store S1
```

◁

▷ Example 10: Residuals-vs-fitted plot to check for heteroskedasticity

A popular tool for investigating within-cluster heteroskedasticity is the plot of residuals against the predicted values and other candidate variance covariates. For growth models, variance is often a function of the mean (predicted values). Below we construct the plot of residuals versus predicted values to evaluate the assumption of homoskedastic errors in example 9.

```
. predict fitweight, yhat
. predict res, residuals
. scatter res fitweight
```



The plot reveals increasing variability with the predicted average leaf weights, which indicates that our within-cluster variance model is misspecified. In *Heteroskedastic within-group errors*, we will show how to account for within-cluster heteroskedasticity by using the `resvariance()` option.

Heteroskedastic within-group errors

Until now, we assumed that the within-group errors—the ϵ's in the considered models—are i.i.d. Gaussian with common variance σ^2_ϵ, labeled as `var(Residual)` by `menl` in the output.

To relax the assumptions of homoskedasticity and the independence of errors, `menl` provides two alternatives. You can model the within-group error variance–covariance matrix, $\sigma^2\mathbf{\Lambda}_j$, directly by using the `rescovariance()` option. If you used the `mixed` command and its `residuals()` option before, you should be familiar with this approach. Alternatively, you can model the error variance–covariance matrix indirectly by modeling the heteroskedasticity structure with the `resvariance()` option and the correlation structure with the `rescorrelation()` option; see *Variance-components parameters*. The latter approach offers more flexibility, particularly in modeling the heteroskedasticity structure. For example, many NLME models exhibit within-subject heterogeneity that is a power function of the mean. The `rescovariance()` option cannot model this, but `resvariance(power _yhat)` can.

If your error structure is simple and is similar to those encountered in `mixed`, you can use the `rescovariance()` option. Otherwise, use `resvariance()`, `rescorrelation()`, or both to model more flexible within-group error covariance structures.

Example 11: Heteroskedastic power structure

Continuing with example 9, for these types of growth data, we find it is common for the intraplot variance to increase systematically with the average leaf weight, as we saw in example 10 from the residuals-versus-fitted plot. Davidian and Giltinan (1995) proposed a variance structure that models the within-group error variance as a power function of the mean to account for the intraplot variability. To reduce the number of parameters to be estimated, the authors assume that the random effects are independent.

Stage 2 specification of the model defined by (13) becomes

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \tag{15}$$

where

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \\ u_{3j} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}), \text{ diagonal } \boldsymbol{\Sigma} = \begin{bmatrix} \sigma^2_{u_1} & 0 & 0 \\ 0 & \sigma^2_{u_2} & 0 \\ 0 & 0 & \sigma^2_{u_3} \end{bmatrix}$$

and

$$\text{Var}\left(\epsilon_{ij}\right) = \sigma^2(\widehat{\texttt{weight}}_{ij})^{2\delta}$$

Parameter σ^2 in the above is no longer an overall error variance σ^2_ϵ but a common multiplier or a (squared) scale parameter.

In `menl`, this type of heteroskedasticity is modeled by specifying `resvariance(power _yhat, noconstant)`. `_yhat` designates that the variance should be modeled as a function of predicted values, $\widehat{\texttt{weight}}_{ij}$. By default, variance function `power` includes a constant, which we suppress by specifying the `noconstant` option.

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb) define(phi3: U3[plot], xb)
> resvariance(power _yhat, noconstant)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -364.02249
Iteration 2:  Linearization log likelihood = -364.22838
Iteration 3:  Linearization log likelihood = -364.43168
Iteration 4:  Linearization log likelihood = -364.38319
Iteration 5:  Linearization log likelihood = -364.38964
Iteration 6:  Linearization log likelihood = -364.38915
Iteration 7:  Linearization log likelihood =  -364.3892

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48

                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10
```

```
Linearization log likelihood =  -364.3892
        phi1: U1[plot], xb
        phi2: U2[plot], xb
        phi3: U3[plot], xb
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	16.82289	.6030531	27.90	0.000	15.64093	18.00485
phi2						
_cons	51.74669	.4579632	112.99	0.000	50.8491	52.64429
phi3						
_cons	7.545371	.0856321	88.11	0.000	7.377535	7.713206

Random-effects parameters		Estimate	Std. err.	[95% conf.	interval]
plot: Independent					
	var(U1)	11.32134	2.831139	6.934849	18.48241
	var(U2)	2.68911	.9344022	1.360932	5.313504
	var(U3)	1.48e-11	1.34e-07	0	.
Residual variance: Power _yhat					
	sigma2	.0509223	.004422	.0429527	.0603706
	delta	.9339856	.0244477	.886069	.9819023

The near-zero estimate of the variance component of u_{3j}, `var(U3)`, suggests that the random-effects model is overparameterized. The within-group heteroskedasticity structure appears to explain enough variability in our data, and we no longer need random effects specific to ϕ_{3j}. This is quite common in mixed-effects models: the random-effects covariance structure and the within-group error covariance structure compete with each other, in the sense that fewer random effects are needed when the within-group error covariance structure is present, and vice versa.

Let's omit u_{3j} from (15) but now assume an unstructured covariance matrix for u_{1j} and u_{2j}. The EM algorithm used by `menl` to obtain initial values produces the starting values for variance components that are, in general, close to the final estimates upon convergence. Thus it can be used as a tool to help us detect potential convergence problems because of an overparameterized random-effects structure at an earlier stage. For example, we can check whether an unstructured covariance matrix is a reasonable choice for the random effects u_{1j} and u_{2j} for these data by displaying estimates after a few iterations. This can be done by specifying the `iterate(#)` option, where # is a small number of iterations, say, between 1 and 4. Below we specify `iterate(3)` to perform only three iterations and the `stddeviations` option to obtain standard deviations and correlations instead of variances and covariances for easier interpretability:

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb)
> covariance(U*, unstructured) resvariance(power _yhat, noconstant)
> iterate(3) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -379.66343
Iteration 2:  Linearization log likelihood = -362.90921
Iteration 3:  Linearization log likelihood = -361.92284

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48

                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10

Linearization log likelihood = -361.93956

        phi1: U1[plot], xb
        phi2: U2[plot], xb
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	16.92772	.567749	29.82	0.000	15.81495	18.04049
phi2						
_cons	51.81715	.4484621	115.54	0.000	50.93818	52.69612
/phi3	7.54089	.0869112	86.77	0.000	7.370547	7.711233

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
plot: Unstructured				
sd(U1)	2.904731	.4070473	2.207115	3.822848
sd(U2)	1.28232	.2555136	.8677347	1.894985
corr(U1,U2)	-.99999	.003294	-1	1
Residual variance:				
Power _yhat				
sigma	.2255165	.0095107	.2076254	.2449493
delta	.955248	.0230662	.910039	1.000457

```
Warning: Convergence not achieved.
```

The `U*` in `covariance(U*, unstructured)` is a shorthand notation to reference all random effects starting with `U`, that is, `U1` and `U2` in this example. The correlation between u_{1j} and u_{2j} is near -1 with a 95% CI of $[-1, 1]$, which indicates that the random-effects model may still be overparameterized. If you try to fit this model without the `iteration(3)` option, it would keep iterating without convergence.

Therefore, we further simplify the random-effects covariance structure by assuming independence between u_{1j} and u_{2j}. Stage 2 specification of the model defined by (13) is now

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 + u_{1j} \\ \beta_2 + u_{2j} \\ \beta_3 \end{bmatrix} \tag{16}$$

where

$$\mathbf{u}_j = \begin{bmatrix} u_{1j} \\ u_{2j} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}), \text{ diagonal } \mathbf{\Sigma} = \begin{bmatrix} \sigma^2_{u_1} & 0 \\ 0 & \sigma^2_{u_2} \end{bmatrix}$$

and

$$\text{Var}\left(\epsilon_{ij}\right) = \sigma^2(\widehat{\texttt{weight}}_{ij})^{2\delta}$$

We fit this model and store its results as S2:

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3})),
> define(phi1: U1[plot], xb) define(phi2: U2[plot], xb)
> resvariance(power _yhat, noconstant)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -402.76182
Iteration 2:  Linearization log likelihood =  -372.4091
Iteration 3:  Linearization log likelihood = -363.83194
Iteration 4:  Linearization log likelihood = -364.37747
Iteration 5:  Linearization log likelihood = -364.38661
Iteration 6:  Linearization log likelihood = -364.38917
Iteration 7:  Linearization log likelihood = -364.38918

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10

Linearization log likelihood = -364.38918
        phi1: U1[plot], xb
        phi2: U2[plot], xb
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	16.8229	.6030458	27.90	0.000	15.64095	18.00485
phi2						
_cons	51.74669	.4579586	112.99	0.000	50.84911	52.64427
/phi3	7.545367	.0856312	88.11	0.000	7.377533	7.713202

Random-effects parameters		Estimate	Std. err.	[95% conf.	interval]
plot: Independent					
	var(U1)	11.32133	2.831137	6.934841	18.4824
	var(U2)	2.689113	.9344039	1.360932	5.313512
Residual variance:					
Power _yhat					
	sigma2	.0509223	.004422	.0429527	.0603706
	delta	.9339853	.0244477	.8860686	.9819019

```
. estimates store S2
```

Because (16) is not nested in (14), we assess the adequacy of the heteroskedastic model by using information criteria. We use `estimates stats` to display the AIC and BIC values for the three models.

```
. estimates stats S1 S2
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
S1	412	.	-739.8344	10	1499.669	1539.879
S2	412	.	-364.3892	7	742.7784	770.9255

```
Note: BIC uses N = number of observations. See [R] IC note.
```

The heteroskedastic model defined by (16) has smaller AIC and BIC values and thus provides a much better representation of the data than (14).

◁

▷ Example 12: Heteroskedastic model with interactions

The main goal of the soybean study was to compare growth patterns of the two genotypes of soybean over the three growing seasons, represented by calendar years 1988 through 1990. More specifically, we would like to compare the limiting growth, the half-life, and the growth rate of soybeans across growing seasons and genotypes.

Let $P_j = I\left(\texttt{variety}_j = \texttt{P}\right)$ be the indicator for genotype variety P, $S_{89,j} = I\left(\texttt{year}_j = 1989\right)$ be the indicator for growing season 1989, and $S_{90,j} = I\left(\texttt{year}_j = 1990\right)$ be the indicator for growing season 1990. Genotype variety F and growing season 1988 are baselines.

Consider an extension of the model defined by (13) and (16), where, in addition to random effects, ϕ_{1j} includes main and interaction effects of growing seasons and genotype variety, ϕ_{2j} includes main effects of growing seasons and genotype variety, and ϕ_{3j} contains main effects of growing seasons only.

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\end{bmatrix} = \begin{bmatrix}\beta_{11} + \beta_{12}S_{89,j} + \beta_{13}S_{90,j} + \beta_{14}P_j + \beta_{15}S_{89,j}\times P_j + \beta_{16}S_{90,j}\times P_j + u_{1j}\\ \beta_{21} + \beta_{22}S_{89,j} + \beta_{23}S_{90,j} + \beta_{24}P_j + u_{2j}\\ \beta_{31} + \beta_{32}S_{89,j} + \beta_{33}S_{90,j}\end{bmatrix} \tag{17}$$

To fit the model defined by (13) and (17) by using `menl`, we extend `menl`'s specification from example 11 by including the full-factorial interaction `i.year##i.variety` in the expression `{phi1:}`, main effects `i.year` and `i.variety` in the expression `{phi2:}`, and main effects `i.year` in the expression `{phi3:}`.

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: i.year##i.variety U1[plot])
> define(phi2: i.year i.variety U2[plot])
> define(phi3: i.year, xb) resvariance(power _yhat, noconstant)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -292.62615
Iteration 2:  Linearization log likelihood = -290.24389
  (iteration log omitted)
Iteration 10: Linearization log likelihood = -290.90729
Iteration 11: Linearization log likelihood =  -290.9073
```

```
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10
                                                Wald chi2(10)     =     413.88
Linearization log likelihood =  -290.9073       Prob > chi2       =     0.0000
       phi1: i.year i.variety i.year#i.variety U1[plot]
       phi2: i.year i.variety U2[plot]
       phi3: i.year
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
year						
1989	-8.837933	1.056113	-8.37	0.000	-10.90788	-6.76799
1990	-3.666206	1.165969	-3.14	0.002	-5.951463	-1.38095
variety						
P	1.648139	1.033433	1.59	0.111	-.3773532	3.673631
year#variety						
1989#P	5.563008	1.167782	4.76	0.000	3.274196	7.851819
1990#P	.0974815	1.178054	0.08	0.934	-2.211462	2.406425
_cons	19.42734	.9445749	20.57	0.000	17.57601	21.27867
phi2						
year						
1989	-2.253227	.9746495	-2.31	0.021	-4.163505	-.3429494
1990	-4.970736	.9778317	-5.08	0.000	-6.887251	-3.054221
variety						
P	-1.294058	.4255317	-3.04	0.002	-2.128085	-.4600314
_cons	54.81257	.7587239	72.24	0.000	53.3255	56.29964
phi3						
year						
1989	-.9023768	.1992358	-4.53	0.000	-1.292872	-.5118818
1990	-.6805314	.2100799	-3.24	0.001	-1.09228	-.2687823
_cons	8.060677	.1459662	55.22	0.000	7.774588	8.346765

Random-effects parameters		Estimate	Std. err.	[95% conf.	interval]
plot: Independent					
	var(U1)	.8643052	.5271131	.2615445	2.8562
	var(U2)	.1341755	.230691	.0046151	3.900891
Residual variance:					
Power _yhat					
	sigma2	.0467091	.0039176	.0396286	.0550546
	delta	.9451193	.0227608	.9005089	.9897297

```
. estimates store S3
```

By including more fixed effects in the model, which explain some of the variability in the average leaf weight, we substantially reduced the estimates of variance components. Compared with example 11, `var(U1)` decreased from 11.32 to 0.86, and `var(U2)` decreased from 2.69 to 0.13. It often happens that specifying a better-fitting model for the fixed effects reduces the need for random effects in the model.

We can compare model `S3` or the model defined by (17) with model `S2` or the one defined by (16) by using, for example, information criteria.

```
. estimates stats S2 S3
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
S2	412	.	-364.3892	7	742.7784	770.9255
S3	412	.	-290.9073	17	615.8146	684.172

Note: BIC uses N = number of observations. See **[R] IC note**.

Even though `S3` has many more parameters, it fits the soybean data better than `S2`.

By inspecting the fixed-effects estimates from the output of model `S3`, we see that both the type of year and genotype variety affect all three parameters: the expected maximum leaf weight, half-life, and scale. For example, all three parameters achieve their highest values in the dry year, baseline year 1988, because coefficient estimates for the other years are negative. Also, the genotype variety F reaches its half-life roughly a day later ($\beta_{24} = -1.29$) than genotype variety P.

◁

▷ Example 13: Obtaining predictions

After estimation, we may want to obtain predicted values for the outcome or for the parameters of interest. Continuing with example 12, we want to predict the asymptotic average leaf weight per soybean plant in each plot, $\widehat{\phi_{1j}}$. The ϕ_{1j} parameter is not constant but varies for each plot, growing season, and genotype variety. We can use `predict` after `menl` to obtain predicted values for ϕ_{1j}; see [ME] **menl postestimation**.

First, we create a new grouping variable for growing seasons, genotype variety, and plot types. We also create the `tolist` variable to mark the first observation in each group.

```
. egen group = group(year variety plot)
. by group, sort: generate byte tolist=(_n==1)
```

Next, we use `predict` to compute predicted values for the expression `{phi1:}` and store them in the new variable `phi1`. We store only unique values in `phi1`, one for each `group`; the remaining observations are replaced with missing values.

```
. predict double (phi1 = {phi1:})
. quietly replace phi1 = . if tolist!=1
```

We now list the five smallest and the five largest values of the asymptotic average leaf weight.

```
. sort phi1
. list plot year variety phi1 if (_n<=5 | _n>43) & phi1<., sep(5)
```

	plot	year	variety	phi1
1.	1989F6	1989	F	8.8421451
2.	1989F4	1989	F	10.449521
3.	1989F5	1989	F	10.473849
4.	1989F1	1989	F	10.721364
5.	1989F7	1989	F	10.810197
44.	1988P8	1988	P	20.86739
45.	1988P2	1988	P	21.237692
46.	1988P4	1988	P	21.310512
47.	1988P3	1988	P	21.506007
48.	1988P6	1988	P	21.581873

Soybean plants with genotype variety P have substantially larger asymptotic average leaf weight in the dry year, 1988, than soybean plants with genotype variety F in the wet year, 1989.

◁

▷ Example 14: Within-group error correlation structure

Pinheiro and Bates (2000, chap. 8) analyzed data from a study of the estrus cycles of mares. Originally analyzed in Pierson and Ginther (1987), the data contain daily records of the number of ovarian follicles larger than 10 mm over a period ranging from 3 days before ovulation to 3 days after the subsequent ovulation. The measurement times for each mare are scaled so that the ovulations for each mare occur at times 0 and 1 and are recorded in `stime`.

The considered model is

$$\texttt{follicles}_{ij} = \phi_{1j} + \phi_{2j}\sin\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \phi_{4j}\cos\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

where ϕ_{1j} is an intercept, ϕ_{3j} is the frequency of the sine wave for the jth mare, and ϕ_{2j} and ϕ_{4j} are terms determining the amplitude and phase of the sine wave for the jth mare. If a_j and p_j are the amplitude and phase for mare j, then $\phi_{2j} = a_j\cos(p_j)$ and $\phi_{4j} = a_j\sin(p_j)$.

This model was fit in example 8 of [ME] **mixed** in the context of a linear mixed-effects model, where the number of ovarian follicles was a periodic function of time with known frequency ϕ_{3j} equal to 1. If we want to estimate frequency, we cannot use the `mixed` command, because ϕ_{3j} enters the model nonlinearly.

Pinheiro and Bates (2000) suggested an AR(1) correlation structure for modeling the within-group error correlation. This structure can be specified by using the `rescorrelation()` option as `rescorrelation(ar 1, t(time))`, where `time` is an integer-valued time variable used to order the observations within mares and to determine the lags between successive observations.

We also considered several random-effects structures and found that we need only one random intercept to model ϕ_{1j}.

The full specification for the stage 2 model is

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \\ \phi_{4j} \end{bmatrix} = \begin{bmatrix} \beta_1 + u_{1j} \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

where

$$\mathbf{u}_j = u_{1j} \sim N(0, \sigma_u^2)\,,\ \boldsymbol{\epsilon}_j \sim N(\mathbf{0},\, \sigma_\epsilon^2 \boldsymbol{\Lambda}_j)$$

and

$$\sigma_\epsilon^2 \boldsymbol{\Lambda}_j = \sigma_\epsilon^2 \begin{bmatrix} 1 & \rho & \rho^2 & \dots & \rho^{n_j-1} \\ \rho & 1 & \rho & \dots & \rho^{n_j-2} \\ \rho^2 & \rho & 1 & \dots & \rho^{n_j-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n_j-1} & \rho^{n_j-2} & \rho^{n_j-3} & \dots & 1 \end{bmatrix}$$

We fit this model by using `menl` as follows:

```
. use https://www.stata-press.com/data/r19/ovary, clear
(Ovarian follicles in mares)
. menl follicles = {phi1: U1[mare], xb} + {phi2}*sin(2*_pi*stime*{phi3}) +
> {phi4}*cos(2*_pi*stime*{phi3}), rescorrelation(ar 1, t(time))
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -789.43415
Iteration 2:  Linearization log likelihood = -789.43439
Iteration 3:  Linearization log likelihood = -789.43439
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
Linearization log likelihood = -789.43439
        phi1: U1[mare], xb
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	11.98929	.9055946	13.24	0.000	10.21436	13.76422
/phi2	.2226033	.3290159	0.68	0.499	-.4222559	.8674626
/phi3	4.18747	.2746499	15.25	0.000	3.649166	4.725774
/phi4	.279653	.3223277	0.87	0.386	-.3520977	.9114036

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
mare: Identity				
var(U1)	4.935352	3.967836	1.020903	23.85899
Residual: AR(1), time time				
var(e)	20.14587	3.492937	14.34177	28.29888
rho	.7332304	.0463231	.6287332	.8117157

By using estimates of ϕ_{2j} and ϕ_{4j}, we can compute the amplitude and phase for the sine wave for mare j. The amplitude and the phase are the same for all the mares because ϕ_{2j} and ϕ_{4j} are constant and not mare specific.

For example, the amplitude a_j can be computed as $\sqrt{\phi_{2j}^2 + \phi_{4j}^2}$ by using the relationship $\phi_{2j}^2 + \phi_{4j}^2 = a_j^2\left\{\sin^2(p_j) + \cos^2(p_j)\right\} = a_j^2$. The phase p_j can be computed as $p_j = \text{atan}(\phi_{4j}/\phi_{2j})$ by using the relationship $\phi_{4j}/\phi_{2j} = \left\{a_j \sin(p_j)\right\} / \left\{a_j \cos(p_j)\right\} = \tan(p_j)$.

We can use `nlcom` to compute the amplitude and the phase.

```
. nlcom (amplitude: sqrt(_b[/phi2]^2 + _b[/phi4]^2))
> (phase: atan(_b[/phi4]/_b[/phi2]))
   amplitude: sqrt(_b[/phi2]^2 + _b[/phi4]^2)
       phase: atan(_b[/phi4]/_b[/phi2])
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
amplitude	.3574325	.2451183	1.46	0.145	-.1229904	.8378555
phase	.8985001	1.090985	0.82	0.410	-1.23979	3.03679

As we mentioned in example 1, it is important to try different initial values when fitting NLME models to investigate potential convergence to a local maximum, especially for models containing periodic functions, as in our example. We explore different initial values for this model in *Linearization approach to finding initial values* by considering the functional form of the mean function and arrive at a different solution with a larger log likelihood.

◁

Restricted maximum likelihood

Like `mixed`, `menl` provides estimation by using ML or REML. The difference between the two approaches is described in detail in *Likelihood versus restricted likelihood* in [ME] **mixed**. Briefly, REML is preferable when you have a small number of groups because it produces unbiased, at least for balanced data, estimates of variance components. In large samples, there is little difference between ML and REML. One disadvantage of REML, however, is that LR tests based on REML are inappropriate for comparing models with different fixed-effects specifications. See example 15 for an example of REML estimation.

Pharmacokinetic modeling

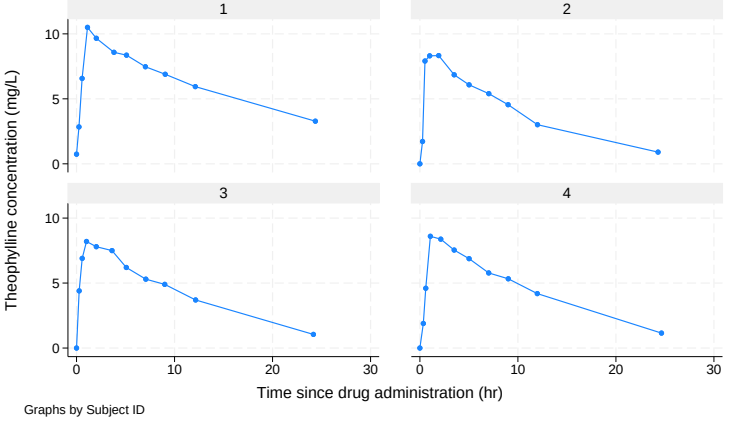
Pharmacokinetics (PKs) is the study of drug absorption, distribution, metabolism, and excretion. It is often referred to as the study of "what the body does with a drug". The goal of PK modeling is to summarize the concentration-time measurements using a model that relates drug input to drug response, to relate the parameters of this model to patient characteristics, and to provide individual dose–response predictions to optimize individual doses. In other words, by understanding between-subject variation in drug disposition, we can individualize the dosage regimen for a particular patient based on relevant physiological information identified by our PK model.

Single-dose pharmacokinetic modeling

▷ Example 15: Single-oral-dose model

Consider a PK study of the antiasthmatic agent theophylline that was reported by Boeckmann, Sheiner, and Beal (2011) and analyzed by Davidian and Giltinan (1995). The drug was administered orally to 12 subjects, where dosage `dose` (mg/kg) was given on a per weight basis. Serum concentrations (in mg/L) were obtained at 11 time points per subject over 25 hours following administration. The graph below shows the resulting concentration-time profiles for four subjects.

```
. use https://www.stata-press.com/data/r19/theoph
(Theophylline kinetics (Boeckmann et al., [1994] 2011))
. twoway connected conc time if subject<=4, connect(L) by(subject)
```



In PKs, the pattern of rapid rise to a peak concentration followed by an apparent exponential decay may be described by a so-called one-compartment open model with first-order absorption and elimination. The model corresponds roughly to viewing the body as one "blood compartment" and is particularly useful for the PK analysis of drugs that distribute relatively rapidly throughout the body, which makes it a reasonable model for the kinetics of theophylline after oral administration. Further details about compartmental modeling may be found in Gibaldi and Perrier (1982). The one-compartment open model for theophylline kinetics may be expressed as

$$\texttt{conc}_{ij} = \frac{\texttt{dose}_j k_{e_j} k_{a_j}}{\text{Cl}_j \left(k_{a_j} - k_{e_j}\right)} \left\{ \exp\left(-k_{e_j}\texttt{time}_{ij}\right) - \exp\left(-k_{a_j}\texttt{time}_{ij}\right)\right\} + \epsilon_{ij} \tag{18}$$

for $i = 1, \ldots, 11$ and $j = 1, \ldots, 12$. Model parameters are the elimination rate constant k_{e_j}, the absorption rate constant k_{a_j}, and the clearance Cl_j for each subject j.

Because each of the model parameters must be positive to be meaningful, we write

$$\begin{aligned} \text{Cl}_j &= \exp\left(\beta_0 + u_{0j}\right) \\ k_{a_j} &= \exp\left(\beta_1 + u_{1j}\right) \\ k_{e_j} &= \exp\left(\beta_2\right) \end{aligned}$$

where u_{0j} and u_{1j} are assumed independent and normally distributed with means zero and variance $\sigma^2_{u_0}$ and $\sigma^2_{u_1}$, respectively.

The model defined by (18) implies that the predicted value for the concentration at time $\texttt{time}_{ij} = 0$ is $\widehat{\texttt{conc}}_{ij} = 0$. Therefore, a power variance function, a natural candidate for this type of heteroskedastic pattern, cannot be used in this example because error variance will be 0 at $\texttt{time}_{ij} = 0$. So the constant plus power variance function, which adds a constant to the power term, is used instead to model the within-group error variance:

$$\text{Var}\left(\epsilon_{ij}\right) = \sigma^2\{(\widehat{\texttt{conc}}_{ij})^\delta + c\}^2$$

In menl, we use the resvariance(power _yhat) option to specify the constant plus power variance function and the following model specification:

```
. menl conc = (dose*{ke:}*{ka:}/({cl:}*({ka:}-{ke:})))*
> (exp(-{ke:}*time)-exp(-{ka:}*time)), define(cl: exp({b0}+{U0[subject]}))
> define(ka: exp({b1}+{U1[subject]})) define(ke: exp({b2}))
> resvariance(power _yhat)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -167.51953
Iteration 2:  Linearization log likelihood = -167.65729
  (iteration log omitted)
Iteration 26: Linearization log likelihood = -167.67966
Iteration 27: Linearization log likelihood = -167.67964

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        132
Group variable: subject                         Number of groups  =         12

                                                Obs per group:
                                                              min =         11
                                                              avg =       11.0
                                                              max =         11

Linearization log likelihood = -167.67964

         cl: exp({b0}+{U0[subject]})
         ka: exp({b1}+{U1[subject]})
         ke: exp({b2})
```

conc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b0	-3.227479	.0598389	-53.94	0.000	-3.344761	-3.110197
/b1	.432931	.1980835	2.19	0.029	.0446945	.8211674
/b2	-2.453742	.0514567	-47.69	0.000	-2.554595	-2.352889

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Independent				
var(U0)	.0288787	.0127763	.0121337	.0687323
var(U1)	.4075667	.1948712	.1596655	1.040367
Residual variance:				
Power _yhat				
sigma2	.0976905	.0833025	.0183661	.5196222
delta	.3187133	.2469503	-.1653005	.8027271
_cons	.7288982	.3822949	.2607509	2.037548

The number of groups, 12, is fairly small in these data, so we now refit the model by using REML estimation.

```
. menl conc = (dose*{ke:}*{ka:}/({cl:}*({ka:}-{ke:})))*
> (exp(-{ke:}*time)-exp(-{ka:}*time)), define(cl: exp({b0}+{U0[subject]}))
> define(ka: exp({b1}+{U1[subject]})) define(ke: exp({b2}))
> resvariance(power _yhat) reml

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log restricted-likelihood = -172.31734
Iteration 2:  Linearization log restricted-likelihood = -172.42325
  (iteration log omitted)
Iteration 23: Linearization log restricted-likelihood = -172.44383
Iteration 24: Linearization log restricted-likelihood = -172.44384

Computing standard errors:

Mixed-effects REML nonlinear regression         Number of obs     =        132
Group variable: subject                         Number of groups  =         12
                                                Obs per group:
                                                              min =         11
                                                              avg =       11.0
                                                              max =         11

Linear. log restricted-likelihood = -172.44384

          cl: exp({b0}+{U0[subject]})
          ka: exp({b1}+{U1[subject]})
          ke: exp({b2})
```

conc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/b0	-3.227295	.0619113	-52.13	0.000	-3.348639	-3.105951
/b1	.4354519	.2072387	2.10	0.036	.0292716	.8416322
/b2	-2.453743	.0517991	-47.37	0.000	-2.555267	-2.352218

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Independent				
var(U0)	.0316416	.014531	.0128634	.0778326
var(U1)	.4500585	.2228203	.1705478	1.187659
Residual variance:				
Power _yhat				
sigma2	.1015759	.0865354	.0191261	.5394529
delta	.3106636	.2466553	-.1727719	.7940991
_cons	.7150935	.3745276	.2561823	1.996073

As expected, the estimates of the random-effects variances are slightly larger than the corresponding ML estimates, but we arrive at similar inferential conclusions based on our REML estimates.

◁

▷ Example 16: Nonlinear functions of parameters

A distinctive feature of example 15 is that parameters of interest are nonlinear functions of the estimated parameters and random effects. To interpret parameters that depend on random effects, we can either integrate random effects out of the parameter expression or condition on them. The former parameter estimates are often referred to as population-based estimates. The latter parameter estimates are

referred to as conditional estimates and, when conditioning on zero random effects, $\mathbf{u}_j = 0$, as estimates for an "average" or typical subject. For linear functions, the population-based estimates coincide with the conditional estimates. This is no longer true for nonlinear functions.

In PK modeling, the parameters of interest are clearance, elimination rate, and absorption rate. These are nonlinear functions of the estimated parameters β_0, β_1, β_2, and subject-specific random effects. Depending on the context, we may be interested in their population-based estimates or in their conditional estimates.

In general, obtaining population-based estimates would require numerical integration to integrate the subject-specific random effects out of the expression. In our example, we can compute population-based estimates directly by using the fact that $\exp(u_{0j})$'s and $\exp(u_{1j})$'s are lognormally distributed.

Thus the population-based clearance Cl^P can be computed as $E\left(\mathrm{Cl}_j\right) = E\left\{\exp\left(\beta_0 + u_{0j}\right)\right\} = \exp\left(\beta_0 + \sigma^2_{u_0}/2\right)$ and the population-based absorption rate k_a^P as $E\left\{\exp\left(\beta_1 + u_{1j}\right)\right\} = \exp\left(\beta_1 + \sigma^2_{u_1}/2\right)$. The elimination rate k_e does not depend on subject-specific effects and can thus be computed simply as $k_e^P = k_e = \exp\left(\beta_2\right)$.

Alternatively, if we want parameters to represent a typical subject, we can simply set $u_{0j} = 0$ and $u_{1j} = 0$ in their expressions. Thus we can compute clearance and absorption rate for a typical subject simply as $\mathrm{Cl} = \exp\left(\beta_0\right)$ and $k_a = \exp\left(\beta_1\right)$. These formulas can also be viewed as a result of exponentiating population-based log-clearance and log-absorption rate; that is, $\mathrm{Cl} = \exp\left[E\left\{\log(\mathrm{Cl}_j)\right\}\right] = \exp\left(\beta_0\right)$ and $k_a = \exp\left[E\left\{\log(k_{a_j})\right\}\right] = \exp\left(\beta_1\right)$.

If we compare the formulas for, say, Cl^P and Cl, the former considers variation in clearances across subjects, whereas the latter ignores such variation and instead reflects what the clearance would be for a typical subject with $u_{0j} = 0$.

Both approaches have merit, and here we will compute, for example, $\mathrm{Cl}^P = \exp(\hat{\beta}_0 + \hat{\sigma}^2_{u_0}/2) = \exp(-3.23 + 0.032/2) = 0.04$. That is, 0.04 liters of serum concentration are cleared of the theophylline drug per hour per kg body weight in the considered population. In other words, for the population of subjects that weigh 75 kg, an average of $75 \times 0.04 \approx 3$ liters of serum concentration are cleared of theophylline every hour.

We can also use `nlcom` to compute the estimates of Cl^P and Cl. To use `nlcom`, we need to know how parameters are labeled by `menl` for postestimation. We can use `menl`'s option `coeflegend` to display parameter names. We also specify `noheader` to suppress the table header.

```
. menl, coeflegend noheader
```

conc	Coefficient	Legend
/b0	-3.227295	_b[/b0]
/b1	.4354519	_b[/b1]
/b2	-2.453743	_b[/b2]
/subject		
lnsd(U0)	-1.726641	_b[/subject:lnsd(U0)]
lnsd(U1)	-.3991888	_b[/subject:lnsd(U1)]
/Residual		
lnsigma	-1.143475	_b[/Residual:lnsigma]
delta	.3106636	_b[/Residual:delta]
ln_cons	-.335342	_b[/Residual:ln_cons]

If we examine the output carefully, we will notice that `menl, coeflegend` displayed results in the estimation metric—as log standard-deviations instead of variances. Although by default `menl` displays parameters in their original metric, it stores them in the estimation metric, the metric that was used during optimization; see *Examples of specifying initial values* and *Methods and formulas* for more details about the estimation metric.

The parameters we need to compute Cl^P and Cl are coefficient `_b[/b0]` and the variance of `U0`, which can be obtained as `exp(2*_b[/subject:lnsd(U0)])` based on the stored estimate of the log standard-deviation of `U0`. We now use `nlcom` to compute our nonlinear estimates.

```
. nlcom (Cl_P: exp(_b[/b0]+0.5*exp(2*_b[/subject:lnsd(U0)]))) (Cl: exp(_b[/b0]))
       Cl_P: exp(_b[/b0]+0.5*exp(2*_b[/subject:lnsd(U0)]))
         Cl: exp(_b[/b0])
```

conc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
Cl_P	.0402972	.002512	16.04	0.000	.0353738	.0452205
Cl	.0396646	.0024557	16.15	0.000	.0348516	.0444777

Working with parameters in the estimation metric can be tedious, especially when nonlinear expressions contain multiple variance components. In that case, you may consider using `estat sd` after `menl` to obtain results in the standard deviation metric or, if you also specify the `variance` option, in the variance metric; see [ME] **menl postestimation**. If you specify the `post` option with `estat sd`, the results will also be stored in the standard deviation or variance metrics, which you can use for further postestimation analysis.

```
. estat sd, post variance coeflegend
```

conc	Coefficient	Legend
/b0	-3.227295	_b[/b0]
/b1	.4354519	_b[/b1]
/b2	-2.453743	_b[/b2]

Random-effects parameters	Estimate	Legend
subject: Independent		
var(U0)	.0316416	_b[/subject:var(U0)]
var(U1)	.4500585	_b[/subject:var(U1)]
Residual variance:		
Power _yhat		
sigma2	.1015759	_b[/Residual:sigma2]
delta	.3106636	_b[/Residual:delta]
_cons	.7150935	_b[/Residual:_cons]

In addition to results being displayed in the variance metric, because of the `post` option, they are stored in that metric. We also specified the `coeflegend` option with `estat sd` to see how parameters are labeled so that we could refer to them in other postestimation commands such as `nlcom`.

Now, we can simply refer to the variance of U0 as _b[/subject:var(U0)] in our nlcom command.

```
. nlcom (Cl_P: exp(_b[/b0]+0.5*_b[/subject:var(U0)]))
       Cl_P: exp(_b[/b0]+0.5*_b[/subject:var(U0)])
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
Cl_P	.0402972	.002512	16.04	0.000	.0353738	.0452205

estat sd's post option should be used with caution because it clears all estimation results except the parameter estimates in e(b) and their VCE in e(V). Thus the only postestimation features that will work after estat sd, post are those that need only e(b) and e(V), such as lincom and nlcom. Other postestimation features will not be available, and you will need to refit your model to run them. To avoid refitting your model, you may consider storing your estimation results in memory (see [R] **estimates store**) or saving them on disk (see [R] **estimates save**) before using estat sd, post. We no longer needed the estimation results from menl, so we did not mind clearing them.

◁

Multiple-dose pharmacokinetic modeling

In example 15, a single dose of the analgesic theophylline was administered to each subject followed by multiple serum concentration measurements per subject. For long-duration illnesses, multiple doses are often given to each subject, with multiple serum concentration measurements interspersed. After a single-dose drug administration, the plasma drug level rises above and then falls below the minimum effective concentration, resulting in a decline in therapeutic effect. To treat chronic diseases, multiple-dosage or intravenous infusion regimens are used to maintain the plasma drug levels within the narrow limits of the therapeutic window to achieve optimal clinical effectiveness.

▷ Example 17: Multiple-intravenous-doses model

Grasela and Donn (1985) report a study of the neonatal PKs of phenobarbital. Data were collected on 59 preterm infants given phenobarbital for prevention of seizures during the first 16 days after birth. Each infant received one or more intravenous doses, dose (mg/kg). One to six blood serum phenobarbital concentration measurements, conc (mg/L), were obtained from each infant, subject, for a total of 155 measurements. The birthweight, in kilograms, and a five-minute Apgar score, a measure of the physical condition, were also obtained on each infant. The Apgar score is obtained by adding points (2, 1, or 0) for heart rate, respiratory effort, muscle tone, response to stimulation, and skin coloration; a score of 10 represents the best possible condition. time is measured in hours. Davidian and Giltinan (1995) and Pinheiro and Bates (2000) also analyze this dataset.

A one-compartment open model with intravenous administration and first-order elimination was used to model the PKs of this phenobarbital study

$$\texttt{conc}_{ij} = \sum_{t\le i} \frac{\texttt{dose}_{ik}}{V_j} \exp\left\{-\frac{\mathrm{Cl}_j}{V_j}\left(\texttt{time}_{ij} - \texttt{time}_{tj}\right)\right\} + \epsilon_{ij} \tag{19}$$

for $i = 1, \ldots, n_j$ and $j = 1, \ldots, 59$. Model parameters are the clearance Cl_j (L/h) and volume of distribution V_j (L) for each subject j. Clearance is the volume of blood or plasma that is totally cleared of its content of drug per unit time. It is the proportionality factor between the rate of elimination and concentration, $dC/dt = -k_e C = -(\mathrm{Cl}/V)\,C$, where C is the plasma concentration and k_e is the elimination rate (h^{-1}). The volume of distribution, V, is defined as the apparent space or volume into which a drug distributes.

To fit this model using menl, we consider an alternative recursive formulation of model (19)

$$\texttt{conc}_{ij} = \mu\left(\mathbf{x}'_{ij}, \boldsymbol{\beta}, \mathbf{u}_j\right) = \frac{\texttt{dose}_{ij}}{V_j} + \mu\left(\mathbf{x}'_{i-1,j}, \boldsymbol{\beta}, \mathbf{u}_j\right) \exp\left\{-\frac{\mathrm{Cl}_j}{V_j}\left(\texttt{time}_{ij} - \texttt{time}_{i-1,j}\right)\right\} + \epsilon_{ij}$$

Here, $\mathbf{x}'_{ij} = \left(\texttt{time}_{ij}, \texttt{dose}_{ij}, \texttt{fapgar}_j, \texttt{weight}_j\right)$ is the vector of covariates corresponding to subject j at $\texttt{time}_{ij}$. Notice that concentration $\texttt{conc}_{ij} = \mu\left(\mathbf{x}'_{ij}, \boldsymbol{\beta}, \mathbf{u}_j\right)$ depends on its previous expected value, $\mu\left(\mathbf{x}'_{i-1,j}, \boldsymbol{\beta}, \mathbf{u}_j\right)$, and on the time difference, $\texttt{time}_{ij} - \texttt{time}_{i-1,j}$. In Stata, we can use the lag operator, L., to refer to previous values and the difference operator, D., to refer to the difference between the two successive values. menl supports time-series operators in the model specification; see *Time-series operators*. We can use D.time to include the time difference in the model. However, we cannot simply use L.conc, because this would include the previous observed value of conc in the model, and we need the previous (predicted) value of the mean function. menl provides a special syntax L._yhat to include lagged predicted values or, equivalently, a special syntax L.{conc:} to include the lagged predicted mean function. {conc:} refers to the nonlinear expression for the mean function of the conc variable. Thus, our menl main specification of the recursive model would be

```
. menl conc = dose/{V:} + L.{conc:}*exp(-{Cl:}/{V:}*D.time), ...
```

where expressions for {V:} and {Cl:} will be defined later.

Because we are using time-series operators in the expression, we need to declare our data to be time-series data. There are two ways to do this: you can specify tsset prior to calling menl or you can specify the time variable in menl's option tsorder(); see *Time-series operators* for details. In this example, we will use the tsorder() option; see the technical note below for an example using tsset.

```
. menl conc = dose/{V:} + L.{conc:}*exp(-{Cl:}/{V:}*D.time), ... tsorder(time)
```

Let's take a quick look at our data by listing the observations for the first subject.

```
. use https://www.stata-press.com/data/r19/phenobarb
(Pharmacokinetics study of phenobarbital in neonatal infants)
. list if subject==1, sepby(subject)
```

	subject	weight	apgar	time	dose	conc	fapgar
1.	1	1.4	7	0	25	.	>= 5
2.	1	1.4	7	2	0	17.3	>= 5
3.	1	1.4	7	12.5	3.5	.	>= 5
4.	1	1.4	7	24.5	3.5	.	>= 5
5.	1	1.4	7	37	3.5	.	>= 5
6.	1	1.4	7	48	3.5	.	>= 5
7.	1	1.4	7	60.5	3.5	.	>= 5
8.	1	1.4	7	72.5	3.5	.	>= 5
9.	1	1.4	7	85.3	3.5	.	>= 5
10.	1	1.4	7	96.5	3.5	.	>= 5
11.	1	1.4	7	108.5	3.5	.	>= 5
12.	1	1.4	7	112.5	0	31	>= 5

The most noticeable feature of our PK data is the presence of many missing values for the concentration. In fact, this is a common structure of PK data in the presence of multiple doses. Notice that the conc variable contains missing values for each nonzero dose. It is typical to measure concentration only after a dose or multiple doses are administered, which gives rise to missing concentration at some time points. By default, Stata commands omit all observations containing missing values in variables used with the command. In this example, we need to retain missing conc observations. We can use menl's option tsmissing to do so.

```
. menl conc = dose/{V:} + L.{conc:}*exp(-{Cl:}/{V:}*D.time), ... tsmissing tsorder(time)
```

When you specify the `tsmissing` option, `menl` uses predicted values in place of system missing `conc` values in the computation. (Observations with extended missing values `.a`, `.b`, and so on in `conc`, if there were any, would have been omitted from the computation.) These predicted values are used to compute predicted values for the observed concentrations but are not used to compute the log likelihood. Only observed concentrations contribute to the log-likelihood calculation.

Another aspect of our data is that they are time-series data. Thus, the first observation in each panel provides starting values for the time-series operators. For example, from the data, the initial time value used by `D.time` for the first subject is $\texttt{time}_{i-1,j} = \texttt{time}_{0,1} = 0$. But how do we initialize `L.{conc:}` given that `{conc:}` does not exist as a variable in our dataset? We use `menl`'s option `tsinit()`.

```
. menl conc = dose/{V:} + L.{conc:}*exp(-{Cl:}/{V:}*D.time), ...
> tsinit({conc:}=dose/{V:}) tsmissing tsorder(time)
```

The `tsinit()` option allows us to specify initial conditions for the lagged predicted mean functions as expressions. In our example, the initial condition for the mean concentration for each subject j at time 0 is $\texttt{dose}_{0,j}/V_j$, which we specified in `tsinit()`.

Let's now return to our nonlinear model specification and provide expressions for `{V:}` and `{Cl:}`. One of the model parameterizations that Davidian and Giltinan (1995) consider for these data use `weight` as a covariate for clearance and volume. They also include a dichotomized Apgar score, factor variable `fapgar` in our dataset, to model volume. They express clearance and volume as

$$\begin{aligned} \text{Cl}_j &= \beta_1 \texttt{weight}_j \times \exp\left(u_{1j}\right) \\ V_j &= \beta_2 \texttt{weight}_j (1 + \beta_3 \texttt{fapgar}_j) \exp\left(u_{2j}\right) \end{aligned}$$

where u_{1j}'s and u_{2j}'s are two independent sets of random effects that follow $N(0, \sigma^2_{u1})$ and $N(0, \sigma^2_{u2})$, respectively.

We specify the above expressions for subject-specific volume and clearance in `menl` using the `define()` options and fit the model:

```
. menl conc = dose/{V:} + L.{conc:}*exp(-{Cl:}/{V:}*D.time),
> define(Cl: {cl:weight}*weight*exp({U1[subject]}))
> define(V: {v:weight}*weight*(1+{v:apgar}*1.fapgar)*exp({U2[subject]}))
> tsinit({conc:} = dose/{V:})
> tsmissing tsorder(time)

Panel variable: subject (unbalanced)
 Time variable: <time>, 1 to 20
         Delta: 1 unit

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -432.58887
Iteration 2:  Linearization log likelihood = -436.35525
Iteration 3:  Linearization log likelihood = -436.36735
Iteration 4:  Linearization log likelihood = -436.36894
Iteration 5:  Linearization log likelihood =   -436.369
Iteration 6:  Linearization log likelihood = -436.36896

Computing standard errors:

Mixed-effects ML nonlinear regression               Number of obs =        685
                                                       Nonmissing =        155
                                                          Missing =        530

      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
subject	59	1	11.6	19
conc	59	1	2.6	6

```
Linearization log likelihood = -436.36896
          Cl: {cl:weight}*weight*exp({U1[subject]})
           V: {v:weight}*weight*(1+{v:apgar}*1.fapgar)*exp({U2[subject]})
```

conc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
/cl						
weight	.004705	.0002219	21.20	0.000	.0042701	.00514
/v						
weight	.9657032	.0294438	32.80	0.000	.9079945	1.023412
apgar	.1749755	.0845767	2.07	0.039	.0092082	.3407429

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Independent				
var(U1)	.0404098	.0187133	.0163044	.1001537
var(U2)	.030259	.0078857	.0181562	.0504295
var(Residual)	7.469354	1.280411	5.337875	10.45196

```
Note: Lagged predicted mean function L.{conc:} is used in the model.
```

From the coefficient table, we see that heavier babies have a higher clearance and volume of distribution. There is a positive association between the volume of distribution and the Apgar score: healthier babies have a better ability to eliminate the drug.

Because we specified the `tsmissing` option, the header reported the number of missing and nonmissing concentration values used in the computation. Also, the table containing the information about the number of groups has an additional entry for `conc` providing the group information for nonmissing observations of `conc`.

When we specified the time variable in the `tsorder()` option, `menl` generated the corresponding consecutive integer-valued time variable and used it with `tsset`. From the output of `tsset`, as displayed by `menl`, we see that `menl` also identified the panel variable, `subject`, from our model specification and used it with `tsset`. The generated time variable used with `tsset` is labeled as `<time>` in the output.

◁

❑ Technical note

In example 17, we used the `tsorder()` option to specify the ordering for time-series operators. We could have used `tsset` instead, but we would need to create the appropriate time variable first. Here, we demonstrate how to do this.

We must specify the panel and time variables with `tsset`. Intuitively, we would want to type

```
. tsset subject time
```

but that would not produce the intended results. First, `tsset` requires an integer time variable, which the `time` variable is not. Second, even if `time` contained integers, it is not equally spaced, which would lead to gaps in the time series and thus missing values for time-series operators.

In our example, we are concerned only with the ordering of observations within a subject with respect to the `time` variable for the purpose of time-series operators. So, we create a new variable, `tsorder`, to contain consecutive integers based on `time` and use it with `tsset`.

```
. sort subject time
. by subject (time): generate long tsorder = _n
. tsset subject tsorder
Panel variable: subject (unbalanced)
 Time variable: tsorder, 1 to 20
         Delta: 1 unit
```

You can verify that the following specification of `menl` will produce the same results as in example 17.

```
. menl conc = dose/{V:} + L.{conc:}*exp(-{Cl:}/{V:}*D.time),
> define(Cl: {cl:weight}*weight*exp({U1[subject]}))
> define(V: {v:weight}*weight*(1+{v:apgar}*1.fapgar)*exp({U2[subject]}))
> tsinit({conc:} = dose/{V:})
> tsmissing
  (output omitted)
```

Note that we still use the `time` variable with the difference operator, `D.`, in the model specification.

❑

Example 18: Multiple-oral-doses model

Verme et al. (1992) evaluated the PK behavior of quinidine, a pharmaceutical agent used to prevent cardiac arrhythmias, in a study of 136 subjects receiving oral quinidine therapy. A total of 361 serum quinidine concentrations (variable `conc`, mg/L) were measured over time (variable `time`, hours), ranging from 1 to 11 observations per subject. Multiple doses (variable `dose`, mg) of quinidine, in two different forms, were administrated to each subject. The doses were adjusted for differences in salt content by conversion into milligrams of quinidine base. These data are also presented as examples in Davidian and Giltinan (1995) and Pinheiro and Bates (2000).

A one-compartment open model with first-order absorption and elimination is assumed for serum quinidine concentrations. This model, expressed in a compact recursive form, is

$$\texttt{conc}_{ij} = \mu_1\left(\mathbf{x}'_{ij},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) = \mu_1\left(\mathbf{x}'_{i-1,j},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) Q_{e_{ij}} + \mathrm{Ca}_{i-1,j}\frac{k_{a_j}}{k_{a_j} - k_{e_j}}\left(Q_{e_{ij}} - Q_{a_{ij}}\right) + \epsilon_{ij} \qquad (20)$$

where

$$\begin{aligned}
\mathrm{Ca}_{ij} =& \mu_2\left(\mathbf{z}'_{ij},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) = \mu_2\left(\mathbf{z}'_{i-1,j},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) Q_{a_{ij}} + \frac{\texttt{dose}_{ij}}{V_j} \\
Q_{e_{ij}} =& \exp\left\{-k_{e_j}\left(\texttt{time}_{ij} - \texttt{time}_{i-1,j}\right)\right\} \\
Q_{a_{ij}} =& \exp\left\{-k_{a_j}\left(\texttt{time}_{ij} - \texttt{time}_{i-1,j}\right)\right\}
\end{aligned}$$

for subject $j = 1, \ldots, 136$ and subject observation $i = 1, \ldots, n_j$, $n_j \in [1, 11]$. The quantities $Q_{a_{ij}}$ and $Q_{e_{ij}}$ are defined for notational convenience to simplify the model expression. $\mathbf{z}'_{ij} = \left(\texttt{time}_{ij}, \texttt{dose}_{ij}\right)$ and $\mathbf{x}'_{ij} = \left(\mathbf{z}'_{ij}, \texttt{glyco}_{ij}, \texttt{creatinine}_j, \texttt{weight}_j\right)$ are vectors of covariates, which we describe later, corresponding to subject j at $\texttt{time}_{ij}$. Because the drug administration is extravascular, the quinidine concentration in the body over time is a function of both the absorption rate, k_{a_j}, and the elimination rate, k_{e_j}, for subject j. The function Ca_{ij} is the apparent concentration of quinidine in the absorption depot over time (indexed by i) for subject j.

From example 17, we know that $k_{e_j} = \mathrm{Cl}_j/V_j$, where Cl_j is the clearance, defined as the volume of plasma or blood that is totally cleared from its content of drug per unit time, and V_j is the apparent volume of distribution, defined as theoretical volume that would be necessary to contain the total amount of an administered drug at the same concentration that is observed in the blood plasma.

The `menl` specification corresponding to model (20) is

```
. menl conc = L.{conc:}*{Qe:}+L.{Ca:}*({ka:}/({ka:}-{ke:}))*({Qe:}-{Qa:}),
> define(Ca: L.{Ca:}*{Qa:}+dose/{V:})
> define(Qe: exp(-{ke:}*D.time))
> define(Qa: exp(-{ka:}*D.time))
> define(ke: {Cl:}/{V:})
> define(ka: exp({lka}))
> ...
```

where expressions for `{Cl:}` and `{V:}` will be defined later. Similarly to example 17, we use `D.time` to specify differences between two successive time values and `L.{conc:}` to specify the lagged predicted mean function; also see *Time-series operators*. New in this specification is the inclusion of the lagged function of model parameters or lagged named expression `L.{Ca:}`. Expression `Ca` is defined in the `define()` option and is a function of its own lag, `L.{Ca:}`. Finally, parameter `{ka}` is reparameterized as `exp({lka})` to ensure that it is positive.

When a patient receives the same dosage at regular time intervals (variable `interval`), model (20) simplifies to the steady-state model

$$\texttt{conc}_{ij}^{ss} = \frac{\texttt{dose}_{ij} k_{a_j}}{V_j\left(k_{a_j} - k_{e_j}\right)}\left(Q_{e_{ij}}^{ss} - Q_{a_{ij}}^{ss}\right) \tag{21}$$

and

$$\text{Ca}_{ij}^{ss} = \frac{\texttt{dose}_{ij}}{V_j} Q_{a_{ij}}^{ss}$$

where

$$Q_{e_{ij}}^{ss} = \frac{1}{1 - \exp\left(-k_{e_j}\texttt{interval}_{ij}\right)}$$

$$Q_{a_{ij}}^{ss} = \frac{1}{1 - \exp\left(-k_{a_j}\texttt{interval}_{ij}\right)}$$

The quantities $Q_{e_{ij}}^{ss}$ and $Q_{a_{ij}}^{ss}$ are also defined for notational convenience.

The `menl` specification corresponding to model (21) is

```
. menl conc = dose*{ka:}/({V:}*({ka:}-{ke:}))*({Qe_ss:} - {Qa_ss:}),
> define(Qe_ss: 1/(1-exp(-{ke:}*interval))
> define(Qa_ss: 1/(1-exp(-{ka:}*interval))
> define(ke: {Cl:}/{V:})
> define(ka: exp({lka}))
> ...
```

For the quinidine model, the steady-state model (21) is assumed whenever $\texttt{interval}_{ij}$ is nonzero and the nonsteady-state model (20) is assumed otherwise. Thus, we need to switch back and forth between these two models in our `menl` specification. We can use the Stata function `cond(`*condition*`,`*expr_if_condition_true*`,`*expr_if_condition_false*`)`.

For example, the `menl` specification becomes

```
. menl conc = cond(interval==0,
>                  L.{conc:}*{Qe:}+L.{Ca:}*({ka:}/({ka:}-{ke:}))*({Qe:}-{Qa:}),
>                  dose*{ka:}/({V:}*({ka:}-{ke:}))*({Qe_ss:} - {Qa_ss:})),
> define(Ca: cond(interval==0, L.{Ca:}*{Qa:}+dose/{V:}, dose/{V:}*{Qa_ss:})
> ...
```

where other expressions such as `{Qe:}` and `{Qa_ss:}` are as defined earlier. We used `cond()` for the main `menl` specification and for the definition of the `{Ca:}` function.

Recall from example 17 that when we specify the lagged predicted mean function, we need to specify an initial condition for it in the `tsinit()` option. Just like the main nonlinear specification, the initial condition for `L.{conc:}` will depend on the value of `interval`. The mean concentration at time 0 will be 0 for observations with zero `interval` values and will be equal to the expression for the steady-state model otherwise: `tsinit({conc:}=cond(interval==0,0,dose*{ka:}/({V:}*({ka:}-{ke:}))*({Qe_ss:}-{Qa_ss:})))`. Similarly, we need to provide an initial condition for the lagged function of model parameters `L.{Ca:}`. It also depends on `interval`: `tsinit({Ca:}=cond(interval==0,dose/{V:},dose/{V:}*{Qa_ss:}))`. Because we are using the same expressions in the function definitions and the initial conditions, we can define additional functions to minimize typing:

```
. menl conc = cond(interval==0,
>                  L.{conc:}*{Qe:}+L.{Ca:}*({ka:}/({ka:}-{ke:}))*({Qe:}-{Qa:}),
>                  {Css:}),
> define(Ca: cond(interval==0,L.{Ca:}*{Qa:}+dose/{V:}, {Ca_ss:})
> define(Css: cond(interval==0,0,dose*{ka:}/({V:}*({ka:}-{ke:}))*({Qe_ss:}-{Qa_ss:})))
> define(Ca_ss: dose/{V:}*{Qa_ss:})
> ...
> tsinit({conc:} = cond(interval==0, 0, {Css:})
> tsinit({Ca:} = cond(interval==0, dose/{V:}, {Ca_ss:})
```

`{Css:}` contains the expression for the steady-state model (or 0 for observations in a nonsteady state), and `{Ca_ss:}` contains the expression for the `Ca` function in the steady state.

Let's now finalize our `menl` specification by defining expressions for `{Cl:}` and `{V:}`. The goal of the study from Verme et al. (1992) was to examine the relationship between quinidine PKs and several potential covariates: body `weight` (kg); `age` (years); `height` (in); `glyco`, α_1-acid glycoprotein concentration (mg/dL); `creatinine`, creatinine clearance ($\geq$ 50 or $<$ 50 ml/min); `race` (Caucasian, Latin, black); `smoke`, smoking status (yes, no); `ethanol`, alcohol abuse (former, none, current); and `heart`, congestive heart failure (no or mild, moderate, severe). We provide more details about covariates `creatinine` and `glyco` below.

Creatinine is a waste product from the normal breakdown of muscle tissue. As creatinine is produced, it is filtered through the kidneys and excreted in urine. Doctors use creatinine and creatinine clearance tests to check renal function (kidney function). Testing the rate of creatinine clearance shows the kidneys' ability to filter the blood. As renal function declines, creatinine clearance also goes down. Creatinine clearance in a healthy young person is about 95 ml/min for women and 120 ml/min for men.

α_1-acid glycoprotein (also known as AAG) is an important plasma protein involved in the binding and transport of many drugs, including quinidine. A healthy range is 50–120 mg/dl. Changes in AAG concentration could potentially alter the free fraction of drugs in plasma or at their target sites and eventually affect their PK disposition and pharmacological action. Because AGG levels are increased in response to stress, serum levels of total quinidine may be greatly increased in settings such as acute myocardial infarction. Protein binding is also increased in chronic renal failure. There tends to be a small increase in AAG with age.

For the purpose of illustration, we fit a modified version of model 2 from pages 248–249 of Davidian and Giltinan (1995). The clearance, Cl_j, is modeled on the log scale as a linear combination `{lCl:}` of `glyco`, `ib1.creatinine`, `weight`, and a random intercept, `U1`, at the `subject` level. The apparent volume, V_j, is modeled on the log scale using a fixed-effect intercept and `weight`. The absorption rate, k_a, is modeled on the log scale as a free parameter `{lka}`, and is assumed fixed for all subjects. The full second-stage specification is as follows:

$$
\begin{aligned}
\mathrm{Cl}_{ij} &= \exp\left(\beta_1 + \beta_2 \texttt{glyco}_{ij} + \beta_3 \texttt{creatinine}_j + \beta_4 \texttt{weight}_j + u_{1j}\right) \\
V_j &= \exp\left(\beta_5 + \beta_6 \texttt{weight}_j\right) \\
k_{a_j} &= \exp\left(\beta_7\right) \\
k_{e_{ij}} &= \frac{\mathrm{Cl}_{ij}}{V_j}
\end{aligned}
$$

where u_{1j}'s are random effects that follow $N(0, \sigma^2_{u1})$.

Similarly to the phenobarbital data from example 17, the quinidine data also contain missing concentration values, so we specify the `tsmissing` option to retain them in the computation. Again, we will specify the time variable in the `tsorder()` option and let `menl` `tsset` the data for us.

```
. use https://www.stata-press.com/data/r19/quinidine
. menl conc = cond(interval==0,
>                  L.{conc:}*{Qe:}+L.{Ca:}*({ka:}/({ka:}-{ke:}))*({Qe:}-{Qa:}),
>                  {Css:}),
> define(Ca: cond(interval==0, L.{Ca:}*{Qa:}+dose/{V:}, {Ca_ss:}))
> define(Qe: exp(-{ke:}*D.time))
> define(Qa: exp(-{ka:}*D.time))
> define(Css: cond(interval==0,0,{ka:}*dose/({V:}*({ka:}-{ke:}))*({Qe_ss:}-{Qa_ss:})))
> define(Ca_ss: cond(interval==0,0,dose/{V:}*{Qa_ss:}))
> define(Qe_ss: 1/(1-exp(-{ke:}*interval)))
> define(Qa_ss: 1/(1-exp(-{ka:}*interval)))
> define(ke: {Cl:}/{V:})
> define(ka: exp({lka}))
> define(Cl: exp({lCl:glyco ib1.creatinine weight U1[subject], xb}))
> define(V: exp({lV: weight, xb}))
> tsinit({conc:} = cond(interval==0, 0, {Css:}))
> tsinit({Ca:} = cond(interval==0, dose/{V:}, {Ca_ss:}))
> tsorder(time) tsmissing

Panel variable: subject (unbalanced)
 Time variable: <time>, 1 to 47
         Delta: 1 unit

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -423.26688
Iteration 2:  Linearization log likelihood = -425.82312
Iteration 3:  Linearization log likelihood = -425.81124
Iteration 4:  Linearization log likelihood =  -425.8119
Iteration 5:  Linearization log likelihood = -425.81241
Iteration 6:  Linearization log likelihood = -425.81223
Iteration 7:  Linearization log likelihood = -425.81233
Iteration 8:  Linearization log likelihood = -425.81228
Iteration 9:  Linearization log likelihood = -425.81231

Computing standard errors:

Mixed-effects ML nonlinear regression               Number of obs =      1,335
                                                       Nonmissing =        361
                                                          Missing =        974

      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
subject	136	1	9.8	46
conc	136	1	2.7	11

```
                                                Wald chi2(4)      =     169.94
Linearization log likelihood = -425.81231       Prob > chi2       =     0.0000

          Ca: cond(interval==0,L.{Ca:}*{Qa:}+dose/{V:},{Ca_ss:})
       Ca_ss: cond(interval==0,0,dose/{V:}*{Qa_ss:})
          Cl: exp({lCl:})
         Css: cond(interval==0,0,{ka:}*dose/({V:}*({ka:}-{ke:}))*({Qe_ss:}-{
              Qa_ss:}))
          Qa: exp(-{ka:}*D.time)
       Qa_ss: 1/(1-exp(-{ka:}*interval))
          Qe: exp(-{ke:}*D.time)
       Qe_ss: 1/(1-exp(-{ke:}*interval))
           V: exp({lV:})
          ka: exp({lka})
          ke: {Cl:}/{V:}
         lCl: glyco ib1.creatinine weight U1[subject], xb
          lV: weight, xb
```

conc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
lCl						
glyco	-.4689097	.0416876	-11.25	0.000	-.5506159	-.3872035
creatinine						
>= 50	.1851334	.0464825	3.98	0.000	.0940294	.2762373
weight	.0036181	.0018213	1.99	0.047	.0000485	.0071877
_cons	2.668191	.1524726	17.50	0.000	2.36935	2.967031
lV						
weight	.0087346	.0058603	1.49	0.136	-.0027514	.0202206
_cons	4.572762	.47765	9.57	0.000	3.636585	5.508939
/lka	-.8956278	.301	-2.98	0.003	-1.485577	-.3056787

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Identity				
var(U1)	.0589024	.0108271	.0410838	.0844492
var(Residual)	.4122599	.0364831	.346612	.4903413

```
Note: Lagged predicted mean function L.{conc:} is used in the model.
Note: Lagged named expression L.{Ca:} is used in the model.
```

From the coefficient table, we see that the clearance decreases with increase of AAG (`glyco`) as would be expected with the greater protein binding. The clearance is greater for creatinine clearance ≥ 50 as would be expected with better renal function. Both clearance and volume increase with weight; although, the effect of weight on volume is not statistically significant at the 5% level. The subject variability for clearance contributes to the model as seen by the confidence interval for the random-effects variance `var(U1)`.

◁

Nonlinear marginal models

The variance–covariance matrix of the response vector $\mathbf{y}_j = \left(y_{1j}, \ldots, y_{n_j j}\right)$ involves two components to model heteroskedasticity and correlation: A random-effects component $\boldsymbol{\Sigma}$ and a within-group error component $\boldsymbol{\Lambda}_j$. In some applications, one may wish to directly model the covariance structure of

the response by choosing the appropriate within-group error component $\mathbf{\Lambda}_j$ without introducing random effects. This results in the so-called nonlinear marginal model (for example, Pinheiro and Bates [2000, sec. 7.5.1]):

$$\text{Stage 1: Individual-level model } \mathbf{y}_j = m\left(\mathbf{x}_j^w,\, \boldsymbol{\phi}_j\right) + \boldsymbol{\epsilon}_j \qquad \boldsymbol{\epsilon}_j \sim N\left(\mathbf{0}, \sigma^2\mathbf{\Lambda}_j\right)$$
$$\text{Stage 2: Group-level model } \boldsymbol{\phi}_j = \boldsymbol{d}\left(\mathbf{x}_j^b,\, \boldsymbol{\beta}\right) \qquad j = 1, \ldots, M$$

The above is essentially a vector representation of (2) after excluding the random effects $\mathbf{u}_j$. Random effects are used in NLME models to explain the between-subject or between-group variation, but they are not used in the specification of nonlinear marginal models. This key difference implies that mixed-effects models allow for subject-specific inference, whereas marginal models do not. For this reason, mixed-effects models are often called subject-specific models, while marginal models are called population-averaged models.

`menl` provides the `group()` suboption within the `rescovariance()` and `rescorrelation()` options to model the dependence between within-group observations without introducing random effects. Below, we show an example of fitting a nonlinear marginal model, without random effects, using the `group()` suboption. See example 22 for the usage of the `group()` suboption in the presence of random effects.

▷ Example 19: Nonlinear marginal model

Vonesh and Carter (1992) analyzed data on 20 high-flux hemodialyzers to assess their in-vitro ultrafiltration performance. Dialyzers are used in hemodialysis, a treatment that replaces the work of kidneys, to filter harmful wastes out of blood for patients with kidney failure. High-flux dialyzers do this more efficiently than conventional dialyzers—they are composed of membranes with larger pores, which allows them to remove larger molecules and water during blood filtration. A dialyzer's ultrafiltration performance, or ability to filter blood, is controlled by so-called transmembrane pressure and also depends on the blood flow rate used during hemodialysis. In these data, the response variable, `rate`, is the dialyzer's ultrafiltration rate in mL/hr measured at 7 different transmembrane pressures, `pressure`, in dmHg. Ten dialyzers were evaluated using bovine blood at a blood flow rate, `qb`, of 200 mL/min, whereas the other 10 dialyzers were evaluated at 300 mL/min.

The ultrafiltration rate, $\texttt{rate}_{ij}$, at the ith transmembrane pressure, $\texttt{pressure}_{ij}$, for the jth subject is represented by the nonlinear model

$$\texttt{rate}_{ij} = \phi_{1j}\left[1 - \exp\left\{-\exp(\phi_{2j})\left(\texttt{pressure}_{ij} - \phi_3\right)\right\}\right] + \epsilon_{ij}$$

The parameters ϕ_1, ϕ_2, and ϕ_3 have physiological interpretation: ϕ_1 is the maximum attainable ultrafiltration rate, ϕ_2 is the logarithm of the hydraulic permeability transport rate of the membrane (rate at which water and molecules pass through the dialyzer membrane), and ϕ_3 is the transmembrane pressure required to offset the oncotic pressure (the transmembrane pressure at which the ultrafiltration rate is 0).

One of the models proposed in Vonesh and Carter (1992) included no random effects and used an exchangeable (also known as compound symmetry) covariance structure to model the within-dialyzer error covariance structure. The full description of the second stage of the model is

$$\phi_{1j} = \beta_{10} + \beta_{11}\texttt{qb}_j$$
$$\phi_{2j} = \beta_{20} + \beta_{21}\texttt{qb}_j$$
$$\phi_{3j} = \beta_3$$

and

$$\boldsymbol{\epsilon}_j \sim N(\mathbf{0}, \sigma^2_\epsilon \boldsymbol{\Lambda}_j), \quad \boldsymbol{\Lambda}_j = \begin{bmatrix} 1 & \rho & \dots & \rho \\ & 1 & \dots & \rho \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}$$

Below, we use `rescovariance(exchangeable, group(dialyzer))` to request an exchangeable within-group error covariance structure where groups are identified by the `dialyzer` variable.

```
. use https://www.stata-press.com/data/r19/dialyzer
(High-flux hemodialyzers (Vonesh and Carter, 1992))
. menl rate = {phi1:}*(1-exp(-exp({phi2:})*(pressure - {phi3}))),
> define(phi1: i.qb, xb) define(phi2: i.qb, xb)
> rescovariance(exchangeable, group(dialyzer)) stddev
Obtaining starting values:
Alternating GNLS/ML algorithm:
Iteration 1:  Log likelihood = -365.34244
Iteration 2:  Log likelihood = -365.32697
Iteration 3:  Log likelihood = -365.32697
Iteration 4:  Log likelihood = -365.32697
Iteration 5:  Log likelihood = -365.32697
Iteration 6:  Log likelihood = -365.32697
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        140
Group variable: dialyzer                        Number of groups  =         20
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
                                                Wald chi2(2)      =     194.77
Log likelihood = -365.32697                     Prob > chi2       =     0.0000
       phi1: i.qb
       phi2: i.qb
```

rate	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
qb						
300	17.23062	1.24589	13.83	0.000	14.78872	19.67252
_cons	44.95795	.8841506	50.85	0.000	43.22505	46.69086
phi2						
qb						
300	-.5034708	.0763513	-6.59	0.000	-.6531166	-.353825
_cons	.7626986	.0630914	12.09	0.000	.6390417	.8863555
/phi3	.2249104	.0102113	22.03	0.000	.2048965	.2449243

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
Residual: Exchangeable				
sd	3.722521	.3064517	3.167839	4.374327
corr	.3867847	.0993617	.1771206	.5628662

The estimated values of ρ and σ_ϵ are $\hat{\rho} = 0.39$ and $\hat{\sigma}_e = 3.72$, respectively. The 95% confidence interval [0.18, 0.56] for ρ suggests a positive correlation within dialyzer measurements. The maximum ultrafiltration rate, ϕ_1, and the logarithm of the hydraulic permeability transport rate, ϕ_2, appear to be affected by the blood flow rate.

◁

Three-level models

Representation of (1) can be extended to, for example, two-nested levels of clustering, to form the following three-level model, with observations composing the first level,

$$\mathbf{y}_{jk} = \boldsymbol{\mu}\left(\mathbf{X}_{jk}, \boldsymbol{\beta}, \mathbf{u}_k^{(3)}, \mathbf{u}_{jk}^{(2)}\right) + \boldsymbol{\epsilon}_{jk}$$

where the first-level observations $i = 1, \ldots, n_{jk}$ are nested within the second-level groups $j = 1, \ldots, M_k$, which are nested within the third-level groups $k = 1, \ldots, M$. Group j nested within group k consists of n_{jk} observations, so $\mathbf{y}_{jk}$, $\mathbf{X}_{jk}$, and $\boldsymbol{\epsilon}_{jk}$ each have row dimension n_{jk}.

Also, assume that

$$\mathbf{u}_k^{(3)} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_3) \qquad \mathbf{u}_{jk}^{(2)} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_2) \qquad \boldsymbol{\epsilon}_{jk} \sim N(\mathbf{0}, \sigma^2 \boldsymbol{\Lambda}_{jk})$$

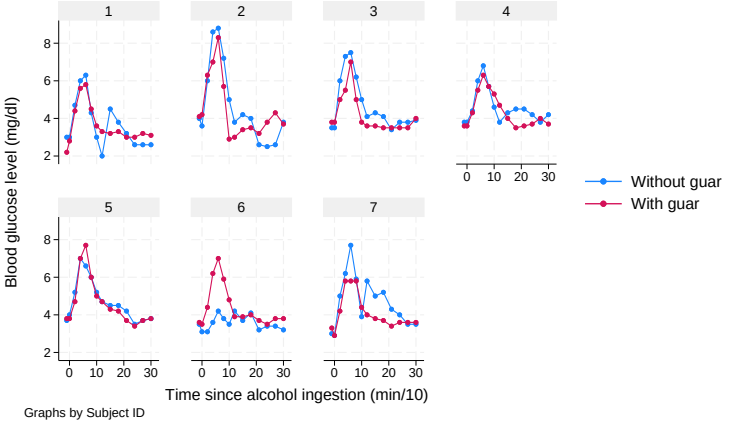
and that $\mathbf{u}_k^{(3)}$, $\mathbf{u}_{jk}^{(2)}$, and $\boldsymbol{\epsilon}_{jk}$ are independent.

▷ Example 20: Three-level model

Hand and Crowder (1996, 118–120) analyzed a study where the blood glucose levels `glucose` of 7 volunteers, `subject`, who took alcohol at time 0 were measured 14 times, `time`, over a period of 5 hours after alcohol consumption. The same experiment was repeated at a later date with the same subjects but with a dietary additive, guar, used for all subjects. Variable `guar` is a binary variable that identifies whether a subject received a dietary additive. It also identifies each experiment, with 0 corresponding to the experiment without guar and 1 corresponding to the experiment with guar. Thus we will use the `guar` variable both as the level indicator and, later, as a fixed-effects variable.

Here is a plot of the whole dataset.

```
. use https://www.stata-press.com/data/r19/glucose
(Glucose levels following alcohol ingestion (Hand and Crowder, 1996))
. twoway connected glucose time if guar==0 ||
> connected glucose time if guar==1 ||, by(subject, rows(2))
> legend(order(1 "Without guar" 2 "With guar"))
```



Our preliminary assessment based on the above graph is that, except for subject 6, the effect of the dietary additive guar on the temporal trajectory of the blood glucose levels does not seem to be important. The effect of guar will be formally tested in example 21.

Hand and Crowder (1996) proposed the following empirical model relating the expected glucose level to time,

$$\texttt{glucose}_{ijk} = \phi_{1jk} + \phi_{2jk}\texttt{time}^3 \exp\left(-\phi_{3jk}\texttt{time}\right) + \epsilon_{ijk} \tag{22}$$

where $k = 1, \ldots, 7$, $j = 1, 2$, and $i = 1, \ldots, 14$. The blood glucose level is ϕ_1 at $\texttt{time} = 0$ and as $\texttt{time} \to \infty$. This is intentional, so that ϕ_1 can be interpreted as both the blood glucose level before ingesting alcohol and the blood glucose level after the effect of alcohol ingestion has washed out.

Pinheiro and Bates (2000, exercise 3, 412) analyzed this dataset in the context of a three-level NLME model. They initially proposed the following stage 2 specification,

$$\begin{aligned} \phi_{1jk} &= \beta_1 + u^{(3)}_{1k} + u^{(2)}_{1j,k} \\ \phi_{2jk} &= \beta_2 + u^{(3)}_{2k} + u^{(2)}_{2j,k} \\ \phi_{3jk} &= \beta_3 \end{aligned} \tag{23}$$

$$\mathbf{u}^{(3)}_k = \begin{bmatrix} u^{(3)}_{1k} \\ u^{(3)}_{2k} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_3) \qquad \mathbf{u}^{(2)}_{j,k} = \begin{bmatrix} u^{(2)}_{1j,k} \\ u^{(2)}_{2j,k} \end{bmatrix} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_2) \qquad \epsilon_{ijk} \sim N(0, \sigma^2_\epsilon)$$

where $\boldsymbol{\Sigma}_2$ and $\boldsymbol{\Sigma}_3$ are general symmetric covariance matrices. $u^{(2)}_{1j,k}$ and $u^{(2)}_{2j,k}$ are random intercepts at the guar-within-subject level and can be specified in menl as `UU1[subject>guar]` and `UU2[subject>guar]`.

The full model defined by (22) and (23) contains many parameters. We will follow our own advice from example 11 and specify the `iterate()` option to check how reasonable our model is for the data we have.

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3}*time),
> define(phi1: U1[subject] UU1[subject>guar])
> define(phi2: U2[subject] UU2[subject>guar])
> covariance(U1 U2, unstructured) covariance(UU*, unstructured)
> stddeviations iterate(3)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -189.44711
Iteration 2:  Linearization log likelihood = -189.44117
Iteration 3:  Linearization log likelihood = -189.44112

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        196
      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
subject	7	28	28.0	28
subject>guar	14	14	14.0	14

```
Linearization log likelihood = -189.44112
        phi1: U1[subject] UU1[subject>guar]
        phi2: U2[subject] UU2[subject>guar]
```

glucose	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	3.661565	.1160345	31.56	0.000	3.434142	3.888989
phi2						
_cons	.4283298	.0530029	8.08	0.000	.3244461	.5322136
/phi3	.5896813	.013861	42.54	0.000	.5625144	.6168483

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Unstructured				
sd(U1)	.2624562	.0926845	.1313594	.5243876
sd(U2)	.0598433	.0724562	.0055771	.6421277
corr(U1,U2)	-.1489335	.9201199	-.963627	.9346878
subject>guar: Unstructured				
sd(UU1)	.0919525	.076423	.0180351	.4688234
sd(UU2)	.122707	.041287	.0634552	.2372856
corr(UU1,UU2)	.99999	.0044417	-1	1
sd(Residual)	.5712261	.0305339	.514409	.6343187

```
Warning: Convergence not achieved.
```

The estimated correlation `corr(UU1,UU2)` is near one with the confidence interval spanning the entire range for the correlation parameter, which indicates that the random-effects structure is overparameterized. The confidence interval for `corr(U1,U2)` contains zero, which suggests that this term does not contribute much to explaining between-subject variability. If we try to fit this model without the `iterate()` option, it will continue iterating without convergence.

We simplify our model by assuming independence between random effects; that is, we assume that random-effects covariance matrices Σ_2 and Σ_3 are diagonal.

Recall that `covariance(, independent)` is assumed by default, so we do not need to explicitly specify the `covariance()` option:

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3}*time),
> define(phi1: U1[subject] UU1[subject>guar])
> define(phi2: U2[subject] UU2[subject>guar]) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -190.35529
Iteration 2:  Linearization log likelihood = -190.36034
Iteration 3:  Linearization log likelihood =  -190.3633
Iteration 4:  Linearization log likelihood = -190.36418
Iteration 5:  Linearization log likelihood = -190.36375
Iteration 6:  Linearization log likelihood = -190.36397
Iteration 7:  Linearization log likelihood = -190.36386
Iteration 8:  Linearization log likelihood = -190.36391
Iteration 9:  Linearization log likelihood = -190.36389

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        196
      Grouping information
------------------------------------------------------------------------------
                        |     No. of       Observations per group
                   Path |     groups    Minimum    Average    Maximum
---------------------------------------------------------------------
                subject |          7         28       28.0         28
            subject>guar |         14         14       14.0         14
------------------------------------------------------------------------------

Linearization log likelihood = -190.36389
        phi1: U1[subject] UU1[subject>guar]
        phi2: U2[subject] UU2[subject>guar]

------------------------------------------------------------------------------
     glucose | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
phi1         |
       _cons |   3.658712   .1168642    31.31   0.000     3.429662    3.887762
-------------+----------------------------------------------------------------
phi2         |
       _cons |   .4239173   .0526333     8.05   0.000      .320758    .5270766
-------------+----------------------------------------------------------------
      /phi3 |   .5876636   .0137214    42.83   0.000     .5607701    .6145571
------------------------------------------------------------------------------
```

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Independent				
sd(U1)	.2685609	.092104	.137126	.5259757
sd(U2)	.0422075	.1078502	.0002821	6.315575
subject>guar: Independent				
sd(UU1)	.0666034	.1527523	.0007435	5.966157
sd(UU2)	.1362263	.0433548	.0730065	.2541912
sd(Residual)	.5732488	.0309928	.5156118	.6373288

The random-effects structure may still be overparameterized, given small estimates for `sd(U2)` and `sd(UU1)`. If we were to perform an LR test of the corresponding variance components being zero, we would have no statistical evidence to reject this null hypothesis; see example 7 for an instance of performing an LR test.

◁

▷ Example 21: Three-level model with continuous-time AR(1) error structure

The main objective of the study from example 20 was to determine whether the use of the dietary additive guar significantly affected time profiles of the blood glucose levels of subjects.

We continue with the model without random effects `U2[subject]` and `UU1[subject>guar]` and include covariate `guar` for all ϕ_{jk}'s. Hand and Crowder (1996) also suggested to use a continuous-time AR(1) correlation structure for the `guar`-within-`subject` errors, which is specified in `menl` as `rescorrelation(ctar1, t(time))`:

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3:}*time),
> define(phi1: i.guar U1[subject]) define(phi2: i.guar UU2[subject>guar])
> define(phi3: i.guar, xb) rescorrelation(ctar1, t(time)) stddeviations

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -180.62304
  (iteration log omitted)
Iteration 25: Linearization log likelihood = -181.18699

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        196
      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
subject	7	28	28.0	28
subject>guar	14	14	14.0	14

```
                                                Wald chi2(3)      =       0.66
Linearization log likelihood = -181.18699       Prob > chi2       =     0.8814
        phi1: i.guar U1[subject]
        phi2: i.guar UU2[subject>guar]
        phi3: i.guar
```

glucose	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
guar						
with guar	-.0814355	.1532735	-0.53	0.595	-.381846	.218975
_cons	3.685365	.1433368	25.71	0.000	3.40443	3.9663
phi2						
guar						
with guar	.0109469	.0883807	0.12	0.901	-.162276	.1841698
_cons	.344372	.0606914	5.67	0.000	.2254191	.4633248
phi3						
guar						
with guar	.0103743	.0330196	0.31	0.753	-.054343	.0750916
_cons	.5514012	.022009	25.05	0.000	.5082642	.5945381

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Identity				
sd(U1)	.2453634	.1013233	.1092206	.5512074
subject>guar: Identity				
sd(UU2)	.1011852	.0276419	.0592358	.1728421
Residual: CTAR1, time time				
sd(e)	.6208598	.0412948	.544977	.7073086
rho	.6547722	.0564848	.544064	.7654804

The dietary additive guar does not seem to affect the blood-glucose-level profiles over time. This actually conforms with the plot of the data from example 20, where, except for subject 6, the profiles with and without guar are similar.

◁

▷ Example 22: Using group() in the presence of random effects

The actual NLME model presented in Hand and Crowder (1996) for these glucose data included random effects for ϕ_1 and ϕ_2 only at the `subject` level and used a continuous-time AR(1) correlation structure on `time` for the `guar`-within-`subject` errors, with errors from different `guar`-within-`subject` clusters assumed to be independent. This model can be specified in `menl` using `rescorrelation()`'s `group()` suboption:

```
. menl glucose = {phi1:} + {phi2:}*c.time#c.time#c.time*exp(-{phi3:}*time),
>         define(phi1: i.guar U1[subject])
>         define(phi2: i.guar U2[subject])
>         define(phi3: i.guar, xb)
>         rescorrelation(ctar1, t(time) group(guar)) stddeviations
note: group variable guar nested in subject assumed.
```

```
Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood =  -183.7208
Iteration 2:  Linearization log likelihood = -183.91698
  (iteration log omitted)
Iteration 13: Linearization log likelihood = -183.90513
Iteration 14: Linearization log likelihood = -183.90511

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        196
      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
subject	7	28	28.0	28
guar	14	14	14.0	14

```
                                                Wald chi2(3)      =       1.01
Linearization log likelihood = -183.90511       Prob > chi2       =     0.7978

        phi1: i.guar U1[subject]
        phi2: i.guar U2[subject]
        phi3: i.guar
```

glucose	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
guar						
with guar	-.0557508	.1714288	-0.33	0.745	-.391745	.2802434
_cons	3.682235	.1503694	24.49	0.000	3.387517	3.976954
phi2						
guar						
with guar	.032163	.0721232	0.45	0.656	-.1091958	.1735219
_cons	.3349061	.0577129	5.80	0.000	.2217908	.4480214
phi3						
guar						
with guar	.0232717	.0346187	0.67	0.501	-.0445798	.0911232
_cons	.5464887	.0243374	22.45	0.000	.4987883	.5941891

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Independent				
sd(U1)	.2288441	.1103909	.0889065	.589042
sd(U2)	.0774363	.0306965	.0356058	.1684103
Residual: CTAR1, time time				
sd(e)	.6663828	.0439279	.5856156	.7582893
rho	.7018854	.0468263	.6101075	.7936633

The fixed-effects estimates are similar to those in example 21, and the same conclusion is reached regarding the effect of the dietary additive guar on the blood-glucose-levels profiles over time. AIC and BIC may be used to decide on which model is better.

Notice the note displayed by `menl` following the command specification about the group variable `guar` being nested within variable `subject`. When you specify `group(`*grpvar*`)` within the `rescorrelation()` (or `rescovariance()`) option in the presence of random effects, *grpvar* is assumed to represent the lowest level of hierarchy and is thus assumed to be nested within other hierarchical levels. ◁

▷ Example 23: Three-level model with block-diagonal covariance matrix

Pinheiro and Bates (2000) report the data from the experiment conducted by Microelectronics Division of Lucent Technologies to study the variability in the manufacturing of analog MOS circuits. The intensities of the `current` (in mA) were collected on n-channel devices at five ascending `voltages`: 0.8, 1.2, 1.6, 2.0, and 2.4 V. Measurements were made on 8 `sites` of each of 10 `wafers`. The main objective of the study was to build an empirical model to simulate the behavior of similar circuits.

The intensity of the current at the ith level of `voltage` in the jth `site` within the kth `wafer` is expressed as

$$\texttt{current}_{ijk} = \phi_{1jk} + \phi_{2jk}\cos\left(\phi_{3jk}\texttt{voltage}_i + \pi/4\right) + \epsilon_{ijk}$$

where

$$\phi_{1jk} = \beta_0 + u^{(3)}_{0k} + u^{(2)}_{0j,k} + \left(\beta_1 + u^{(3)}_{1k} + u^{(2)}_{1j,k}\right)\texttt{voltage}_i + \left(\beta_2 + u^{(3)}_{2k} + u^{(2)}_{2j,k}\right)\texttt{voltage}_i^2$$
$$\phi_{2jk} = \beta_3 + u^{(3)}_{3k} + u^{(2)}_{3j,k}$$
$$\phi_{3jk} = \beta_4 + u^{(3)}_{4k}$$

$$\mathbf{u}^{(3)}_k = \begin{bmatrix} u^{(3)}_{0k} \\ u^{(3)}_{1k} \\ u^{(3)}_{2k} \\ u^{(3)}_{3k} \\ u^{(3)}_{4k} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}_3) \qquad \mathbf{u}^{(2)}_{j,k} = \begin{bmatrix} u^{(2)}_{0j,k} \\ u^{(2)}_{1j,k} \\ u^{(2)}_{2j,k} \\ u^{(2)}_{3j,k} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}_2) \qquad \epsilon_{ijk} \sim N(0, \sigma^2_\epsilon)$$

Parameters β_0, β_1, and β_2 characterize the quadratic component of the model, and amplitude β_3 and frequency β_4 characterize the periodic component represented by the cosine wave.

For illustration, consider the following random-effects covariance structures:

$$\mathbf{\Sigma}_3 = \begin{bmatrix} \sigma^{(3)}_{11} & & & & \\ & \sigma^{(3)}_{22} & & & \\ & & \sigma^{(3)}_{33} & & \\ & & & \sigma^{(3)}_{44} & \\ & & & & \sigma^{(3)}_{55} \end{bmatrix} \qquad \mathbf{\Sigma}_2 = \begin{bmatrix} \sigma^{(2)}_{11} & \sigma^{(2)}_{12} & 0 & 0 \\ \sigma^{(2)}_{12} & \sigma^{(2)}_{22} & 0 & 0 \\ 0 & 0 & \sigma^{(2)}_{33} & \sigma^{(2)}_{34} \\ 0 & 0 & \sigma^{(2)}_{34} & \sigma^{(2)}_{44} \end{bmatrix}$$

If we were to fit this model by using `menl`, we would type

```
. use https://www.stata-press.com/data/r19/wafer
(Modeling of analog MOS circuits)
. menl current = {phi1:}+{phi2:}*cos({phi3:}*voltage + _pi/4),
> define(phi1: voltage c.voltage#c.voltage W0[wafer] S0[wafer>site]
>              c.voltage#(W1[wafer] S1[wafer>site])
>              c.voltage#c.voltage#(W2[wafer] S2[wafer>site]))
> define(phi2: W3[wafer] S3[wafer>site]) define(phi3: W4[wafer], xb)
> covariance(S0 S1, unstructured) covariance(S2 S3, unstructured)
> covariance(W*, independent) stddeviations
```

In the specification above, $\boldsymbol{\Sigma}_3$ is specified as `covariance(W*, independent)`, although this specification could have been omitted because `independent` is `menl`'s default random-effects covariance structure. The block-diagonal matrix $\boldsymbol{\Sigma}_2$ is specified by using repeated `covariance()` options: `covariance(S0 S1, unstructured)` and `covariance(S2 S3, unstructured)`. If we tried to run this model, we would find out that it is overparameterized.

Because of the large number of random effects at each grouping level, to avoid numerically unstable estimates, we will further simplify our model by assuming independence between $u^{(2)}_{2j,k}$ and $u^{(2)}_{3j,k}$, which implies that $\sigma^{(2)}_{34} = 0$:

$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} \sigma^{(3)}_{11} & & & & \\ & \sigma^{(3)}_{22} & & & \\ & & \sigma^{(3)}_{33} & & \\ & & & \sigma^{(3)}_{44} & \\ & & & & \sigma^{(3)}_{55} \end{bmatrix} \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix} \sigma^{(2)}_{11} & \sigma^{(2)}_{12} & 0 & 0 \\ \sigma^{(2)}_{12} & \sigma^{(2)}_{22} & 0 & 0 \\ 0 & 0 & \sigma^{(2)}_{33} & 0 \\ 0 & 0 & 0 & \sigma^{(2)}_{44} \end{bmatrix}$$

We now try to fit the above simpler model. Note that given the complexity of this model, it takes some time to execute.

```
. use https://www.stata-press.com/data/r19/wafer
(Modeling of analog MOS circuits)
. menl current = {phi1:}+{phi2:}*cos({phi3:}*voltage + _pi/4),
> define(phi1: voltage c.voltage#c.voltage W0[wafer] S0[wafer>site]
>              c.voltage#(W1[wafer] S1[wafer>site])
>              c.voltage#c.voltage#(W2[wafer] S2[wafer>site]))
> define(phi2: W3[wafer] S3[wafer>site]) define(phi3: W4[wafer], xb)
> covariance(S0 S1, unstructured) covariance(S2 S3, independent)
> covariance(W*, independent) stddeviations
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood =  735.58995
Iteration 2:  Linearization log likelihood =  766.83552
Iteration 3:  Linearization log likelihood =   825.9155
Iteration 4:  Linearization log likelihood =   825.9171
Iteration 5:  Linearization log likelihood =   825.9171
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        400
      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
wafer	10	40	40.0	40
wafer>site	80	5	5.0	5

```
                                                Wald chi2(2)      =    8763.94
Linearization log likelihood =   825.9171       Prob > chi2       =     0.0000
       phi1: voltage c.voltage#c.voltage W0[wafer] S0[wafer>site]
             c.voltage#W1[wafer] c.voltage#S1[wafer>site]
             c.voltage#c.voltage#W2[wafer]
             c.voltage#c.voltage#S2[wafer>site]
       phi2: W3[wafer] S3[wafer>site]
       phi3: W4[wafer], xb
```

current	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
voltage	6.046937	.1022632	59.13	0.000	5.846504	6.247369
c.voltage# c.voltage	1.158782	.0159669	72.57	0.000	1.127487	1.190076
_cons	-4.658034	.0361763	-128.76	0.000	-4.728938	-4.58713
phi2						
_cons	.1684428	.002054	82.01	0.000	.1644171	.1724686
phi3						
_cons	6.449391	.0019631	3285.32	0.000	6.445543	6.453238

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
wafer: Independent				
sd(W0)	.1107108	.0262518	.0695588	.1762088
sd(W1)	.3041975	.0764653	.1858623	.4978746
sd(W2)	.0449994	.0125441	.026057	.0777122
sd(W3)	.0057862	.0016144	.0033489	.0099974
sd(W4)	.0061349	.0013878	.0039377	.0095579
wafer>site: Unstructured				
sd(S0)	.0729495	.0062969	.0615953	.0863968
sd(S1)	.2930062	.0252423	.2474835	.3469024
corr(S0,S1)	-.8113227	.0413358	-.8782237	-.7132787
wafer>site: Independent				
sd(S2)	.0627587	.0053067	.0531738	.0740712
sd(S3)	.0080611	.0006861	.0068227	.0095244
sd(Residual)	.0008407	.0000711	.0007122	.0009922

In this example, our primary focus was to demonstrate how to use `menl` to fit a block-diagonal random-effects covariance structure. But if we were to interpret our fixed-effects estimates, the average frequency of the cosine wave, $\beta_4 = E(\phi_{3jk})$, for example, is estimated to be $6.45V^{-1}$, with a corresponding estimated period of $2\pi/\hat{\beta}_4 \approx 0.97V$. Also, some of the estimates of standard deviations such as `sd(W2)`, `sd(W3)`, and `sd(W4)` are very small, which suggests that this model may still be too rich for the observed data. If we proceeded to further analyze these data, we would consider simpler models. For example, at the very least, we would have omitted the term `W3[wafer]` from this model.

◁

Obtaining initial values

Obtaining good starting or initial values is important for the estimation of many statistical models, but it is often crucial for the estimation of NLME models. NLME models are known to be sensitive to the initial values and to have difficulty converging. Highly nonlinear mean specification or complicated variance–covariance structures for random effects and errors can often lead to multiple solutions, which requires considering different sets of initial values.

By default, `menl` uses the EM algorithm to obtain initial values. This default routine works well in many cases but cannot be guaranteed to provide good initial values in all situations. Sometimes, you may need to specify your own initial values. Trying different initial values can also be useful to investigate the existence of multiple solutions and to verify convergence to a global maximum.

So far we have been "lucky" that all the examples worked without us having to specify initial estimates. You may not be that lucky with your data and model. So, in this section, we provide some guidance on how to find good initial values when the default initial values do not work well.

We present three approaches that you may choose to explore to find good initial estimates for the fixed effects. In some cases, you may also be able to obtain initial estimates for covariance parameters; see *Linearization approach to finding initial values*.

Linearization approach to finding initial values

Sometimes, we can use an LME model to obtain initial values of the NLME model by holding some of the parameters fixed at specific values. We can then fit the resulting LME model by using the `mixed` command and use the corresponding estimates as initial values for the NLME model. We refer to this initialization method as the linearization method.

We could have used this method in example 14 and example 23, if the default EM method did not provide reasonable initial estimates. In any case, it is good practice to specify different initial values to investigate potential convergence of the algorithm to a local maximum.

For instance, in example 14, we fit

$$\texttt{follicles}_{ij} = \phi_{1j} + \phi_{2j}\sin\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \phi_{4j}\cos\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

where

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\\ \phi_{4j}\end{bmatrix} = \begin{bmatrix}\beta_1 + u_{1j}\\ \beta_2\\ \beta_3\\ \beta_4\end{bmatrix}$$

This model is nonlinear because of the parameter ϕ_{3j}. To obtain initial values, we can hold ϕ_{3j} (or β_3) fixed at a specific value, say, $\beta_3 = 1$, thus making the above model linear,

$$\texttt{follicles}_{ij} = \phi_{1j} + \phi_{2j}\sin\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \phi_{4j}\cos\left(2\pi\phi_{3j}\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

where

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\\ \phi_{4j}\end{bmatrix} = \begin{bmatrix}\beta_1 + u_{1j}\\ \beta_2\\ 1\\ \beta_4\end{bmatrix}$$

Or, more compactly,

$$\texttt{follicles}_{ij} = \beta_1 + u_{1j} + \beta_2\sin\left(2\pi\texttt{stime}_{ij}\right) + \beta_4\cos\left(2\pi\texttt{stime}_{ij}\right) + \epsilon_{ij}$$

Now that the model is linear, we can use the `mixed` command to obtain initial values for β_1, β_2, and β_4 to be used in `menl`. In the code below, variables `sin1` and `cos1` are $\sin\left(2\pi\texttt{stime}_{ij}\right)$ and $\cos\left(2\pi\texttt{stime}_{ij}\right)$, respectively, and `|| mare:` specifies a random intercept at the `mare` level (see [ME] **mixed**). Also, for consistency with example 13, we assume an AR(1) within-group error correlation structure:

```
. mixed follicles sin1 cos1 || mare:, residuals(ar 1, t(time)) nolog
Mixed-effects ML regression                     Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
                                                Wald chi2(2)      =      39.00
Log likelihood = -776.51731                     Prob > chi2       =     0.0000
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
sin1	-2.958619	.4935054	-6.00	0.000	-3.925872	-1.991366
cos1	-.8798847	.5031763	-1.75	0.080	-1.866092	.1063228
_cons	12.18963	.9017441	13.52	0.000	10.42224	13.95701

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
mare: Identity				
var(_cons)	7.095514	3.76488	2.508051	20.07388
Residual: AR(1)				
rho	.5974664	.0547217	.4795551	.6941854
var(e)	13.08097	1.765325	10.04078	17.0417

```
LR test vs. linear model: chi2(2) = 242.63                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We will now use the estimates of the fixed effects shown in the output table as initial values for menl by specifying the initial() option. We use 1 as the initial value for /phi3. There are three ways to specify initial values in the initial() option; see *Specifying initial values*. Here we will use the specification where we repeatedly list a parameter name followed by its initial value; also see *Examples of specifying initial values*.

```
. local xb phi1:_cons 12.2 /phi2 -3.0 /phi3 1 /phi4 -.88
. menl follicles = {phi1: U1[mare], xb} + {phi2}*sin(2*_pi*stime*{phi3}) +
> {phi4}*cos(2*_pi*stime*{phi3}), rescorrelation(ar 1, t(time)) init(`xb')
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -775.62937
Iteration 2:  Linearization log likelihood = -775.62433
Iteration 3:  Linearization log likelihood = -775.62433
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
Linearization log likelihood = -775.62433
        phi1: U1[mare], xb
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	12.18125	.9055128	13.45	0.000	10.40647	13.95602
/phi2	-2.874413	.5389583	-5.33	0.000	-3.930751	-1.818074
/phi3	.919114	.0512333	17.94	0.000	.8186986	1.019529
/phi4	-1.675314	.6766091	-2.48	0.013	-3.001444	-.3491848

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
mare: Identity				
var(U1)	7.207072	3.755603	2.595363	20.01334
Residual: AR(1), time time				
var(e)	12.63377	1.646897	9.785277	16.31146
rho	.5823733	.0544508	.4656903	.679153

In the above, we initialized only fixed-effects parameters and used naïve initial estimates of 1 for random-intercept and error variances and 0 for the correlation. We could have specified initial()'s fixed suboption to use the EM algorithm to compute initial estimates for the random-effects parameters; see *Examples of specifying initial values* for details.

With the linearization approach, we can also use estimates of the random-effects parameters from the mixed command to initialize the corresponding parameters of menl. This is an advantage of the linearization approach over the other two approaches we discuss in subsequent sections. One complication with the initialization of random-effects parameters is that the initial values must be supplied in the estimation metric, the metric used during estimation, instead of the parameter original metric. For example, instead of variances, we must supply estimates of log standard-deviations, and instead of co-

variances or correlations, we must supply inverse hyperbolic tangents of correlation parameters. Luckily for us, `mixed` stores results using the same metric as `menl` and provides the `estmetric` option to display parameters in that metric.

In our example, the random-effects parameters are the random-intercept variance, the within-group error variance, and the correlation between error terms. We refit the earlier `mixed` command but now with the `estmetric` option to obtain the estimates of the random-effects parameters as they are stored in `e(b)`.

```
. mixed follicles sin1 cos1 || mare:, residuals(ar 1, t(time)) nolog estmetric
Mixed-effects ML regression                     Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
                                                Wald chi2(2)      =      39.00
Log likelihood = -776.51731                     Prob > chi2       =     0.0000
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
follicles						
sin1	-2.958619	.4935054	-6.00	0.000	-3.925872	-1.991366
cos1	-.8798847	.5031763	-1.75	0.080	-1.866092	.1063228
_cons	12.18963	.9017441	13.52	0.000	10.42224	13.95701
lns1_1_1						
_cons	.9797314	.2653			.5762507	1.665722
lnsig_e						
_cons	1.285579	.0674768	19.05	0.000	1.153327	1.417832
r_atr1						
_cons	.6891978	.0850992	8.10	0.000	.5224064	.8559891

`menl` uses the same ordering of the parameters as `mixed` does, so we can simply list all the estimates directly in the `initial()` option. When we list the values without parameter names, we must specify `initial()`'s `copy` suboption and specify the values for all parameters. In our example, we specify four fixed-effects coefficients and three random-effects parameters.

```
. menl follicles = {phi1: U1[mare], xb} + {phi2}*sin(2*_pi*stime*{phi3}) +
> {phi4}*cos(2*_pi*stime*{phi3}), rescorrelation(ar 1, t(time))
> initial(12.2 -3.0 1 -.88 .98 1.29 .69, copy)

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -775.62433
Iteration 2:  Linearization log likelihood = -775.62433

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31

Linearization log likelihood = -775.62433
        phi1: U1[mare], xb
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	12.18125	.9055135	13.45	0.000	10.40647	13.95602
/phi2	-2.874434	.5389241	-5.33	0.000	-3.930706	-1.818162
/phi3	.919119	.0512356	17.94	0.000	.818699	1.019539
/phi4	-1.675261	.6766409	-2.48	0.013	-3.001452	-.3490689

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
mare: Identity				
var(U1)	7.207072	3.755605	2.595361	20.01336
Residual: AR(1), time time				
var(e)	12.63377	1.646897	9.785276	16.31146
rho	.5823733	.0544508	.4656903	.679153

The results are different from those in example 14. The value of the linearization log likelihood in this example, −775.62, is larger than that from example 14, −789.43. So it appears that we have converged to a local maximum of the linearization log likelihood in example 14.

Our initial values based on `mixed` turned out to be better than those computed by default by `menl`. This is not surprising. In general, `menl`'s EM algorithm should produce reasonable initial values for many nonlinear models, but the initial values may not necessarily be optimal for all of those models. In this example, our initial values were tailored to the ovary data and the model.

In general, sensitivity to initial values is one of the key issues in NLME models, especially for models that involve periodic functions. Therefore, it is important to try different sets of initial values to verify global convergence before reporting your final results. Sometimes, you may even have to rely on your knowledge of the science behind the problem to decide which set of results is more reasonable.

Graphical approach to finding initial values

If your model has parameters that have natural physical interpretations, you may be able to obtain starting values from a graph of the data.

Draper and Smith (1998) presented a dataset in which the trunk circumference `circumf` (in mm) of five different orange trees was measured over seven different time points, stored in `age`. Pinheiro and Bates (2000) suggested the following model for these data:

$$\texttt{circumf}_{ij} = \frac{\phi_{1j}}{1 + \exp\left\{-\left(\texttt{age}_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij} \tag{24}$$

In this model, ϕ_{1j} is the asymptotic trunk circumference for the jth tree as $\texttt{age}_{ij} \to \infty$, ϕ_{2j} is the age at which the jth tree attains half of its asymptotic trunk circumference ϕ_{1j}, and ϕ_{3j} is a scale parameter; see the graph below.

The stage 2 specification of this model is

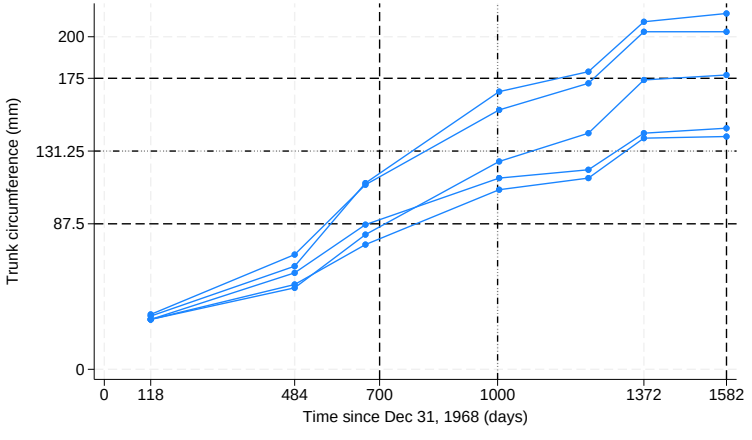
$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\end{bmatrix} = \begin{bmatrix}\beta_1 + u_{1j}\\ \beta_2\\ \beta_3\end{bmatrix}$$

where

$$u_{1j} \sim N\left(0, \sigma^2_{u_1}\right),\ \epsilon_{ij} \sim N\left(0, \sigma^2_{\epsilon}\right)$$

Because the model parameters have graphical interpretations, we can plot our data and obtain initial values from the graph.

```
. use https://www.stata-press.com/data/r19/orange
(Growth of orange trees (Draper and Smith, 1998))
. twoway connected circumf age, connect(L) yline(175) xline(1582)
> yline(87.5, lpattern(dash)) xline(700, lpattern(dash))
> yline(131.25, lpattern("-...")) xline(1000, lpattern("-..."))
> xlabel(0 118 484 700 1000 1372 1582) ylabel(#5 87.5 131.25 175)
```



From the above graph, the mean asymptotic trunk circumference can be estimated as 175 mm, which is roughly the mean of the circumference values at age 1,582 (in days). The trees attain half of their asymptotic trunk circumference, $175/2 = 87.5$, at about age 700 (in days). Therefore, we use the initial estimates $\beta_1 = 175$ for the asymptotic trunk circumference and $\beta_2 = 700$ for the location of the inflection point. To obtain an initial estimate for β_3, we note that when $\texttt{age} = \beta_2 + \beta_3$ in (24), $E(\texttt{circumf}_{ij}) = \beta_1/\{1 + \exp(-1)\} = 0.73\beta_1$, which we will approximate as $0.75\beta_1$ for the purpose of the graph.

That is, the logistic curve reaches approximately 3/4 of its asymptotic value, $0.75 \times 175 = 131.25$, at age $= \beta_2 + \beta_3$. The above graph suggests that the trees attain 3/4 of their final trunk circumference at about 1,000 days ($= \beta_2 + \beta_3$), giving an initial estimate of $\beta_3 = 1000 - 700 = 300$. We can now supply these values to `menl` in the `initial()` option.

Unfortunately, the graph does not provide us with the estimates for variance components. In this case, we can use `initial()`'s `fixed` suboption to specify that the EM algorithm still be used to initialize variance components, while the supplied values be used to initialize fixed effects. If we do not specify `fixed`, `menl` will use naïve initial estimates for variance components such as ones for variances and zeros for covariances.

We now fit the model using our own initial estimates for fixed effects:

```
. menl circumf = {phi1: U1[tree], xb}/(1+exp(-(age-{phi2})/{phi3})),
> initial(phi1:_cons 175 /phi2 700 /phi3 300, fixed)

Obtaining starting values by EM:

Alternating PNLS/LME algorithm:

Iteration 1:  Linearization log likelihood = -131.58494
Iteration 2:  Linearization log likelihood = -131.58458
Iteration 3:  Linearization log likelihood = -131.58458

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7

Linearization log likelihood = -131.58458

       phi1: U1[tree], xb
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	191.049	16.15403	11.83	0.000	159.3877	222.7103
/phi2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/phi3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Identity				
var(U1)	991.1514	639.4637	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

For comparison, we fit the same model but now using the default initial values for fixed effects:

```
. menl circumf = {phi1: U1[tree], xb}/(1+exp(-(age-{phi2})/{phi3}))
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -131.58458
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =         35
Group variable: tree                            Number of groups  =          5
                                                Obs per group:
                                                              min =          7
                                                              avg =        7.0
                                                              max =          7
Linearization log likelihood = -131.58458
        phi1: U1[tree], xb
```

circumf	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
_cons	191.049	16.15403	11.83	0.000	159.3877	222.7103
/phi2	722.556	35.15082	20.56	0.000	653.6616	791.4503
/phi3	344.1624	27.14739	12.68	0.000	290.9545	397.3703

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
tree: Identity				
var(U1)	991.1514	639.4637	279.8776	3510.038
var(Residual)	61.56371	15.89568	37.11466	102.1184

The results are identical except for the iteration log.

Smart regressions approach to finding initial values

Consider the following NLME model,

$$\mathrm{y}_{ij} = \phi_{1j} + \left(\phi_{2j} - \phi_{1j}\right) \exp\left\{-\exp\left(\phi_{3j}\right) \mathrm{x}_{ij}\right\} + \epsilon_{ij}$$

where

$$\boldsymbol{\phi}_j = \begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \phi_{3j} \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 + u_{1j} \\ \beta_3 \end{bmatrix}$$

Here ϕ_{1j} is the asymptote as $\mathrm{x}_{ij} \to \infty$ and ϕ_{2j} is the value of y_{ij} at $\mathrm{x}_{ij} = 0$. Thus initial estimates, $\beta_1^{(0)}$ and $\beta_2^{(0)}$, may be obtained by using the graphical approach as described in *Graphical approach to finding initial values*. To obtain an initial estimate for β_3, notice that, ignoring the error term ϵ_{ij} and setting $u_{1j} = 0$,

$$\log\left(\left|\mathrm{y}_{ij} - \beta_1\right|\right) = \log\left(\beta_2 - \beta_1\right) + \left\{-\exp\left(\beta_3\right)\right\} \mathrm{x}_{ij}$$

Therefore, we can regress $\log(|\mathtt{y} - \beta_1^{(0)}|)$ on x and use the estimated slope, $\hat{\beta}_x = -\exp(\beta_3^{(0)})$, to obtain the initial value for $\beta_3^{(0)} = \log(-\hat{\beta}_x)$.

Examples of specifying initial values

When you want to assign initial values for a subset of the model parameters, for example, fixed effects or random-effects covariance parameters, you will often need to know their estimation names or, in other words, how `menl` labels them in `e(b)`. To learn the names, you can fit the model with the `iterate(0)` and `coeflegend` options first.

```
. menl ..., ... iterate(0) coeflegend
```

The `iterate(0)` option specifies to bypass maximization and only report the initial values and the likelihood evaluated at those values. The `coeflegend` option specifies that the legend of the parameters and how to specify them in an expression be displayed rather than displaying the statistics for the parameters.

Keep in mind, however, that `menl` does not perform estimation in the original parameter metric. For computational stability, the estimation is performed, loosely speaking, in a metric that transforms all parameters to be defined on a real line. For example, a log transformation is used for standard deviations, and an inverse hyperbolic tangent transformation is used for correlations. When you specify initial values, you must specify them for parameters in the estimation metric and not the original metric.

`coeflegend` displays parameter names as they are stored in `e(b)`, which, for `menl`, are the names of estimation parameters. If you also want to see parameters in the original metric, you can specify `coeflegend` on replay.

```
. menl ..., ... iterate(0)
. menl, coeflegend
```

For example, recall the NLME model for the soybean data from example 9. Suppose that we want to supply our own initial values.

We fit the model with `iterate(0)` and `coeflegend`:

```
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: U1[plot], xb)
> define(phi2: U2[plot], xb)
> define(phi3: U3[plot], xb)
> covariance(U*, unstructured) iterate(0) coeflegend

Obtaining starting values by EM:

Computing standard errors:

Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10

Linearization log likelihood = -740.06177

        phi1: U1[plot], xb
        phi2: U2[plot], xb
        phi3: U3[plot], xb
```

weight	Coefficient	Legend
phi1		
_cons	19.26527	_b[phi1:_cons]
phi2		
_cons	55.05299	_b[phi2:_cons]
phi3		
_cons	8.385531	_b[phi3:_cons]
/plot		
lnsd(U1)	1.650846	_b[/plot:lnsd(U1)]
lnsd(U2)	1.436634	_b[/plot:lnsd(U2)]
lnsd(U3)	.4081525	_b[/plot:lnsd(U3)]
athcorr(U2, U1)	.9055785	_b[/plot:athcorr(U2,U1)]
athcorr(U3, U1)	.8482105	_b[/plot:athcorr(U3,U1)]
athcorr(U3, U2)	1.537798	_b[/plot:athcorr(U3,U2)]
/Residual		
lnsigma	.1069986	_b[/Residual:lnsigma]

```
Warning: Convergence not achieved.
```

Parameter names are listed within the `_b[]` specifier.

In what follows, we will outline only the syntax of the specifications. If you actually want to run all the examples to see the initialization in action, we suggest that you specify `iterate(0)` for speed.

Let's first specify initial values for fixed effects only. The fixed-effects parameters are `phi1:_cons`, `phi2:_cons`, and `phi3:_cons`. Suppose that we want to initialize them with 19, 55, and 8.

We can type

```
. menl ..., ... initial(phi1:_cons 19 phi2:_cons 55 phi3:_cons 8)
```

Or, more compactly, we can type

```
. local fe phi1:_cons 19 phi2:_cons 55 phi3:_cons 8
. menl ..., ... initial('fe')
```

When you specify the `initial()` option, `menl` does not perform the EM algorithm to initialize the parameters but instead uses the values you supplied. If you specify values for only a subset of parameters, the remaining parameters will be initialized with naïve initial values such as zeros for fixed effects and correlations and ones for variances. Often, you may have good initial values for fixed effects but not for variance components. In this situation, `menl` provides `initial()`'s `fixed` suboption. This option specifies that the supplied values be used for fixed effects but that the EM algorithm still be used to obtain initial values for variance components. If you specify only a subset of values for fixed effects, the remaining fixed effects will still be initialized with zeros even if `fixed` is specified. We recommend that you specify `fixed` when you intend to supply initial values only for the fixed effects.

```
. local fe phi1:_cons 19 phi2:_cons 55 phi3:_cons 8
. menl ..., ... initial('fe', fixed)
```

Now suppose that we also want to assign initial values for random-effects parameters. As we mentioned earlier, remember that we assign initial values for standard deviations in the log metric and for correlation in the inverse hyperbolic tangent or atanh metric. For example, if you want to assign an initial value of 2 to σ_ϵ, then you should supply $\log(2)$ to the `initial()` option. Similarly, if you want to assign a value of 0.7 to the correlation of two random effects, then you should provide $\texttt{atanh}(0.7)$ to the `initial()` option.

Continuing with example 9, suppose that we want to specify the following initial values for the random-effects covariance parameters:

$$\begin{array}{c} \texttt{U1[plot]} \quad \texttt{U2[plot]} \quad \texttt{U3[plot]} \\ \begin{pmatrix} \sigma_1 = 5 & & \\ \rho_{21} = 0.72 & \sigma_2 = 4 & \\ \rho_{31} = 0.71 & \rho_{32} = 0.94 & \sigma_3 = 1.4 \end{pmatrix} \end{array}$$

The names of the parameters in the estimation metric that correspond to σ_1, σ_2, and σ_3 are `/plot:lnsd(U1)`, `/plot:lnsd(U2)`, and `/plot:lnsd(U3)` and that correspond to ρ_{21}, ρ_{31}, and ρ_{32} are `/plot:athcorr(U2,U1)`, `/plot:athcorr(U3,U1)`, and `/plot:athcorr(U3,U2)`.

When specifying initial values for free parameters such as random-effects covariance parameters, you can omit the forward slash (`/`) at the beginning of their names. Keeping in mind that initial values for covariance parameters are supplied in the log and atanh metrics, we can type

```
. local re_cov          plot:lnsd(U1) log(5)            // log(5)
. local re_cov 're_cov' plot:lnsd(U2) 1.4               // log(4)
. local re_cov 're_cov' plot:lnsd(U3) 0.34              // log(1.4)
. local re_cov 're_cov' plot:athcorr(U2,U1) atanh(0.72) // atanh(0.72)
. local re_cov 're_cov' plot:athcorr(U3,U1) 0.89        // atanh(0.71)
. local re_cov 're_cov' plot:athcorr(U3,U2) 1.7         // atanh(0.94)
. menl ..., ... initial('fe' 're_cov' Residual:lnsigma 0.5)
```

In the above, we also specified an initial value of 0.5 for the log of the error standard deviation. For parameters `/plot:lnsd(U1)` and `/plot:athcorr(U2,U1)`, instead of specifying the values, we specified the corresponding expression. This is allowed, as long as your expression is simple and does not contain spaces.

Instead of using parameter names, we can specify a list of values directly in the initial() option, in which case we must also specify initial()'s copy suboption.

```
. menl ..., ... initial(19 55 8 1.6 1.4 0.34 0.9 0.89 1.7 0.5, copy)
```

Or we can provide these values as a matrix:

```
. matrix initvals = (19, 55, 8, 1.6, 1.4, 0.34, 0.9, 0.89, 1.7, 0.5)
. matrix list initvals
initvals[1,10]
     c1   c2   c3   c4   c5   c6   c7   c8   c9  c10
r1   19   55    8  1.6  1.4  .34   .9  .89  1.7   .5
. menl ..., ... initial(initvals, copy)
```

If we label the columns of the initvals matrix properly, we do not need to specify copy:

```
. local fullcolnames : colfullnames e(b)
. matrix colnames initvals = `fullcolnames'
. matrix list initvals
initvals[1,10]
           phi1:        phi2:        phi3:       /plot:       /plot:

          _cons        _cons        _cons      lnsd(U1)     lnsd(U2)
r1           19           55            8           1.6          1.4
          /plot:       /plot:       /plot:       /plot:   /Residual:
                  athcorr(U2,  athcorr(U3,  athcorr(U3,
        lnsd(U3)          U1)          U1)          U2)     lnsigma
r1           .34           .9          .89          1.7           .5
. menl ..., ... initial(initvals)
```

Using a properly labeled initial-value matrix, we can also specify initial values for a subset of parameters. For example, we can specify initial values for fixed effects only as follows:

```
. matrix initvals = initvals[1,1..3]
. matrix list initvals
initvals[1,3]
     phi1:  phi2:  phi3:
    _cons  _cons  _cons
r1     19     55      8
. menl ... ... initial(initvals)
```

Stored results

menl stores the following in e():

Scalars

e(N)	number of observations
e(N_nonmiss)	number of nonmissing *depvar* observations, if tsmissing is specified
e(N_miss)	number of missing *depvar* observations, if tsmissing is specified
e(N_ic)	number of nonmissing *depvar* observations to be used for BIC computation when tsmissing is specified
e(k)	number of parameters
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances

e(k_rc)	number of covariances
e(k_res)	number of within-group error parameters
e(k_eq)	number of equations
e(k_feq)	number of fixed-effects equations
e(k_req)	number of random-effects equations
e(k_reseq)	number of within-group error equations
e(df_m)	model degrees of freedom
e(df_c)	degrees of freedom for comparison test
e(ll)	linearization log (restricted) likelihood
e(ll_c)	log likelihood, comparison model
e(chi2)	χ^2
e(chi2_c)	χ^2 for comparison test
e(p)	p-value for model test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	menl
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(ivars)	grouping variables
e(title)	title in estimation output
e(varlist)	variables used in the specified equation
e(key_N_ic)	nonmissing obs, if tsmissing is specified
e(tsmissing)	tsmissing, if specified
e(tsorder)	tsorder() specification
e(eq_*depvar*)	user-specified equation
e(tsinit_*depvar*)	tsinit() specification for L.{*depvar*:}
e(expressions)	names of defined expressions, *expr_1*, *expr_2*, …, *expr_k*
e(expr_*expr_i*)	defined expression *expr_i*, $i = 1, \ldots, k$
e(tsinit_*expr*)	tsinit() specification for L.{*expr*:}
e(hierarchy)	random-effects hierarchy structure, (*path*:*covtype*:*REs*) (...)
e(revars)	names of random effects
e(rstructlab)	within-group error covariance output label
e(timevar)	within-group error covariance t() variable, if specified
e(indexvar)	within-group error covariance index() variable, if specified
e(covbyvar)	within-group error covariance by() variable, if specified
e(stratavar)	within-group error variance strata() variable, if specified
e(corrbyvar)	within-group error correlation by() variable, if specified
e(rescovopt)	within-group error covariance option, if rescovariance() specified
e(resvaropt)	within-group error variance option, if resvariance() specified
e(rescorropt)	within-group error correlation option, if rescorrelation() specified
e(groupvar)	lowest-level group() variable, if specified
e(chi2type)	Wald; type of model χ^2 test
e(vce)	conventional
e(method)	MLE or REML
e(opt)	type of optimization, lbates
e(crittype)	optimization criterion
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(marginsdefault)	default predict() specification for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	factor-variable constraint matrix
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(b_sd)	random-effects and within-group error estimates in the standard deviation metric
e(V_sd)	VCE for parameters in the standard deviation metric
e(b_var)	random-effects and within-group error estimates in the variance metric
e(V_var)	VCE for parameters in the variance metric
e(cov_#)	random-effects covariance structure at the hierarchical level $k - \# + 1$ in a k-level model
e(hierstats)	group-size statistics for each hierarchy

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction
Variance-components parameters
Inference based on linearization
Initial values

Introduction

Recall (1), a two-level NLME model, from the *Introduction*,

$$y_{ij} = \mu\left(\mathbf{x}'_{ij},\, \boldsymbol{\beta},\, \mathbf{u}_j\right) + \epsilon_{ij} \quad i = 1, \ldots, n_j;\ j = 1, \ldots, M$$

where M is the number of clusters and, for each cluster j, n_j is the number of observations in that cluster; $\mathbf{y}_j = (y_{1j}, y_{2j}, \ldots, y_{n_j j})'$ is the $n_j \times 1$ response vector; $\mathbf{X}_j = (\mathbf{x}_{1j}, \mathbf{x}_{2j}, \ldots, \mathbf{x}_{n_j j})'$ is the $n_j \times l$ matrix of covariates, including within-subject and between-subjects covariates; $\boldsymbol{\beta}$ is the $p \times 1$ vector of unknown parameters; $\mathbf{u}_j$ is the $q \times 1$ vector of random effects; and $\boldsymbol{\epsilon}_j = (\epsilon_{1j}, \epsilon_{2j}, \ldots, \epsilon_{n_j j})'$ is the $n_j \times 1$ vector of within-group or within-cluster errors. $\mathbf{u}_j$'s follow a multivariate normal distribution with mean 0 and $q \times q$ variance–covariance matrix $\boldsymbol{\Sigma}$, and $\boldsymbol{\epsilon}_j$'s follow a multivariate normal distribution with mean 0 and $n_j \times n_j$ variance–covariance matrix $\sigma^2\boldsymbol{\Lambda}_j$; $\mathbf{u}_j$'s are assumed to be independent of $\boldsymbol{\epsilon}_j$'s. Depending on the form of $\boldsymbol{\Lambda}_j$, σ^2 is either a within-group error variance σ^2_ϵ or a squared scale parameter σ^2. For example, when errors are i.i.d., that is, when $\boldsymbol{\Lambda}_j$ is the identity matrix, $\sigma^2 = \sigma^2_\epsilon$ is the within-group error variance. When $\boldsymbol{\Lambda}_j$ corresponds to the heteroskedastic power structure, σ^2 is a multiplier or a scale parameter.

Positive-definite matrices $\boldsymbol{\Sigma}/\sigma^2$ and $\boldsymbol{\Lambda}_j$ are expressed as functions of unconstrained parameter vectors $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_w$, respectively, to recast a constrained optimization problem into an unconstrained one. Thus $\boldsymbol{\alpha}_u$ contains unconstrained random-effects covariance parameters and $\boldsymbol{\alpha}_w$ contains unconstrained within-group error covariance parameters. $\boldsymbol{\Lambda}_j$ may also depend on the random effects $\mathbf{u}_j$ and the fixed effects $\boldsymbol{\beta}$. For more details about $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}_j$ and about functional forms of parameter vectors $\boldsymbol{\alpha}_u$ and $\boldsymbol{\alpha}_w$ given different covariance structures, see *Variance-components parameters*.

Based on (1), the marginal, with respect to $\mathbf{u}_j$'s, log likelihood for $(\boldsymbol{\beta}, \boldsymbol{\alpha}, \sigma^2)$ is

$$L(\boldsymbol{\beta}, \boldsymbol{\alpha}, \sigma^2) = \log\left\{\prod_{j=1}^{M}\int f\left(\mathbf{y}_j|\mathbf{X}_j, \mathbf{u}_j; \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \sigma^2\right) f\left(\mathbf{u}_j; \boldsymbol{\alpha}_u\right) d\mathbf{u}_j\right\} \tag{25}$$

where $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_u', \boldsymbol{\alpha}_w')'$, $f\left(\mathbf{y}_j|\mathbf{X}_j, \mathbf{u}_j; \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \sigma^2\right)$ is the conditional density of $\mathbf{y}_j$ given $\mathbf{X}_j$ and $\mathbf{u}_j$, and $f\left(\mathbf{u}_j; \boldsymbol{\alpha}_u\right)$ is the density of $\mathbf{u}_j$.

In general, there are no closed-form expressions for (25) or the marginal moments of an NLME model. This is because the random effects $\mathbf{u}_j$ enter the model nonlinearly, making the q-dimensional integral in (25) analytically intractable in all but simpler cases. Several estimation techniques have been proposed for estimating parameters $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$, and σ^2, including numerical integration of the integral in (25) by using an adaptive Gaussian quadrature and a linearization of the mean function in (1) by using a Taylor-series expansion.

`menl` implements the linearization method of Lindstrom and Bates (1990), with extensions from Pinheiro and Bates (1995), which is described in *Inference based on linearization*.

Variance-components parameters

For numerical stability, maximization of (25) is performed with respect to the unique elements of the matrix $\boldsymbol{G} = \boldsymbol{\Sigma}/\sigma^2$ expressed as logarithms of standard deviations for the diagonal elements and hyperbolic arctangents of the correlations for off-diagonal elements. Let $\boldsymbol{\alpha}_u$ be the vector containing these elements. For example, if we assume that the elements of the random-effects vector $\mathbf{u}_j$ are independent, then $\boldsymbol{\Sigma}$ is diagonal and $\boldsymbol{\alpha}_u$ will contain q distinct parameters—q logarithms of standard deviations. Table 1 lists the vectors of parameters $\boldsymbol{\alpha}_u$ for all random-effects covariance structures supported by `menl` in the `covariance(`*vartype*`)` option.

Table 1. Variance-components parameters

vartype	$\boldsymbol{\alpha}_u'$
`independent`	$(g_1, g_2, \ldots, g_q)$
`exchangeable`	(g_1, g_{12})
`identity`	g_1
`unstructured`	$(g_1, g_2, \ldots, g_q, g_{12}, g_{13}, \ldots, g_{q-1q})$

Notes: $g_u = \log(\sqrt{[\mathbf{G}]_{uu}})$, $g_{uv} =$ `atanh`$([\mathbf{G}]_{uv})$.
`unstructured` has $q(q+1)/2$ parameters.

The within-group error covariance matrix is parameterized as follows,

$$\mathrm{Var}\left(\boldsymbol{\epsilon}_j|\mathbf{u}_j\right) = \sigma^2\boldsymbol{\Lambda}_j\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j, \boldsymbol{\alpha}_w\right) = \sigma^2\mathbf{S}_j\left(\boldsymbol{\delta}, \boldsymbol{\upsilon}_j\right)\mathbf{C}_j(\boldsymbol{\rho})\mathbf{S}_j\left(\boldsymbol{\delta}, \boldsymbol{\upsilon}_j\right)$$

where $\boldsymbol{\alpha}_w = \left(\boldsymbol{\delta}^{*\prime}, \boldsymbol{\rho}^{*\prime}\right)'$ and $\boldsymbol{\delta}^*$ and $\boldsymbol{\rho}^*$ are unconstrained versions of $\boldsymbol{\delta}$ and $\boldsymbol{\rho}$ defined in table 2 and table 3, respectively. For example, for a positive δ_1, $\delta_1^* = \log(\delta_1)$. $\mathbf{S}_j = \mathbf{S}_j\left(\boldsymbol{\delta}, \boldsymbol{\upsilon}_j\right)$ is an $n_j \times n_j$ diagonal matrix with nonnegative diagonal elements $g\left(\boldsymbol{\delta}, \upsilon_{1j}\right), g\left(\boldsymbol{\delta}, \upsilon_{2j}\right), \ldots, g\left(\boldsymbol{\delta}, \upsilon_{n_jj}\right)$ such that $\mathrm{Var}\left(\epsilon_{ij}\right) = \sigma^2[\mathbf{S}_j]_{ii}^2 = \sigma^2 g^2\left(\boldsymbol{\delta}, \upsilon_{ij}\right)$, where υ_{ij}'s are the values of a variance covariate or the values of a mean function $\mu\left(\mathbf{x}_{ij}', \boldsymbol{\beta}, \mathbf{u}_j\right)$, in which case $\boldsymbol{\Lambda}_j$ will depend on $\mathbf{X}_j$, $\boldsymbol{\beta}$, and $\mathbf{u}_j$. $\mathbf{C}_j = \mathbf{C}_j(\boldsymbol{\rho})$ is a correlation

matrix such that corr $(\epsilon_{ij}, \epsilon_{kj}) = [\mathbf{C}_j]_{ik} = h(|t_{ij} - t_{lj}|, \boldsymbol{\rho})$, where t_{ij} is a value of a time variable for time-dependent correlation structures such as AR, MA, and Toeplitz structures or an index variable for `banded` and `unstructured` correlation structures. A list of the supported $g(\cdot)$ and $h(\cdot)$ functions is given in table 2 and table 3, respectively.

Carroll and Ruppert (1988) introduced various variance functions $g(\boldsymbol{\delta}, v_{ij})$ to model heteroskedasticity, which were further studied in the context of NLME models by Davidian and Giltinan (1995). Table 2 lists variance functions supported by the `resvariance(`*resvarfunc* `...)` option.

Table 2. Supported variance functions $g(\cdot)$

resvarfunc	$g(\boldsymbol{\delta}, v_{ij})$	$\boldsymbol{\delta}'$
`identity`	1	–
`linear`	$\sqrt{v_{ij}}$	–
`power`	$c + \lvert v_{ij}\rvert^{\delta}$	(c, δ), $c \geq 0$
`power, noconstant`	$\lvert v_{ij}\rvert^{\delta}$	δ
`exponential`	$\exp(\delta v_{ij})$	δ
`distinct`	$\sum_{l=1}^{L} \delta_l I(v_{ij} = l)$	$(\delta_1 = 1, \delta_2, \ldots, \delta_L)$

In table 2, the variance function `distinct` models a distinct parameter δ_l for each level l ($l = 1, 2, \ldots, L$) of the index variable $v_{ij} \in \{1, 2, \ldots, L\}$ such that for $v_{ij} = l$, Var$(\epsilon_{ij}) = \sigma_l^2 = \sigma^2\delta_l^2$, where $\delta_1 = 1$ for identifiability purposes and $\delta_l = \sigma_l/\sigma$. `menl` estimates and stores results as δ's but displays results as variances σ_l^2, $l = 1, \ldots, L$.

The variance function $g(\cdot)$ and thus the within-group error covariance may depend on $\boldsymbol{\beta}$ and $\mathbf{u}_j$ through $\mu(\cdot)$, when $v_{ij} = \mu_{ij} = \mu(\mathbf{x}'_{ij}, \mathbf{u}_j, \boldsymbol{\beta})$ in table 2. This is particularly appealing in PK applications, where there is often considerable intraindividual heterogeneity that is modeled, for example, as a power function of the mean.

The within-group error correlation structure is governed by the $h(\cdot)$ function. Table 3 lists correlation structures that are supported by the `rescorrelation(`*rescorr* `...)` option and also have a closed-form expression. In addition, the AR and MA correlation structures are defined below.

The `ar` p structure assumes that the errors have an AR structure of order p. That is,

$$\epsilon_{ij} = \phi_1\epsilon_{i-1,j} + \cdots + \phi_p\epsilon_{i-p,j} + z_{ij}$$

where z_{ij} are i.i.d. Gaussian with mean 0 and variance σ_z^2. `menl` reports estimates of $\phi_1, \ldots, \phi_p$ and the overall error variance σ_ϵ^2, which can be derived from the above expression. This structure has a closed-form expression only for $p = 1$, in which case $\phi_1 = \rho$ is the correlation between error terms.

The `ma` q structure assumes that the errors are an MA process of order q. That is,

$$\epsilon_{ij} = Z_i + \theta_1 Z_{i-1} + \cdots + \theta_q Z_{i-q}$$

where Z_l are i.i.d. Gaussian with mean 0 and variance σ^2_Z. `menl` reports estimates of $\theta_1, \ldots, \theta_q$ and the overall error variance σ^2_ϵ, which can be derived from the above expression.

Table 3. Within-group error correlation functions $h(\cdot)$

rescorr	$h(\lvert t_{ij} - t_{lj}\rvert, \boldsymbol{\rho})$	Expression	$\boldsymbol{\rho}$
`identity`	$h(k)$	$I(k=0)$	–
`exchangeable`	$h(k,\rho)$	$\rho,\ k = 1,2,\ldots$	$\rho, \lvert\rho\rvert < 1$
`ar 1`	$h(k,\rho)$	$\rho^k,\ k = 0,1,\ldots$	$\rho, \lvert\rho\rvert < 1$
`ar` p, $p > 1$	$h(k,\boldsymbol{\phi})$	no closed form	$(\phi_1, \phi_2, \ldots, \phi_p)$
`ctar1`	$h(s,\rho)$	$\rho^s,\ s \geq 0$	$\rho,\ \lvert\rho\rvert < 1$
`ma` q	$h(k,\boldsymbol{\theta})$	$\begin{cases} \frac{\sum_{j=0}^{q-\lvert k\rvert} \theta_j \theta_{j+\lvert k\rvert}}{\sum_{j=0}^{q} \theta_j^2} & k \leq q \\ 0 & k > q \end{cases}$	$(\theta_0 = 1, \theta_1, \ldots, \theta_q)$
`toeplitz`	$h(k,\boldsymbol{\rho})$	$\rho_k I(k \leq q),\ k = 1,2,\ldots,q$	$(\rho_1, \rho_2, \ldots, \rho_q)$
`banded`	$h(\lvert i-l\rvert, \boldsymbol{\rho})$	$\rho_{il} I(\lvert i-l\rvert \leq q),\ 1 \leq i < l \leq n_j$	$\{\rho_{il}\colon 0 < l - i \leq q\}$
`unstructured`	$h(\lvert i-l\rvert, \boldsymbol{\rho})$	$\rho_{il},\ 1 \leq i < l \leq n_j$	$\left(\rho_{12}, \ldots, \rho_{(n_j-1)n_j}\right)$

You can build many flexible within-group error covariance structures by combining different functions $g(\cdot)$ and $h(\cdot)$, that is, by combining the `resvariance()` and `rescorrelation()` options. For example, you can combine an AR(1) correlation structure with a heteroskedastic structure that is expressed as a power function of the mean by specifying `rescorrelation(ar 1, t(timevar))` and `resvariance(power _yhat)`.

Inference based on linearization

Let's write (1), equivalently, in matrix form as

$$\mathbf{y}_j = \boldsymbol{\mu}\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j\right) + \boldsymbol{\Lambda}_j^{\frac{1}{2}}\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j, \boldsymbol{\alpha}_w\right)\boldsymbol{e}_j$$

Here $\boldsymbol{\mu}\left(\mathbf{X}_j, \boldsymbol{\beta}, \mathbf{u}_j\right)$ depends on $\boldsymbol{\beta}$ and $\mathbf{u}_j$ through the function $\boldsymbol{d}(\cdot)$ in (2), and $\boldsymbol{e}_j$'s $\sim N\left(\mathbf{0}, \sigma^2 I_{n_j}\right)$, where I_{n_j} is the identity matrix of dimension n_j. In what follows, for brevity, we suppress the dependence of $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}_j$ on $\mathbf{X}_j$.

Following Lindstrom and Bates (1990), we will initially assume that $\boldsymbol{\Lambda}_j$ does not depend on $\mathbf{X}_j$, $\boldsymbol{\beta}$, and $\mathbf{u}_j$ or, equivalently, on $\boldsymbol{\phi}_j$ but rather on j only through its dimension; that is, $\boldsymbol{\Lambda}_j = \boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)$. Therefore, heteroskedastic structures that depend on the mean are not yet allowed in this context. Toward the end of this section, we will present a modified version of the algorithm that accounts for the dependence of $\boldsymbol{\Lambda}_j$ on $\boldsymbol{\phi}_j$.

Lindstrom and Bates discuss a natural extension of the methods for the LME models to NLME models. For a known $\boldsymbol{\alpha}$ (and thus known $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}_j$) and σ^2, the estimates of $\boldsymbol{\beta}$ and $\mathbf{u}_j$ jointly minimize

$$\sum_{j=1}^{M}\Big[\log|\boldsymbol{\Sigma}(\boldsymbol{\alpha}_u)| + \mathbf{u}_j'\{\boldsymbol{\Sigma}(\boldsymbol{\alpha}_u)\}^{-1}\mathbf{u}_j + \log\left|\sigma^2\boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)\right| + \sigma^{-2}\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}'\boldsymbol{\Lambda}_j^{-1}(\boldsymbol{\alpha}_w)\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}\Big]$$

which is twice the negative log likelihood for $\boldsymbol{\beta}$ when $\mathbf{u}_j$ is fixed or twice the negative log of the posterior density of $\mathbf{u}_j$ when $\boldsymbol{\beta}$ is fixed. Consequently, one strategy for estimating $\boldsymbol{\beta}$ and (predicting) $\mathbf{u}_j$ is to minimize the above objective function with respect to $\boldsymbol{\beta}$ and $\mathbf{u}_j$ given suitable estimates of $\boldsymbol{\alpha}$ and σ^2. Estimation of $\boldsymbol{\alpha}$ and σ^2 can be accomplished by using MLE with respect to the marginal density of $\mathbf{y}_j$, in which $\mathbf{u}_j$'s are integrated out. But because no closed-form expression for this density is available, we approximate the conditional distribution of $\mathbf{y}_j$ given $\mathbf{u}_j$ by a multivariate normal distribution with an expectation that is linear in $\mathbf{u}_j$ and $\boldsymbol{\beta}$. This is illustrated in step 2 of the algorithm below.

Lindstrom and Bates (1990) propose the following two-step estimation method or alternating algorithm.

Step 1 (PNLS step). Given current estimates $\widehat{\boldsymbol{\alpha}}$ (and thus $\widehat{\boldsymbol{\alpha}}_u$ and $\widehat{\boldsymbol{\alpha}}_w$) of $\boldsymbol{\alpha}$ and $\hat{\sigma}^2$ of σ^2, minimize with respect to $\boldsymbol{\beta}$ and $\mathbf{u}_j$

$$\sum_{j=1}^{M}\Big[\log|\boldsymbol{\Sigma}(\widehat{\boldsymbol{\alpha}}_u)| + \mathbf{u}_j'\{\boldsymbol{\Sigma}(\widehat{\boldsymbol{\alpha}}_u)\}^{-1}\mathbf{u}_j + \log\left|\hat{\sigma}^2\boldsymbol{\Lambda}_j(\widehat{\boldsymbol{\alpha}}_w)\right| + \hat{\sigma}^{-2}\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}'\boldsymbol{\Lambda}_j^{-1}(\widehat{\boldsymbol{\alpha}}_w)\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}\Big] \tag{26}$$

Define $\boldsymbol{\Delta}$ such that $\sigma^2\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Delta}'\boldsymbol{\Delta}$. Note that $\boldsymbol{\Delta} = \boldsymbol{\Delta}(\boldsymbol{\alpha}_u)$, but for notational convenience, this dependency is suppressed throughout the rest of this section. Equation (26) is equivalent to minimizing the penalized least-squares objective function

PNLS step:
$$\sum_{j=1}^{M}\left[\left\|\{\boldsymbol{\Lambda}_j'(\boldsymbol{\alpha}_w)\}^{-1/2}\{\mathbf{y}_j - \boldsymbol{\mu}(\boldsymbol{\beta},\mathbf{u}_j)\}\right\|^2 + ||\boldsymbol{\Delta}\mathbf{u}_j||^2\right]$$

with respect to $\boldsymbol{\beta}$ and $\mathbf{u}_j$ while holding the current estimates of $\boldsymbol{\alpha}$ (and, consequently, of $\boldsymbol{\Delta}$ and of $\boldsymbol{\Lambda}_j$) fixed. `pnlsopts(iterate(#))` iterations are performed at this step, unless the convergence criterion (CC) is met. The CC for PNLS optimization is controlled by `pnlsopts(nrtolerance(#))` and one of `pnlsopts(ltolerance(#))` or `pnlsopts(tolerance(#))`; see *menlmaxopts* for details.

Denote the resulting estimates as $\hat{\mathbf{u}}_j$ and $\widehat{\boldsymbol{\beta}}$.

In the absence of random effects in the model (see example 19), the previous formulas no longer include the random effects and related components. In particular, $\mathbf{u}_j$ and $\boldsymbol{\Delta}$ are set to $\mathbf{0}$, and $\boldsymbol{\alpha} = \boldsymbol{\alpha}_w$. In this case, the PNLS step reduces to what we call a GNLS estimation step. Furthermore, if no within-group error covariance structure is specified, that is, when all observations are assumed i.i.d., $\boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)$ is set to the identity matrix I, and the PNLS step reduces to the classical NLS estimation.

Step 2 (LME step). Perform a first-order Taylor-series expansion of the model mean function around the current estimates of $\boldsymbol{\beta}$ and of the conditional modes of the random effects $\mathbf{u}_j$, yielding

$$\mathbf{y}_j = \boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + \widehat{\mathbf{Z}}_j\left(\mathbf{u}_j - \hat{\mathbf{u}}_j\right) + \boldsymbol{\Lambda}_j^{\frac{1}{2}}\left(\boldsymbol{\alpha}_w\right)\boldsymbol{e}_j \tag{27}$$

where

$$\widehat{\mathbf{X}}_j = \left.\frac{\partial \boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)}{\partial \boldsymbol{\beta}'}\right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}, \mathbf{u}_j=\hat{\mathbf{u}}_j}$$

$$\widehat{\mathbf{Z}}_j = \left.\frac{\partial \boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)}{\partial \mathbf{u}_j'}\right|_{\boldsymbol{\beta}=\widehat{\boldsymbol{\beta}}, \mathbf{u}_j=\hat{\mathbf{u}}_j}$$

Model (27) is essentially an LME model, and we use notations $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Z}}_j$ for the derivatives to emphasize this. That is, $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Z}}_j$ represent the corresponding fixed-effects and random-effects design matrices of an LME model.

Thus the approximate conditional distribution of $\mathbf{y}_j$ is

$$\mathbf{y}_j|\mathbf{u}_j \sim N\left\{\boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + \widehat{\mathbf{Z}}_j(\mathbf{u}_j - \hat{\mathbf{u}}_j),\ \sigma^2\boldsymbol{\Lambda}_j\right\}$$

Because the expectation is now linear in random effects $\mathbf{u}_j$, the approximate conditional distribution of $\mathbf{y}_j$, along with distribution of $\mathbf{u}_j$, allows us to approximate the marginal distribution of $\mathbf{y}_j$ as

$$\mathbf{y}_j \sim N\left\{\boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) - \widehat{\mathbf{Z}}_j\hat{\mathbf{u}}_j,\ \sigma^2\mathbf{V}_j(\boldsymbol{\alpha})\right\} \tag{28}$$

where $\mathbf{V}_j(\boldsymbol{\alpha}) = \widehat{\mathbf{Z}}_j\boldsymbol{\Delta}^{-1}\left(\boldsymbol{\Delta}^{-1}\right)'\widehat{\mathbf{Z}}_j' + \boldsymbol{\Lambda}_j\left(\boldsymbol{\alpha}_w\right)$.

Let $\hat{\mathbf{w}}_j = \mathbf{y}_j - \boldsymbol{\mu}\left(\widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j\right) + \widehat{\mathbf{X}}_j\widehat{\boldsymbol{\beta}} + \widehat{\mathbf{Z}}_j\hat{\mathbf{u}}_j$. Estimation of $\boldsymbol{\alpha}$ and σ^2 can now be accomplished by maximizing the log likelihood corresponding to the approximate marginal distribution in (28),

LME step:

$$\begin{aligned} l_{\mathrm{LB}}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2}\sum_{j=1}^{M}\Big\{ &\log|\mathbf{V}_j(\boldsymbol{\alpha})| \\ &+ \sigma^{-2}\left(\hat{\mathbf{w}}_j - \widehat{\mathbf{X}}_j\boldsymbol{\beta}\right)'\mathbf{V}_j{}^{-1}(\boldsymbol{\alpha})\left(\hat{\mathbf{w}}_j - \widehat{\mathbf{X}}_j\boldsymbol{\beta}\right)\Big\} \end{aligned} \tag{29}$$

where $n = \sum_{j=1}^{M} n_j$.

Alternatively, when the `reml` option is specified, we take an REML approach and maximize

$$l_{\mathrm{LB},R}(\boldsymbol{\alpha}, \sigma^2) = l_{\mathrm{LB}}(\boldsymbol{\alpha}, \widehat{\boldsymbol{\beta}}(\boldsymbol{\alpha}), \sigma^2) - \frac{1}{2}\sum_{j=1}^{M}\log\left|\sigma^{-2}\widehat{\mathbf{X}}_j{}'\mathbf{V}_j{}^{-1}(\boldsymbol{\alpha})\widehat{\mathbf{X}}_j\right| \tag{30}$$

The LME step (step 2) of the alternating algorithm consists of optimizing an LME log likelihood, in which the response vector is given by $\hat{\mathbf{w}}_j$ and the fixed- and random-effects design matrices are given by $\widehat{\mathbf{X}}_j$ and $\widehat{\mathbf{Z}}_j$, respectively. `lmeopts(iterate(#))` iterations are performed at this step, unless the CC is met. The CC for LME optimization is controlled by `lmeopts(nrtolerance(#))` and one of `lmeopts(ltolerance(#))` or `lmeopts(tolerance(#))`; see *menlmaxopts* for details.

The LME step produces estimates $\widehat{\boldsymbol{\alpha}}$ and $\hat{\sigma}^2$. (The estimates $\widehat{\boldsymbol{\beta}}$ can also be obtained at this step, but it is generally more computationally efficient to compute them at the PNLS step.) These estimates will now be used in step 1, the PNLS step.

In the absence of random effects in the model (see example 19), $\mathbf{u}_j$, $\hat{\mathbf{u}}_j$, $\boldsymbol{\Delta}$, and $\widehat{\mathbf{Z}}_j$ are all set to $\mathbf{0}$, and $\boldsymbol{\alpha} = \boldsymbol{\alpha}_w$. In this case, the LME step is referred to as the ML step or, if the `reml` option is specified, the REML step in the `menl` output. Furthermore, if all observations are assumed i.i.d., then step 2 of the alternating algorithm is not needed, and only step 1 (NLS) is performed.

Stopping rules. One PNLS step and one LME step correspond to one iteration of the alternating algorithm. The log likelihood reported by `menl` at each iteration is the log likelihood (29) or, if the `reml` option is specified, (30) from the last iteration of the LME step. `menl` refers to this log likelihood as "linearization log likelihood" because it corresponds to the log likelihood of the LME model, which was the result of the linearization of the NLME model. The algorithm stops when the linearization likelihoods from successive iterations satisfy `ltolerance(#)`, when the parameter estimates from successive iterations satisfy `tolerance(#)`, or if the model does not converge, when the maximum number of iterations in `iterate()` is reached; see *menlmaxopts* for details about maximization options. Because the alternating algorithm does not provide a joint Hessian matrix for all parameters, there is no check for the tolerance of the scaled gradient; thus the convergence cannot be established in its strict sense. The convergence is declared based on the stopping rules described above.

When $\boldsymbol{\Lambda}_j = \boldsymbol{\Lambda}_j\left(\boldsymbol{\beta}, \mathbf{u}_j, \boldsymbol{\alpha}_w\right)$ depends on $\mathbf{u}_j$ and $\boldsymbol{\beta}$, which is the case, for example, with `resvariance(power _yhat)` and `resvariance(exponential _yhat)`), an intermediate step between the PNLS and the LME step is performed to replace the fixed effects and random effects in $\boldsymbol{\Lambda}_j$, or more precisely in the variance function $g(\cdot)$, by their current estimates from the PNLS step. After that, $\boldsymbol{\Lambda}_j\left(\boldsymbol{\alpha}_w; \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j\right) = \boldsymbol{\Lambda}_j(\boldsymbol{\alpha}_w)$ depends only on $\boldsymbol{\alpha}_w$ because both $\mathbf{u}_j$ and $\boldsymbol{\beta}$ are held fixed at their current estimates throughout the LME step.

Efficient methods for computing (29) or (30) are given in chapters 2 and 5 of Pinheiro and Bates (2000). Namely, to simplify the optimization problem, one can express the optimal values of $\boldsymbol{\beta}$ and σ^2 as functions of $\boldsymbol{\alpha}$ (and thus of $\boldsymbol{\Delta}$ and $\boldsymbol{\alpha}_w$) and work with the profiled log likelihood of $\boldsymbol{\alpha}$.

For the PNLS step, the objective function to be minimized is the penalized sum of squares

$$\sum_{j=1}^{M}\left[||(\boldsymbol{\Lambda}_j')^{-1/2}\left\{\mathbf{y}_j - \boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)\right\}||^2 + ||\boldsymbol{\Delta}\mathbf{u}_j||^2\right]$$

By adding "pseudo"-observations to the data, the PNLS problem can be reexpressed as a standard nonlinear least-squares problem. Thus step 1 of the alternating algorithm is sometimes called the "pseudodata step". Define pseudo-observations $\tilde{\mathbf{y}}_j$ as follows:

$$\tilde{\mathbf{y}}_j = \begin{bmatrix}(\boldsymbol{\Lambda}_j')^{-1/2}\mathbf{y}_j \\ \mathbf{0}\end{bmatrix} \qquad \widetilde{\boldsymbol{\mu}}\left(\boldsymbol{\beta}, \mathbf{u}_j\right) = \begin{bmatrix}(\boldsymbol{\Lambda}_j')^{-1/2}\boldsymbol{\mu}\left(\boldsymbol{\beta}, \mathbf{u}_j\right) \\ \boldsymbol{\Delta}\mathbf{u}_j\end{bmatrix}$$

Then, the PNLS step can be rewritten as

$$\sum_{j=1}^{M}\left|\left|\tilde{\mathbf{y}}_j - \widetilde{\boldsymbol{\mu}}\left(\boldsymbol{\beta}, \mathbf{u}_j\right)\right|\right|^2$$

Hence, for values of $\boldsymbol{\alpha}$ and σ^2 fixed at the current estimates, the estimation of $\boldsymbol{\beta}$ and $\mathbf{u}_j$ in the PNLS step can be regarded as a standard nonlinear least-squares problem. A popular iterative estimation technique for standard nonlinear least-squares is the Gauss–Newton method (see Pinheiro and Bates [2000, chap. 7] for more details).

After the completion of the alternating algorithm, an extra LME iteration is performed, with fixed effects profiled-out of the likelihood, to reparameterize $[\boldsymbol{\alpha},\ \log(\sigma)]$ to their natural metric and to compute their standard errors with the delta method. This step is labeled `Computing standard errors:` in the output of `menl`. If you are interested only in standard errors for fixed effects, you can skip this step by specifying the `nostderr` option, in which case standard errors for the random-effects and within-group error covariance parameters will not be computed and will be shown as missing in the output table. The standard errors for the fixed effects are obtained from the PNLS step, and the standard errors for random-effects parameters are obtained from the LME step.

Inference on the parameters of the NLME model is based on the approximating LME model with log likelihood and restricted log likelihood functions defined in (29) and (30). Therefore, all the inferential machinery available within the context of LME models can be used. For example, under the LME approximation, the distribution of the (restricted) MLE $\widehat{\boldsymbol{\beta}}$ of the fixed effects is

$$\widehat{\boldsymbol{\beta}} \sim N\left\{\boldsymbol{\beta},\ \sigma^2\left(\sum_{j=1}^{M}\widehat{\mathbf{X}}_j{}'\mathbf{V}_j{}^{-1}(\boldsymbol{\alpha})\widehat{\mathbf{X}}_j\right)^{-1}\right\}$$

and for random-effects and within-group error parameters is

$$\begin{bmatrix}\widehat{\boldsymbol{\alpha}}\\ \log\widehat{\sigma}\end{bmatrix} \sim N\left\{\begin{bmatrix}\boldsymbol{\alpha}\\ \log\sigma\end{bmatrix}, I^{-1}(\boldsymbol{\alpha},\sigma)\right\}$$

where

$$I(\boldsymbol{\alpha},\sigma) = -\begin{bmatrix}\partial^2 l_{\mathrm{LB}_p}/\partial\boldsymbol{\alpha}\partial\boldsymbol{\alpha}' & \partial^2 l_{\mathrm{LB}_p}/\partial\log\sigma\partial\boldsymbol{\alpha}'\\ \partial^2 l_{\mathrm{LB}_p}/\partial\boldsymbol{\alpha}\partial\log\sigma & \partial^2 l_{\mathrm{LB}_p}/\partial^2\log\sigma\end{bmatrix}$$

and $l_{\mathrm{LB}_p} = l_{\mathrm{LB}_p}(\boldsymbol{\alpha},\sigma)$ is the approximated log likelihood from the LME step with fixed effects profiled out. Because inference is based on the LME approximation of the original NLME model, asymptotic results are technically "approximately asymptotic" and are thus less accurate than the asymptotic inferential results for LME models as described in [ME] **mixed**.

Initial values

The PNLS step requires starting values for $\boldsymbol{\beta}$ and $\mathbf{u}_j$. These are obtained from the EM algorithm; see, for example, Bates and Pinheiro (1998) for details. You can control optimization within the EM algorithm by specifying the `emtolerance()` and `emiterate()` options. You can also supply your own initial values; see *Examples of specifying initial values*. NLME models are often sensitive to initial values, so it is good practice to try different sets of initial values to verify that your results are robust to them.

References

Assaad, H. 2017. Nonlinear multilevel mixed-effects models. *The Stata Blog: Not Elsewhere Classified*. https://blog.stata.com/2017/11/03/nonlinear-multilevel-mixed-effects-models/.

Bates, D. M., and J. C. Pinheiro. 1998. "Computational methods for multilevel modelling". In *Technical Memorandum BL0112140-980226-01TM*. Murray Hill, NJ: Bell Labs, Lucent Technologies.

Boeckmann, A. J., L. B. Sheiner, and S. L. Beal. 2011. *NONMEM Users Guide, Part V: Introductory Guide*. San Francisco: Regents of the University of California. https://nonmem.iconplc.com/nonmem720/guides/v.pdf.

Carroll, R. J., and D. Ruppert. 1988. *Transformation and Weighting in Regression*. New York: Chapman and Hall. https://doi.org/10.1201/9780203735268.

Davidian, M., and D. M. Giltinan. 1995. *Nonlinear Models for Repeated Measurement Data*. Boca Raton, FL: Chapman and Hall/CRC. https://doi.org/10.1201/9780203745502.

———. 2003. Nonlinear models for repeated measurement data: An overview and update. *Journal of Agricultural, Biological, and Environmental Statistics* 8: 387–419. https://doi.org/10.1198/1085711032697.

Demidenko, E. 2013. *Mixed Models: Theory and Applications with R*. 2nd ed. Hoboken, NJ: Wiley.

Draper, N., and H. Smith. 1998. *Applied Regression Analysis*. 3rd ed. New York: Wiley. https://doi.org/10.1002/9781118625590.

Fitzmaurice, G. M., M. Davidian, G. Verbeke, and G. Molenberghs, eds. 2009. *Longitudinal Data Analysis*. Boca Raton, FL: Chapman and Hall/CRC. https://doi.org/10.1201/9781420011579.

Gibaldi, M., and D. Perrier. 1982. *Pharmacokinetics*. 2nd ed, revised and expanded. New York: Dekker. https://doi.org/10.1201/b14095.

Grasela, T. H., Jr., and S. M. Donn. 1985. Neonatal population pharmacokinetics of phénobarbital derived from routine clinical data. *Developmental Pharmacology and Therapeutics* 8: 374–383. https://doi.org/10.1159/000457062.

Hand, D. J., and M. J. Crowder. 1996. *Practical Longitudinal Data Analysis*. Boca Raton, FL: Chapman and Hall. https://doi.org/10.1201/9780203742372.

Harring, J. R., and J. Liu. 2016. A comparison of estimation methods for nonlinear mixed-effects models under model misspecification and data sparseness: A simulation study. *Journal of Modern Applied Statistical Methods* 15(1): Article 27. https://doi.org/10.22237/jmasm/1462076760.

Joyner, W. B., and D. M. Boore. 1981. Peak horizontal acceleration and velocity from strong-motion records including records from the 1979 imperial valley, California, earthquake. *Bulletin of the Seismological Society of America* 71: 2011–2038.

Lindstrom, M. J., and D. M. Bates. 1990. Nonlinear mixed effects models for repeated measures data. *Biometrics* 46: 673–687. https://doi.org/10.2307/2532087.

Pierson, R. A., and O. J. Ginther. 1987. Follicular population dynamics during the estrous cycle of the mare. *Animal Reproduction Science* 14: 219–231. https://doi.org/10.1016/0378-4320(87)90085-6.

Pinheiro, J. C., and D. M. Bates. 1995. Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics* 4: 12–35. https://doi.org/10.2307/1390625.

———. 2000. *Mixed-Effects Models in S and S-PLUS*. New York: Springer. https://doi.org/10.1007/b98882.

Plan, E. L., A. Maloney, F. Mentré, M. O. Karlsson, and J. Bertrand. 2012. Performance comparison of various maximum likelihood nonlinear mixed-effects estimation methods for dose–response models. *AAPS Journal* 14: 420–432. https://doi.org/10.1208/s12248-012-9349-2.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Thompson, W. A., Jr. 1962. The problem of negative estimates of variance components. *Annals of Mathematical Statistics* 33: 273–289. https://doi.org/10.1214/aoms/1177704731.

Verme, C. N., T. M. Ludden, W. A. Clementi, and S. C. Harris. 1992. Pharmacokinetics of quinidine in male patients: A population analysis. *Clinical Pharmacokinetics* 22: 468–480. https://doi.org/10.2165/00003088-199222060-00005.

Vonesh, E. F., and R. L. Carter. 1992. Mixed-effects nonlinear regression for unbalanced repeated measures. *Biometrics* 48: 1–17. https://doi.org/10.2307/2532734.

Vonesh, E. F., and V. M. Chinchilli. 1997. *Linear and Nonlinear Models for the Analysis of Repeated Measurements*. New York: Dekker. https://doi.org/10.1201/9781482293272.

Wolfinger, R. D., and X. Lin. 1997. Two Taylor-series approximation methods for nonlinear mixed models. *Computational Statistics and Data Analysis* 25: 465–490. https://doi.org/10.1016/S0167-9473(97)00012-1.

Also see

[ME] **menl postestimation** — Postestimation tools for menl

[ME] **meglm** — Multilevel mixed-effects generalized linear models

[ME] **mixed** — Multilevel mixed-effects linear regression+

[ME] **me** — Introduction to multilevel mixed-effects models

[R] **nl** — Nonlinear least-squares estimation

[U] 20 Estimation and postestimation commands

menl postestimation — Postestimation tools for menl

Postestimation commands predict margins Remarks and examples
Methods and formulas Reference Also see

Postestimation commands

The following postestimation commands are of special interest after `menl`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat recovariance`	display the estimated random-effects covariance matrices
`estat sd`	display variance components as standard deviations and correlations
`estat wcorrelation`	display within-cluster correlations and standard deviations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`etable`	table of estimation results
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
`lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	predictions and their SEs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

predict

Description for predict

`predict` creates a new variable containing predictions of mean values, residuals, or standardized residuals. It can also create multiple new variables containing estimates of random effects and their standard errors or containing predicted named substitutable expressions.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] *newvar* [*if*] [*in*] [, *statistic* `fixedonly` `relevel(`*levelvar*`)` *options*]

Syntax for predicting named substitutable expressions (parameters)

Predict all parameters

`predict` [*type*] {*stub*`*`|*newvars*} [*if*] [*in*], `parameters`
[`fixedonly` `relevel(`*levelvar*`)` *options*]

Predict specific parameters

`predict` [*type*] (*newvar* = {*param*`:`}) [(*newvar* = {*param*`:`})] [...] [*if*] [*in*]
[, `fixedonly` `relevel(`*levelvar*`)` *options*]

`predict` [*type*] {*stub*`*`|*newvars*} [*if*] [*in*], `parameters(`*paramnames*`)`
[`fixedonly` `relevel(`*levelvar*`)` *options*]

Syntax for obtaining predictions of random effects and their standard errors

`predict` [*type*] {*stub*`*`|*newvars*} [*if*] [*in*], `reffects` [`relevel(`*levelvar*`)`
`reses(`*stub*`*`|*newvars*`)` *options*]

paramnames is *param* [*param* [...]] and *param* is a name of a substitutable expression as specified in one of `menl`'s `define()` options.

statistic	Description
Main	
`yhat`	prediction for the expected response conditional on the random effects
`mu`	synonym for `yhat`
`residuals`	residuals, response minus predicted values
* `rstandard`	standardized residuals

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

options	Description
Options	
`iterate(#)`	maximum number of iterations when computing random effects; default is `iterate(10)`
`tolerance(#)`	convergence tolerance when computing random effects; default is `tolerance(1e-6)`
`nrtolerance(#)`	scaled gradient tolerance when computing random effects; default is `nrtolerance(1e-5)`
`nonrtolerance`	ignore the `nrtolerance()` option

Options for predict

Main

`yhat` calculates the predicted values, which are the mean-response values conditional on the random effects, $\mu(\mathbf{x}'_{ij}, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j)$. By default, the predicted values account for random effects from all levels in the model; however, if the `relevel(`*levelvar*`)` option is specified, then the predicted values are fit beginning with the topmost level down to and including level *levelvar*. For example, if `classes` are nested within `schools`, then typing

```
. predict yhat_school, yhat relevel(school)
```

would produce school-level predictions. That is, the predictions would incorporate school-specific random effects but not those for each class nested within each school. If the `fixedonly` option is specified, predicted values conditional on zero random effects, $\mu(\mathbf{x}'_{ij}, \widehat{\boldsymbol{\beta}}, \mathbf{0})$, are calculated based on the estimated fixed effects (coefficients) in the model when the random effects are fixed at their theoretical mean value of $\mathbf{0}$.

`mu` is a synonym for `yhat`.

`residuals` calculates residuals, equal to the responses minus the predicted values `yhat`. By default, the predicted values account for random effects from all levels in the model; however, if the `relevel(`*levelvar*`)` option is specified, then the predicted values are fit beginning at the topmost level down to and including level *levelvar*.

`rstandard` calculates standardized residuals, equal to the residuals multiplied by the inverse square root of the estimated error covariance matrix.

parameters and parameters(*paramnames*) calculate predictions for all or a subset of the named substitutable expressions in the model. By default, the predictions account for random effects from all levels in the model; however, if the relevel(*levelvar*) option is specified, then the predictions would incorporate random effects from the topmost level down to and including level *levelvar*. Option parameters(*param*) is useful with margins. parameters() does not appear in the dialog box.

reffects calculates predictions of the random effects. For the Lindstrom–Bates estimation method of menl, these are essentially the best linear unbiased predictions (BLUPs) of the random effects in the LME approximated log likelihood; see *Inference based on linearization* in [ME] **menl**. By default, estimates of all random effects in the model are calculated. However, if the relevel(*levelvar*) option is specified, then estimates of random effects for only level *levelvar* in the model are calculated. For example, if classes are nested within schools, then typing

```
. predict b*, reffects relevel(school)
```

would produce estimates at the school level. You must specify q new variables, where q is the number of random-effects terms in the model (or level). However, it is much easier to just specify *stub*∗ and let Stata name the variables *stub*1, *stub*2, …, *stubq* for you.

fixedonly specifies that all random effects be set to zero, equivalent to using only the fixed portion of the model.

relevel(*levelvar*) specifies the level in the model at which predictions involving random effects are to be obtained; see the options above for the specifics. *levelvar* is the name of the model level; it is the name of the variable describing the grouping at that level.

reses(*stub*∗ | *newvars*) calculates the standard errors of the estimates of the random effects. By default, standard errors for all random effects in the model are calculated. However, if the relevel(*levelvar*) option is specified, then standard errors of the estimates of the random effects for only level *levelvar* in the model are calculated; see the reffects option.

You must specify q new variables, where q is the number of random-effects terms in the model (or level). However, it is much easier to just specify *stub*∗ and let Stata name the variables *stub*1, *stub*2, …, *stubq* for you. The new variables will have the same storage type as the corresponding random-effects variables.

The reffects and reses() options often generate multiple new variables at once. When this occurs, the random effects (or standard errors) contained in the generated variables correspond to the order in which the variance components are listed in the output of menl. Still, examining the variable labels of the generated variables (with the describe command, for instance) can be useful in deciphering which variables correspond to which terms in the model.

Options

iterate(#) specifies the maximum number of iterations when computing estimates of the random effects. The default is iterate(10). This option is relevant only to predictions that depend on random effects. This option is not allowed if the fixedonly option is specified.

tolerance(#) specifies a convergence tolerance when computing estimates of the random effects. The default is tolerance(1e-6). This option is relevant only to predictions that depend on random effects. This option is not allowed if the fixedonly option is specified.

nrtolerance(#) and nonrtolerance control the tolerance for the scaled gradient when computing estimates of the random effects.

`nrtolerance(#)` specifies the tolerance for the scaled gradient. Convergence is declared when $g(-H^{-1})g'$ is less than `nrtolerance(#)`, where g is the gradient row vector and H is the approximated Hessian matrix from the current iteration. The default is `nrtolerance(1e-5)`.

`nonrtolerance` specifies that the default `nrtolerance()` criterion be turned off.

margins

Description for margins

`margins` estimates margins of response for predicted mean values or named substitutable expressions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* `...)` [*options*]

statistic	Description
`yhat`	predicted values conditional on zero random effects; the default
`mu`	synonym for `yhat`
`residuals`	not allowed with `margins`
`rstandard`	not allowed with `margins`
`parameters`	predicted parameters
`parameters(`*param*`)`	predicted named substitutable expression *param* conditional on zero random effects
`reffects`	not allowed with `margins`

The `fixedonly` option is assumed for the predictions used with `margins`.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting an NLME model using `menl`. For the most part, calculation centers on obtaining estimates of the random effects. Random effects are not estimated when the model is fit but instead need to be predicted after estimation. The estimates of the random effects are in turn used to obtain predicted values and residuals at different nesting levels. These are useful for checking model assumptions and may be used in general as model-building tools.

▷ Example 1: Testing variance components

In example 9 and example 12 of [ME] **menl**, we modeled the average leaf weight of two genotypes of soybean plants over three growing seasons as

$$\texttt{weight}_{ij} = \frac{\phi_{1j}}{1 + \exp\left\{-\left(\texttt{time}_{ij} - \phi_{2j}\right)/\phi_{3j}\right\}} + \epsilon_{ij}$$

for $j = 1, \ldots, 48$ and $i = 1, \ldots, n_j$, with $8 \leq n_j \leq 10$. Here we consider a simplified version of the stage 2 model specification from example 12,

$$\boldsymbol{\phi}_j = \begin{bmatrix}\phi_{1j}\\ \phi_{2j}\\ \phi_{3j}\end{bmatrix} = \begin{bmatrix}\beta_{11} + \beta_{12}S_{89,j} + \beta_{13}S_{90,j} + u_{1j}\\ \beta_{21} + \beta_{22}S_{89,j} + \beta_{23}S_{90,j} + \beta_{24}P_j\\ \beta_{31} + \beta_{32}S_{89,j} + \beta_{33}S_{90,j}\end{bmatrix}$$

where $P_j = I(\texttt{variety}_j = \texttt{P})$, $S_{89,j} = I(\texttt{year}_j = 1989)$, and $S_{90,j} = I(\texttt{year}_j = 1990)$. The random effects u_{1j}'s are normally distributed with mean 0 and variance σ^2_{u1} and errors ϵ_{ij}'s are normally distributed with mean 0 and error variance

$$\text{Var}\left(\epsilon_{ij}\right) = \sigma^2(\widehat{\texttt{weight}}_{ij})^{2\delta}$$

Let's fit this model using `menl`.

```
. use https://www.stata-press.com/data/r19/soybean
(Growth of soybean plants (Davidian and Giltinan, 1995))
. menl weight = {phi1:}/(1+exp(-(time-{phi2:})/{phi3:})),
> define(phi1: i.year U1[plot])
> define(phi2: i.year i.variety)
> define(phi3: i.year, xb) resvariance(power _yhat, noconstant)
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood = -324.21579
Iteration 2:  Linearization log likelihood = -313.89733
Iteration 3:  Linearization log likelihood = -314.76287
Iteration 4:  Linearization log likelihood =  -314.4317
Iteration 5:  Linearization log likelihood =  -314.5131
Iteration 6:  Linearization log likelihood = -314.49399
Iteration 7:  Linearization log likelihood = -314.49922
Iteration 8:  Linearization log likelihood = -314.49838
Iteration 9:  Linearization log likelihood = -314.49853
Iteration 10: Linearization log likelihood = -314.49851
```

```
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        412
Group variable: plot                            Number of groups  =         48
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.6
                                                              max =         10
                                                Wald chi2(7)      =     193.98
Linearization log likelihood = -314.49851       Prob > chi2       =     0.0000
       phi1: i.year U1[plot]
       phi2: i.year i.variety
       phi3: i.year
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
year						
1989	-6.614797	1.195633	-5.53	0.000	-8.958195	-4.271399
1990	-3.749016	1.264716	-2.96	0.003	-6.227814	-1.270218
_cons	20.28009	.953959	21.26	0.000	18.41036	22.14981
phi2						
year						
1989	-2.623514	.9752055	-2.69	0.007	-4.534882	-.7121468
1990	-5.142726	.9879783	-5.21	0.000	-7.079128	-3.206324
variety						
P	-2.265482	.3274228	-6.92	0.000	-2.907219	-1.623745
_cons	55.25935	.7554917	73.14	0.000	53.77861	56.74008
phi3						
year						
1989	-.9538782	.1963606	-4.86	0.000	-1.338738	-.5690186
1990	-.7220007	.2081227	-3.47	0.001	-1.129914	-.3140877
_cons	8.042677	.1452638	55.37	0.000	7.757965	8.327389

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
plot: Identity				
var(U1)	4.260962	1.114483	2.551939	7.11451
Residual variance:				
Power _yhat				
sigma2	.046573	.0038271	.0396449	.0547118
delta	.9667451	.0229472	.9217694	1.011721

`menl` does not report tests against zeros for parameters in the random-effects table because they are not appropriate for all types of parameters such as variances. For some parameters such as power parameter δ in our example, labeled as `delta` in the output, the test of H_0: $\delta = 0$ is sensible. In fact, it corresponds to the test of homoskedastic within-plot errors because under the null hypothesis the error variance $\text{Var}(\epsilon_{ij}) = \sigma^2(\widehat{\texttt{weight}}_{ij})^{2\delta}$ reduces to σ^2.

We can use the `test` command to perform this test.

```
. test _b[/Residual:delta] = 0
 ( 1)  [/Residual]delta = 0
           chi2(  1) = 1774.87
         Prob > chi2 =    0.0000
```

The Wald test strongly rejects the null hypothesis of homoskedastic errors.

◁

▷ Example 2: Obtaining predictions

Continuing with example 1, we can also obtain the estimates of the plot-level random effects u_{1j}'s. Because `menl` used the Lindstrom–Bates linearization method, the estimated random effects are essentially BLUPs; see *Inference based on linearization* in [ME] **menl**.

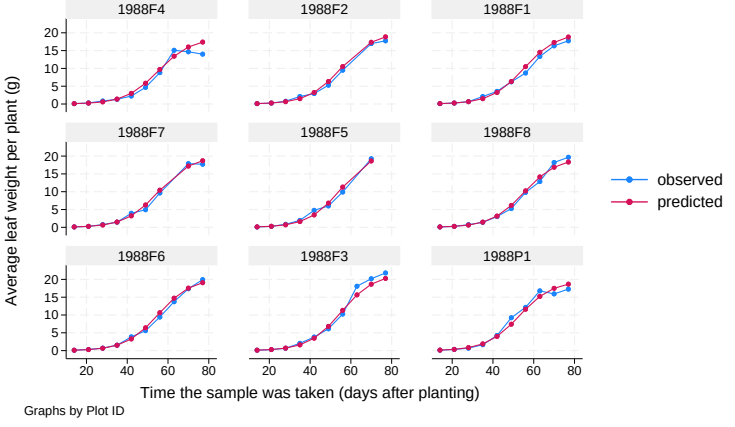
We need to specify the name of the variable to be created and then use `predict, reffects`. For example, below we obtain the predictions of random effects for the first 10 plots.

```
. predict u1, reffects
. by plot, sort: generate tolist = (_n==1)
. list plot u1 if plot <=10 & tolist
```

	plot	u1
1.	1988F4	-1.716238
11.	1988F2	-.1668753
20.	1988F1	-.2153712
30.	1988F7	-.3337681
39.	1988F5	1.331566
47.	1988F8	-.7153563
57.	1988F6	.0699629
67.	1988F3	1.353845
77.	1988P1	-.6681811
87.	1988P5	-.9152615

Next, we obtain the predicted mean values and plot them. By default, the mean response conditional on the estimated random effects is computed. Predicted values based on the fixed-effects estimates alone, that is, conditional on zero random effects, may be obtained by specifying the `fixedonly` option.

```
. predict fitweight, yhat
. twoway connected weight fitweight time if plot<=9, sort by(plot)
> ytitle("Average leaf weight per plant (g)")
> legend(order(1 "observed" 2 "predicted"))
```



The predicted values closely match the observed average leaf weights, confirming the adequacy of the model.

Also see example 13 in [ME] **menl** for how to predict parameters defined as functions of other parameters with substitutable expressions.

◁

▷ Example 3: Checking model assumptions based on residuals

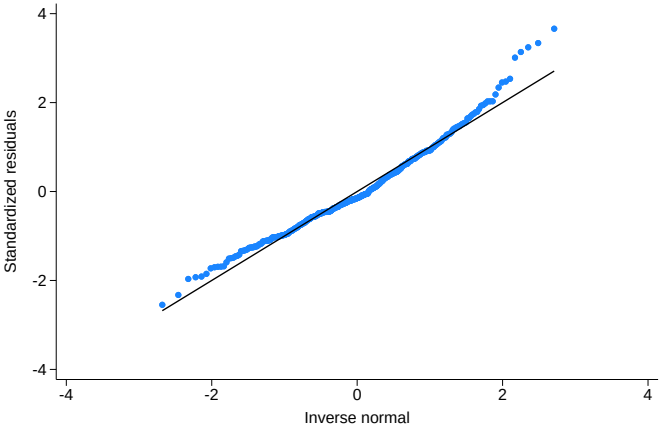
The raw residuals are useful to check for heterogeneity of the within-group error variance; see example 10 in [ME] **menl**. They are less recommended, however, for checking normality assumptions and for detecting outlying observations. This is because raw residuals are usually correlated and have different variances. Instead, we can use standardized residuals to check for normality and outlying observations. If the normality assumption is reasonable and the model fits data well, standardized residuals should follow a standard normal distribution.

Let's check whether the standardized residuals from our model are approximately normally distributed with mean zero and variance one.

```
. predict rs, rstandard
. summarize rs
```

Variable	Obs	Mean	Std. dev.	Min	Max
rs	412	.0150192	.9564895	-2.547287	3.661603

```
. qnorm rs
```



The plot does not indicate serious departures from normality, and the estimated mean and standard deviation are close to zero and one, respectively. It appears that the power of the mean function is a reasonable choice for modeling heteroskedasticity of the within-group errors in this example.

Example 4: estat group and level-specific predictions

In example 23 of [ME] **menl**, we modeled the intensity of current at the ith level of `voltage` in the jth site within the kth wafer as

$$\texttt{current}_{ijk} = \phi_{1jk} + \phi_{2jk}\cos\left(\phi_{3jk}\texttt{voltage}_i + \pi/4\right) + \epsilon_{ijk}$$

for $k = 1,\ldots,10$, $j = 1,\ldots,8$, and $i = 1,\ldots,5$. In that example, we considered fairly complicated specifications for ϕ_j's in stage 2 with many random effects at different levels, which lead to slow execution of the command. To illustrate some of the commands available after `menl`, we will substantially simplify the stage 2 specification to speed up the estimation of the model.

$$\begin{aligned}
\phi_{1jk} &= \beta_0 + u^{(3)}_{0k} + u^{(2)}_{0j,k} + \left(\beta_1 + u^{(3)}_{1k} + u^{(2)}_{1j,k}\right)\texttt{voltage}_i \\
\phi_{2jk} &= \beta_3 \\
\phi_{3jk} &= \beta_4
\end{aligned}$$

$$\mathbf{u}^{(3)}_k = \begin{bmatrix} u^{(3)}_{0k} \\ u^{(3)}_{1k} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}_3) \qquad \mathbf{u}^{(2)}_{j,k} = \begin{bmatrix} u^{(2)}_{0j,k} \\ u^{(2)}_{1j,k} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{\Sigma}_2) \qquad \epsilon_{ijk} \sim N(0, \sigma^2_\epsilon)$$

where

$$\mathbf{\Sigma}_3 = \begin{bmatrix} \sigma^{(3)}_{11} & 0 \\ 0 & \sigma^{(3)}_{22} \end{bmatrix} \qquad \mathbf{\Sigma}_2 = \begin{bmatrix} \sigma^{(2)}_{11} & 0 \\ 0 & \sigma^{(2)}_{22} \end{bmatrix}$$

We fit this model by using menl.

```
. use https://www.stata-press.com/data/r19/wafer, clear
(Modeling of analog MOS circuits)
. menl current = {phi1:}+{phi2}*cos({phi3}*voltage + _pi/4),
> define(phi1: voltage W0[wafer] S0[wafer>site]
>              c.voltage#(W1[wafer] S1[wafer>site]))
Obtaining starting values by EM:
Alternating PNLS/LME algorithm:
Iteration 1:  Linearization log likelihood =  503.60719
Iteration 2:  Linearization log likelihood =  503.60719
Computing standard errors:
Mixed-effects ML nonlinear regression           Number of obs     =        400
      Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
wafer	10	40	40.0	40
wafer>site	80	5	5.0	5

```
                                                Wald chi2(1)      =    4860.30
Linearization log likelihood =  503.60719       Prob > chi2       =     0.0000
        phi1: voltage W0[wafer] S0[wafer>site] c.voltage#W1[wafer]
              c.voltage#S1[wafer>site]
```

current	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
phi1						
voltage	-25.20026	.3614709	-69.72	0.000	-25.90873	-24.49179
_cons	64.41187	.4984303	129.23	0.000	63.43497	65.38878
/phi2	-93.6509	.6367582	-147.07	0.000	-94.89892	-92.40287
/phi3	.3828109	.001525	251.02	0.000	.379822	.3857999

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
wafer: Independent				
var(W0)	.0048116	.0027516	.0015686	.014759
var(W1)	.0396212	.0184213	.0159284	.0985558
wafer>site: Independent				
var(S0)	.0086449	.0017812	.0057726	.0129463
var(S1)	.0118703	.0021115	.0083763	.0168217
var(Residual)	.001069	.0000952	.0008978	.0012729

We can use `estat group` to see how the data are broken down by wafer and site:

```
. estat group
Grouping information
```

Path	No. of groups	Observations per group Minimum	Average	Maximum
wafer	10	40	40.0	40
wafer>site	80	5	5.0	5

We are reminded that we have balanced data for each site (all sites were measured at 5 ascending voltages).

Suppose that we want to predict random effects at the wafer level only; that is, we want to compute $\hat{\mathbf{u}}_k^{(3)}$. This can be done by specifying the `relevel(wafer)` option:

```
. predict u_wafer*, reffects relevel(wafer)
```

Notice how `predict` labels the generated variables for you to avoid confusion.

```
. describe u_wafer*
```

Variable name	Storage type	Display format	Value label	Variable label
u_wafer1	float	%9.0g		BLUP r.e. for W0[wafer]
u_wafer2	float	%9.0g		BLUP r.e. for W1[wafer]

We can use `predict, yhat` to get the predicted values $\mu(\texttt{voltage}_i, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_k^{(3)}, \hat{\mathbf{u}}_{j,k}^{(2)})$. If instead we want to predict values at the wafer level, $\mu(\texttt{voltage}_i, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_k^{(3)}, \mathbf{0})$, we again need to specify the `relevel()` option:

```
. predict curr_wafer, yhat relevel(wafer)
. list wafer site current curr_wafer in 1/10
```

	wafer	site	current	curr_w~r
1.	1	1	.90088	.8898317
2.	1	1	3.8682	3.920231
3.	1	1	7.6406	7.65254
4.	1	1	11.736	11.76189
5.	1	1	15.934	15.91457
6.	1	2	1.032	.8898317
7.	1	2	4.1022	3.920231
8.	1	2	7.9316	7.65254
9.	1	2	12.064	11.76189
10.	1	2	16.294	15.91457

The predicted values `curr_wafer` do not vary across sites, because $\mu(\texttt{voltage}_i, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_k^{(3)}, \mathbf{0})$ does not depend on j.

◁

Methods and formulas

Following the notation defined throughout [ME] **menl**, estimates of random effects $\mathbf{u}_j$ are obtained by using PNLS iterations with parameters $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$, and σ^2 held fixed at their values obtained at convergence. Starting with $\hat{\mathbf{u}}_j^{(0)} = \mathbf{0}$, at the kth iteration, we have

$$\hat{\mathbf{u}}_j^{(k)} = \widehat{\boldsymbol{\Sigma}}\widehat{\mathbf{Z}}_j'^{(k-1)} \left(\widehat{\mathbf{Z}}_j^{(k-1)}\widehat{\boldsymbol{\Sigma}}\widehat{\mathbf{Z}}_j'^{(k-1)} + \hat{\sigma}^2\widehat{\boldsymbol{\Lambda}}_j\right)^{-1} \left(\widehat{\mathbf{w}}_j^{(k-1)} - \widehat{\mathbf{X}}_j^{(k-1)}\widehat{\boldsymbol{\beta}}\right)$$

where $\widehat{\boldsymbol{\Sigma}}$ and $\widehat{\boldsymbol{\Lambda}}$ are $\boldsymbol{\Sigma}$ and $\boldsymbol{\Lambda}$ with maximum likelihood (ML) or restricted maximum likelihood (REML) estimates of the variance components plugged in and $\widehat{\mathbf{X}}_j^{(k-1)} = \widehat{\mathbf{X}}_j(\hat{\mathbf{u}}_j^{(k-1)})$, $\widehat{\mathbf{Z}}_j^{(k-1)} = \widehat{\mathbf{Z}}_j(\hat{\mathbf{u}}_j^{(k-1)})$, and $\widehat{\mathbf{w}}_j^{(k-1)} = \widehat{\mathbf{w}}_j(\hat{\mathbf{u}}_j^{(k-1)})$ are defined in the LME step of *Inference based on linearization* in *Methods and formulas* of [ME] **menl**. When the variance structure depends on $\mathbf{u}_j$, such as when the `resvariance(power _yhat)` option is specified during estimation, $\widehat{\boldsymbol{\Lambda}}_j$ will also be updated at each iteration; that is, $\widehat{\boldsymbol{\Lambda}}_j = \widehat{\boldsymbol{\Lambda}}_j(\hat{\mathbf{u}}_j^{(k-1)})$. The iterative process stops when the relative difference between $\hat{\mathbf{u}}_j^{(k-1)}$ and $\hat{\mathbf{u}}_j^{(k)}$ is less than `tolerance(#)` or, if the stopping rule is not met, when the maximum number of iterations in `iterate(#)` is reached.

Standard errors for the estimates of the random effects are calculated based on Bates and Pinheiro (1998, sec. 3.3). If estimation is done by REML, these standard errors account for uncertainty in the estimate of $\boldsymbol{\beta}$, whereas for ML, the standard errors treat $\boldsymbol{\beta}$ as known. As such, standard errors of REML-based estimates will usually be larger.

Predicted mean values are computed as $\boldsymbol{\mu}_j(\mathbf{X}_j, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j)$, predicted parameters as $\widehat{\boldsymbol{\phi}}_j = [\mathbf{d}](\mathbf{x}_j^b, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j)$, residuals as $\hat{\boldsymbol{\epsilon}}_j = \mathbf{y}_j - \boldsymbol{\mu}_j(\mathbf{X}_j, \widehat{\boldsymbol{\beta}}, \hat{\mathbf{u}}_j)$, and standardized residuals as

$$\hat{\boldsymbol{\epsilon}}_j^* = \hat{\sigma}^{-1}\widehat{\boldsymbol{\Lambda}}_j^{-1/2}\hat{\boldsymbol{\epsilon}}_j$$

If the `relevel(`*levelvar*`)` option is specified, predicted values, residuals, and standardized residuals consider only those random-effects terms up to and including level *levelvar* in the model. If the `fixedonly` option is specified, all statistics and named substitutable expressions are computed conditional on zero random effects; that is, their computation is based on the estimated fixed effects only.

Reference

Bates, D. M., and J. C. Pinheiro. 1998. "Computational methods for multilevel modelling". In *Technical Memorandum BL0112140-980226-01TM*. Murray Hill, NJ: Bell Labs, Lucent Technologies.

Also see

[ME] **menl** — Nonlinear mixed-effects regression

[U] 20 Estimation and postestimation commands

meologit — Multilevel mixed-effects ordered logistic regression

Description Quick start Menu Syntax Options
Remarks and examples Stored results Methods and formulas References Also see

Description

`meologit` fits mixed-effects logistic models for ordered responses. The actual values taken on by the response are irrelevant except that larger values are assumed to correspond to "higher" outcomes. The conditional distribution of the response given the random effects is assumed to be multinomial, with success probability determined by the logistic cumulative distribution function.

Quick start

Two-level ordered logit regression of `y` on indicators for levels of `a` and random intercepts by `lev2`

```
meologit y i.a || lev2:
```

Two-level model including fixed and random coefficients for `x`

```
meologit y i.a x || lev2: x
```

Same as above, but report odds ratios instead of coefficients

```
meologit y i.a x || lev2: x, or
```

Three-level model of `y` on `a`, `x`, and their interaction using factor variable notation and random intercepts by `lev2` and `lev3` with `lev2` nested within `lev3`

```
meologit y a##c.x || lev3: || lev2:
```

Menu

Statistics > Multilevel mixed-effects models > Ordered logistic regression

Syntax

```
meologit depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`or`	report fixed-effects coefficients as odds ratios
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: meologit**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`offset(`*varname*`)` specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

noconstant suppresses the constant (intercept) term; may be specified for any of or all the random-effects equations.

fweight(*varname*) specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, [fw=*fwtvar1*]. *varname* can be any valid Stata variable name, and you can specify fweight() at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable wt1 would hold the first-level (the observation-level) frequency weights, and wt2 would hold the second-level (the school-level) frequency weights.

iweight(*varname*) specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, [iw=*iwtvar1*]. *varname* can be any valid Stata variable name, and you can specify iweight() at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable wt1 would hold the first-level (the observation-level) importance weights, and wt2 would hold the second-level (the school-level) importance weights.

pweight(*varname*) specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, [pw=*pwtvar1*]. *varname* can be any valid Stata variable name, and you can specify pweight() at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable wt1 would hold the first-level (the observation-level) sampling weights, and wt2 would hold the second-level (the school-level) sampling weights.

constraints(*constraints*); see [R] **Estimation options**.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim, opg), that are robust to some kinds of misspecification (robust), and that allow for intragroup correlation (cluster *clustvar*); see [R] ***vce_option***. If vce(robust) is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

level(#); see [R] **Estimation options**.

or reports estimated fixed-effects coefficients transformed to odds ratios, that is, $\exp(\beta)$ rather than β. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated. or may be specified either at estimation or upon replay.

nocnsreport; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite performs mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for meologit are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with meologit but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

Mixed-effects ordered logistic regression is ordered logistic regression containing both fixed effects and random effects. An ordered response is a variable that is categorical and ordered, for instance, “poor”, “good”, and “excellent”, which might indicate a person’s current health status or the repair record of a car.

meologit allows for many levels of random effects. However, for simplicity, for now we consider the two-level model, where for a series of M independent clusters, and conditional on a set of fixed effects $\mathbf{x}_{ij}$, a set of cutpoints $\boldsymbol{\kappa}$, and a set of random effects $\mathbf{u}_j$, the cumulative probability of the response being in a category higher than k is

$$\Pr(y_{ij} > k|\mathbf{x}_{ij}, \boldsymbol{\kappa}, \mathbf{u}_j) = H(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j - \kappa_k) \tag{1}$$

for $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The cutpoints $\boldsymbol{\kappa}$ are labeled $\kappa_1, \kappa_2, \ldots, \kappa_{K-1}$, where K is the number of possible outcomes. $H(\cdot)$ is the logistic cumulative distribution function that represents cumulative probability.

The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard logistic regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. In our parameterization, $\mathbf{x}_{ij}$ does not contain a constant term because its effect is absorbed into the cutpoints. For notational convenience here and throughout this manual entry, we suppress the dependence of y_{ij} on $\mathbf{x}_{ij}$.

The $1 \times q$ vector $\mathbf{z}_{ij}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$, so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

From (1), we can derive the probability of observing outcome k as

$$\begin{aligned}
\Pr(y_{ij} = k|\boldsymbol{\kappa}, \mathbf{u}_j) &= \Pr(\kappa_{k-1} < \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij} \leq \kappa_k) \\
&= \Pr(\kappa_{k-1} - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j < \epsilon_{ij} \leq \kappa_k - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j) \\
&= H(\kappa_k - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j) - H(\kappa_{k-1} - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j)
\end{aligned}$$

where κ_0 is taken as $-\infty$ and κ_K is taken as $+\infty$.

From the above, we may also write the model in terms of a latent linear response, where observed ordinal responses y_{ij} are generated from the latent continuous responses, such that

$$y^*_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij}$$

and

$$y_{ij} = \begin{cases} 1 & \text{if} \quad y^*_{ij} \leq \kappa_1 \\ 2 & \text{if} \quad \kappa_1 < y^*_{ij} \leq \kappa_2 \\ \vdots & \\ K & \text{if} \quad \kappa_{K-1} < y^*_{ij} \end{cases}$$

The errors ϵ_{ij} are distributed as logistic with mean 0 and variance $\pi^2/3$ and are independent of $\mathbf{u}_j$.

Below we present two short examples of mixed-effects ordered logistic regression; refer to [ME] **me** and [ME] **meglm** for examples of other random-effects models. A two-level ordered logistic model can also be fit using xtologit with the re option; see [XT] **xtologit**. In the absence of random effects, mixed-effects ordered logistic regression reduces to standard ordered logistic regression; see [R] **ologit**.

Example 1: Two-level random-intercept model

We use the data from the Television, School, and Family Smoking Prevention and Cessation Project (Flay et al. 1988; Rabe-Hesketh and Skrondal 2022, chap. 11), where schools were randomly assigned into one of four groups defined by two treatment variables. Students within each school are nested in classes, and classes are nested in schools. In this example, we ignore the variability of classes within schools and fit a two-level model; we incorporate classes in a three-level model in example 2. The dependent variable is the tobacco and health knowledge (THK) scale score collapsed into four ordered categories. We regress the outcome on the treatment variables and their interaction and control for the pretreatment score.

```
. use https://www.stata-press.com/data/r19/tvsfpors
(Television, School, and Family Project)
. meologit thk prethk cc##tv || school:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood =  -2125.509
Iteration 2:  Log likelihood = -2125.1034
Iteration 3:  Log likelihood = -2125.1032
Refining starting values:
Grid node 0:  Log likelihood = -2136.2426
Fitting full model:
Iteration 0:  Log likelihood = -2136.2426  (not concave)
Iteration 1:  Log likelihood = -2120.2577
Iteration 2:  Log likelihood = -2119.7574
Iteration 3:  Log likelihood = -2119.7428
Iteration 4:  Log likelihood = -2119.7428
Mixed-effects ologit regression               Number of obs     =      1,600
Group variable: school                        Number of groups  =         28
                                              Obs per group:
                                                            min =         18
                                                            avg =       57.1
                                                            max =        137
Integration method: mvaghermite               Integration pts.  =          7
                                              Wald chi2(4)      =     128.06
Log likelihood = -2119.7428                   Prob > chi2       =     0.0000
```

thk	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
prethk	.4032892	.03886	10.38	0.000	.327125	.4794534
1.cc	.9237904	.204074	4.53	0.000	.5238127	1.323768
1.tv	.2749937	.1977424	1.39	0.164	-.1125744	.6625618
cc#tv						
1 1	-.4659256	.2845963	-1.64	0.102	-1.023724	.0918728
/cut1	-.0884493	.1641062			-.4100916	.233193
/cut2	1.153364	.165616			.8287625	1.477965
/cut3	2.33195	.1734199			1.992053	2.671846
school						
var(_cons)	.0735112	.0383106			.0264695	.2041551

```
LR test vs. ologit model: chibar2(01) = 10.72         Prob >= chibar2 = 0.0005
```

The estimation table reports the fixed effects, the estimated cutpoints $(\kappa_1, \kappa_2, \kappa_3)$, and the estimated variance components. The fixed effects can be interpreted just as you would the output from `ologit`. We find that students with higher preintervention scores tend to have higher postintervention scores. Because of their interaction, the impact of the treatment variables on the knowledge score is not straightforward; we defer this discussion to example 1 of [ME] **meologit postestimation**.

Underneath the fixed effects and the cutpoints, the table shows the estimated variance components. The random-effects equation is labeled `school`, meaning that these are random effects at the `school` level. Because we have only one random effect at this level, the table shows only one variance component. The estimate of σ_u^2 is 0.07 with standard error 0.04. The reported likelihood-ratio test shows that there is enough variability between schools to favor a mixed-effects ordered logistic regression over a standard ordered logistic regression; see *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

We now store our estimates for later use.

```
. estimates store r_2
```

◁

Two-level models extend naturally to models with three or more levels with nested random effects. Below we continue with example 1.

▷ Example 2: Three-level random-intercept model

In this example, we fit a three-level model incorporating classes nested within schools. The fixed-effects part remains the same. Our model now has two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `school` level (level three), and the second is a random intercept at the `class` level (level two). The order in which these are specified (from left to right) is significant—`meologit` assumes that `class` is nested within `school`.

```
. meologit thk prethk cc##tv || school: || class:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood =  -2125.509
Iteration 2:  Log likelihood = -2125.1034
Iteration 3:  Log likelihood = -2125.1032
Refining starting values:
Grid node 0:  Log likelihood = -2152.1514
Fitting full model:
Iteration 0:  Log likelihood = -2152.1514  (not concave)
Iteration 1:  Log likelihood = -2125.9213  (not concave)
Iteration 2:  Log likelihood = -2120.1861
Iteration 3:  Log likelihood = -2115.6177
Iteration 4:  Log likelihood = -2114.5896
Iteration 5:  Log likelihood = -2114.5881
Iteration 6:  Log likelihood = -2114.5881
Mixed-effects ologit regression                 Number of obs     =      1,600
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
school	28	18	57.1	137
class	135	1	11.9	28

```
Integration method: mvaghermite                 Integration pts.  =         7
                                                Wald chi2(4)      =    124.39
Log likelihood = -2114.5881                     Prob > chi2       =    0.0000
```

thk	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
prethk	.4085273	.039616	10.31	0.000	.3308814	.4861731
1.cc	.8844369	.2099124	4.21	0.000	.4730161	1.295858
1.tv	.236448	.2049065	1.15	0.249	-.1651614	.6380575
cc#tv						
1 1	-.3717699	.2958887	-1.26	0.209	-.951701	.2081612
/cut1	-.0959459	.1688988			-.4269815	.2350896
/cut2	1.177478	.1704946			.8433151	1.511642
/cut3	2.383672	.1786736			2.033478	2.733865
school						
var(_cons)	.0448735	.0425387			.0069997	.2876749
school>class						
var(_cons)	.1482157	.0637521			.063792	.3443674

```
LR test vs. ologit model: chi2(2) = 21.03                 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

We see that we have 135 classes from 28 schools. The variance-component estimates are now organized and labeled according to level. The variance component for `class` is labeled `school>class` to emphasize that classes are nested within schools.

Compared with the two-level model from example 1, the estimate of the variance of the random intercept at the school level dropped from 0.07 to 0.04. This is not surprising because we now use two random components versus one random component to account for unobserved heterogeneity among students. We can use `lrtest` and our stored estimation result from example 1 to see which model provides a better fit:

```
. lrtest r_2 .
Likelihood-ratio test
Assumption: r_2 nested within .
 LR chi2(1) =  10.31
Prob > chi2 = 0.0013
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

The likelihood-ratio test favors the three-level model. For more information about the likelihood-ratio test in the context of mixed-effects models, see *Distribution theory for likelihood-ratio test* in [ME] **me**. ◁

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by `||`.

Stored results

meologit stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_dv)	number of dependent variables
e(k_cat)	number of categories
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(N_clust)	number of clusters
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	meologit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	ologit
e(title)	title in estimation output
e(link)	logit
e(family)	ordinal
e(clustvar)	name of cluster variable
e(offset)	offset
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V

e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(marginsdefault)	default predict() specification for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved
Matrices	
e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(cat)	category values
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
Functions	
e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices	
r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

meologit is a convenience command for meglm with a logit link and an ordinal family; see [ME] **meglm**.

Without a loss of generality, consider a two-level ordered logistic model. The probability of observing outcome k for response y_{ij} is then

$$\begin{aligned} p_{ij} = \Pr(y_{ij}=k|\boldsymbol{\kappa},\mathbf{u}_j) &= \Pr(\kappa_{k-1} < \boldsymbol{\eta}_{ij} + \epsilon_{it} \le \kappa_k) \\ &= \frac{1}{1+\exp(-\kappa_k+\boldsymbol{\eta}_{ij})} - \frac{1}{1+\exp(-\kappa_{k-1}+\boldsymbol{\eta}_{ij})} \end{aligned}$$

where $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \text{offset}_{ij}$, κ_0 is taken as $-\infty$, and κ_K is taken as $+\infty$. Here $\mathbf{x}_{ij}$ does not contain a constant term because its effect is absorbed into the cutpoints.

For cluster j, $j = 1,\ldots,M$, the conditional distribution of $\mathbf{y}_j = (y_{j1},\ldots,y_{jn_j})'$ given a set of cluster-level random effects $\mathbf{u}_j$ is

$$\begin{aligned} f(\mathbf{y}_j|\boldsymbol{\kappa},\mathbf{u}_j) &= \prod_{i=1}^{n_j} p_{ij}^{I_k(y_{ij})} \\ &= \exp\sum_{i=1}^{n_j}\left\{I_k(y_{ij})\,\log(p_{ij})\right\} \end{aligned}$$

where

$$I_k(y_{ij}) = \begin{cases} 1 & \text{if } y_{ij} = k \\ 0 & \text{otherwise} \end{cases}$$

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j, \mathbf{u}_j)$,

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\kappa}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2} \, |\boldsymbol{\Sigma}|^{-1/2} \int f(\mathbf{y}_j | \boldsymbol{\kappa}, \mathbf{u}_j) \exp\left(-\mathbf{u}_j' \boldsymbol{\Sigma}^{-1} \mathbf{u}_j / 2\right) d\mathbf{u}_j \\ &= (2\pi)^{-q/2} \, |\boldsymbol{\Sigma}|^{-1/2} \int \exp\left\{h\left(\boldsymbol{\beta}, \boldsymbol{\kappa}, \boldsymbol{\Sigma}, \mathbf{u}_j\right)\right\} d\mathbf{u}_j \end{aligned} \tag{2}$$

where

$$h\left(\boldsymbol{\beta}, \boldsymbol{\kappa}, \boldsymbol{\Sigma}, \mathbf{u}_j\right) = \sum_{i=1}^{n_j} \left\{ I_k(y_{ij}) \, \log(p_{ij}) \right\} - \mathbf{u}_j' \boldsymbol{\Sigma}^{-1} \mathbf{u}_j / 2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j, \mathbf{r}_j, \mathbf{X}_j, \mathbf{Z}_j)$.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`meologit` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Flay, B. R., B. R. Brannon, C. A. Johnson, W. B. Hansen, A. L. Ulene, D. A. Whitney-Saltiel, L. R. Gleason, S. Sussman, M. D. Gavin, K. M. Glowacz, D. F. Sobol, and D. C. Spiegel. 1988. The television, school, and family smoking cessation and prevention project: I. Theoretical basis and program development. *Preventive Medicine* 17: 585–607. https://doi.org/10.1016/0091-7435(88)90053-9.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Also see

[ME] **meologit postestimation** — Postestimation tools for meologit

[ME] **meoprobit** — Multilevel mixed-effects ordered probit regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: meologit** — Bayesian multilevel ordered logistic regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtologit** — Random-effects ordered logistic model

[U] 20 Estimation and postestimation commands

meologit postestimation — Postestimation tools for meologit

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation command is of special interest after `meologit`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as probabilities, linear predictions, density and distribution functions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`pr`	predicted probabilities; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
`outcome(`*outcome*`)`	outcome category for predicted probabilities
Integration	
int_options	integration options

You specify one or k new variables in *newvars* with `pr`, where k is the number of outcomes. If you do not specify `outcome()`, these options assume `outcome(#1)`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`pr`, the default, calculates the predicted probabilities.

You specify one or k new variables, where k is the number of categories of the dependent variable. If you specify the `outcome()` option, the probabilities will be predicted for the requested outcome only, in which case you specify only one new variable. If you specify one new variable and do not specify `outcome()`, `outcome(#1)` is assumed.

`eta`, `xb`, `stdp`, `density`, `distribution`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`outcome(`*outcome*`)` specifies the outcome for which the predicted probabilities are to be calculated. `outcome()` should contain either one value of the dependent variable or one of `#1`, `#2`, ..., with `#1` meaning the first category of the dependent variable, `#2` meaning the second category, etc.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`, see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
default	probabilities for each outcome
`pr`	predicted probabilities for a specified outcome
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

`pr` defaults to the first outcome.

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.

Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting an ordered logistic mixed-effects model with `meologit`. Here we show a short example of predicted probabilities and predicted random effects; refer to [ME] **meglm postestimation** for additional examples applicable to mixed-effects generalized linear models.

▷ Example 1: Obtaining predicted probabilities and random effects

In example 2 of [ME] **meologit**, we modeled the tobacco and health knowledge (`thk`) score—coded 1, 2, 3, 4—among students as a function of two treatments (`cc` and `tv`) by using a three-level ordered logistic model with random effects at the school and class levels.

```
. use https://www.stata-press.com/data/r19/tvsfpors
(Television, School, and Family Project)
. meologit thk prethk cc##tv || school: || class:
```

(*output omitted*)

We obtain predicted probabilities for all four outcomes based on the contribution of both fixed effects and random effects by typing

```
. predict pr*
(option pr assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

As the note says, the predicted values are based on the posterior means of random effects. You can use the `modes` option to obtain predictions based on the posterior modes of random effects.

Because we specified a stub name, Stata saved the predicted random effects in variables `pr1` through `pr4`. Here we list the predicted probabilities for the first two classes for school 515:

```
. list class thk pr? if school==515 & (class==515101 | class==515102),
> sepby(class)
```

	class	thk	pr1	pr2	pr3	pr4
1464.	515101	2	.1485538	.2354556	.2915916	.3243991
1465.	515101	2	.372757	.3070787	.1966117	.1235526
1466.	515101	1	.372757	.3070787	.1966117	.1235526
1467.	515101	4	.2831409	.3021398	.2397316	.1749877
1468.	515101	3	.2079277	.2760683	.2740791	.2419248
1469.	515101	3	.2831409	.3021398	.2397316	.1749877
1470.	515102	1	.3251654	.3074122	.2193101	.1481123
1471.	515102	2	.4202843	.3011963	.1749344	.103585
1472.	515102	2	.4202843	.3011963	.1749344	.103585
1473.	515102	2	.4202843	.3011963	.1749344	.103585
1474.	515102	2	.3251654	.3074122	.2193101	.1481123
1475.	515102	1	.4202843	.3011963	.1749344	.103585
1476.	515102	2	.3251654	.3074122	.2193101	.1481123

For each observation, our best guess for the predicted outcome is the one with the highest predicted probability. For example, for the very first observation in the table above, we would choose outcome 4 as the most likely to occur.

We obtain predictions of the posterior means themselves at the school and class levels by typing

```
. predict re_s re_c, reffects
(calculating posterior means of random effects)
(using 7 quadrature points)
```

Here we list the predicted random effects for the first two classes for school 515:

```
. list class re_s re_c if school==515 & (class==515101 | class==515102),
> sepby(class)
```

	class	re_s	re_c
1464.	515101	-.0473739	.0633081
1465.	515101	-.0473739	.0633081
1466.	515101	-.0473739	.0633081
1467.	515101	-.0473739	.0633081
1468.	515101	-.0473739	.0633081
1469.	515101	-.0473739	.0633081
1470.	515102	-.0473739	-.1354929
1471.	515102	-.0473739	-.1354929
1472.	515102	-.0473739	-.1354929
1473.	515102	-.0473739	-.1354929
1474.	515102	-.0473739	-.1354929
1475.	515102	-.0473739	-.1354929
1476.	515102	-.0473739	-.1354929

We can see that the predicted random effects at the school level (`re_s`) are the same for all classes and that the predicted random effects at the class level (`re_c`) are constant within each class.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

Also see

[ME] **meologit** — Multilevel mixed-effects ordered logistic regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

meoprobit — Multilevel mixed-effects ordered probit regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`meoprobit` fits mixed-effects probit models for ordered responses. The actual values taken on by the response are irrelevant except that larger values are assumed to correspond to "higher" outcomes. The conditional distribution of the response given the random effects is assumed to be multinomial, with success probability determined by the standard normal cumulative distribution function.

Quick start

Two-level ordered probit regression of `y` on `x` and random intercepts by `lev2`

```
meoprobit y x || lev2:
```

Add random coefficients for `x`

```
meoprobit y x || lev2: x
```

Nested three-level ordered probit model with random intercepts by `lev2` and `lev3` for `lev2` nested within `lev3`

```
meoprobit y x || lev3: || lev2:
```

Menu

Statistics > Multilevel mixed-effects models > Ordered probit regression

Syntax

```
meoprobit depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: meoprobit**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`offset(`*varname*`)` specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

noconstant suppresses the constant (intercept) term; may be specified for any of or all the random-effects equations.

fweight(*varname*) specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, [fw=*fwtvar1*]. *varname* can be any valid Stata variable name, and you can specify fweight() at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable wt1 would hold the first-level (the observation-level) frequency weights, and wt2 would hold the second-level (the school-level) frequency weights.

iweight(*varname*) specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, [iw=*iwtvar1*]. *varname* can be any valid Stata variable name, and you can specify iweight() at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable wt1 would hold the first-level (the observation-level) importance weights, and wt2 would hold the second-level (the school-level) importance weights.

pweight(*varname*) specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, [pw=*pwtvar1*]. *varname* can be any valid Stata variable name, and you can specify pweight() at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable wt1 would hold the first-level (the observation-level) sampling weights, and wt2 would hold the second-level (the school-level) sampling weights.

constraints(*constraints*); see [R] **Estimation options**.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim, opg), that are robust to some kinds of misspecification (robust), and that allow for intragroup correlation (cluster *clustvar*); see [R] ***vce_option***. If vce(robust) is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

level(#), nocnsreport; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

`intmethod(`*intmethod*`)` specifies the integration method to be used for the random-effects model. `mvaghermite` performs mean–variance adaptive Gauss–Hermite quadrature; `mcaghermite` performs mode-curvature adaptive Gauss–Hermite quadrature; `ghermite` performs nonadaptive Gauss–Hermite quadrature; and `laplace` performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is `mvaghermite` unless a crossed random-effects model is fit, in which case the default integration method is `laplace`. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

`intpoints(`#`)` sets the number of integration points for quadrature. The default is `intpoints(7)`, which means that seven quadrature points are used for each level of random effects. This option is not allowed with `intmethod(laplace)`.

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(`#`)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(`#`)`, `ltolerance(`#`)`, `nrtolerance(`#`)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize**. Those that require special mention for `meoprobit` are listed below.

`from()` accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with `meoprobit` but are not shown in the dialog box:

`startvalues(`*svmethod*`)`, `startgrid[(`*gridspec*`)]`, `noestimate`, and `dnumerical`; see [ME] **meglm**.

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

Mixed-effects ordered probit regression is ordered probit regression containing both fixed effects and random effects. An ordered response is a variable that is categorical and ordered, for instance, "poor", "good", and "excellent", which might indicate a person's current health status or the repair record of a car.

`meoprobit` allows for many levels of random effects. However, for simplicity, for now we consider the two-level model, where for a series of M independent clusters, and conditional on a set of fixed effects $\mathbf{x}_{ij}$, a set of cutpoints $\boldsymbol{\kappa}$, and a set of random effects $\mathbf{u}_j$, the cumulative probability of the response being in a category higher than k is

$$\Pr(y_{ij} > k|\mathbf{x}_{ij}, \boldsymbol{\kappa}, \mathbf{u}_j) = \Phi(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j - \kappa_k) \tag{1}$$

for $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The cutpoints are labeled κ_1, κ_2, …, κ_{K-1}, where K is the number of possible outcomes. $\Phi(\cdot)$ is the standard normal cumulative distribution function that represents cumulative probability.

The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard probit regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. In our parameterization, $\mathbf{x}_{ij}$ does not contain a constant term because its effect is absorbed into the cutpoints. For notational convenience here and throughout this manual entry, we suppress the dependence of y_{ij} on $\mathbf{x}_{ij}$.

The $1 \times q$ vector $\mathbf{z}_{ij}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$ so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

From (1), we can derive the probability of observing outcome k as

$$\begin{aligned}
\Pr(y_{ij} = k|\boldsymbol{\kappa}, \mathbf{u}_j) &= \Pr(\kappa_{k-1} < \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij} \le \kappa_k) \\
&= \Pr(\kappa_{k-1} - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j < \epsilon_{ij} \le \kappa_k - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j) \\
&= \Phi(\kappa_k - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j) - \Phi(\kappa_{k-1} - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j)
\end{aligned}$$

where κ_0 is taken as $-\infty$ and κ_K is taken as $+\infty$.

From the above, we may also write the model in terms of a latent linear response, where observed ordinal responses y_{ij} are generated from the latent continuous responses, such that

$$y^*_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij}$$

and

$$y_{ij} = \begin{cases} 1 & \text{if} \quad y^*_{ij} \le \kappa_1 \\ 2 & \text{if} \quad \kappa_1 < y^*_{ij} \le \kappa_2 \\ \vdots & \\ K & \text{if} \quad \kappa_{K-1} < y^*_{ij} \end{cases}$$

The errors ϵ_{ij} are distributed as standard normal with mean 0 and variance 1 and are independent of $\mathbf{u}_j$.

Below we present two short examples of mixed-effects ordered probit regression; refer to [ME] **me** and [ME] **meglm** for examples of other random-effects models. A two-level ordered probit model can also be fit using `xtoprobit` with the `re` option; see [XT] **xtoprobit**. In the absence of random effects, mixed-effects ordered probit regression reduces to standard ordered probit regression; see [R] **oprobit**.

Example 1: Two-level random-intercept model

We use the data from the Television, School, and Family Smoking Prevention and Cessation Project (Flay et al. 1988; Rabe-Hesketh and Skrondal 2022, chap. 11), where schools were randomly assigned into one of four groups defined by two treatment variables. Students within each school are nested in classes, and classes are nested in schools. In this example, we ignore the variability of classes within schools and fit a two-level model; we incorporate classes in a three-level model in example 2. The dependent variable is the tobacco and health knowledge (THK) scale score collapsed into four ordered categories. We regress the outcome on the treatment variables and their interaction and control for the pretreatment score.

```
. use https://www.stata-press.com/data/r19/tvsfpors
(Television, School, and Family Project)
. meoprobit thk prethk cc##tv || school:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood = -2127.8111
Iteration 2:  Log likelihood = -2127.7612
Iteration 3:  Log likelihood = -2127.7612
Refining starting values:
Grid node 0:  Log likelihood = -2149.7302
Fitting full model:
Iteration 0:  Log likelihood = -2149.7302  (not concave)
Iteration 1:  Log likelihood = -2129.6838  (not concave)
Iteration 2:  Log likelihood = -2123.5143
Iteration 3:  Log likelihood = -2122.2896
Iteration 4:  Log likelihood = -2121.7949
Iteration 5:  Log likelihood = -2121.7716
Iteration 6:  Log likelihood = -2121.7715
Mixed-effects oprobit regression               Number of obs     =      1,600
Group variable: school                         Number of groups  =         28
                                               Obs per group:
                                                             min =         18
                                                             avg =       57.1
                                                             max =        137
Integration method: mvaghermite                Integration pts.  =          7
                                               Wald chi2(4)      =     128.05
Log likelihood = -2121.7715                    Prob > chi2       =     0.0000
```

thk	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
prethk	.2369804	.0227739	10.41	0.000	.1923444	.2816164
1.cc	.5490957	.1255108	4.37	0.000	.303099	.7950923
1.tv	.1695405	.1215889	1.39	0.163	-.0687693	.4078504
cc#tv						
1 1	-.2951837	.1751969	-1.68	0.092	-.6385634	.0481959
/cut1	-.0682011	.1003374			-.2648587	.1284565
/cut2	.67681	.1008836			.4790817	.8745382
/cut3	1.390649	.1037494			1.187304	1.593995
school						
var(_cons)	.0288527	.0146201			.0106874	.0778937

```
LR test vs. oprobit model: chibar2(01) = 11.98        Prob >= chibar2 = 0.0003
```

The estimation table reports the fixed effects, the estimated cutpoints $(\kappa_1, \kappa_2, \kappa_3)$, and the estimated variance components. The fixed effects can be interpreted just as you would the output from `oprobit`. We find that students with higher preintervention scores tend to have higher postintervention scores. Because of their interaction, the impact of the treatment variables on the knowledge score is not straightforward; we defer this discussion to example 1 of [ME] **meoprobit postestimation**.

Underneath the fixed effects and the cutpoints, the table shows the estimated variance components. The random-effects equation is labeled `school`, meaning that these are random effects at the `school` level. Because we have only one random effect at this level, the table shows only one variance component. The estimate of σ_u^2 is 0.03 with standard error 0.01. The reported likelihood-ratio test shows that there is enough variability between schools to favor a mixed-effects ordered probit regression over a standard ordered probit regression; see *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

We now store our estimates for later use.

```
. estimates store r_2
```

◁

Two-level models extend naturally to models with three or more levels with nested random effects. Below we continue with example 1.

▷ Example 2: Three-level random-intercept model

In this example, we fit a three-level model incorporating classes nested within schools. The fixed-effects part remains the same. Our model now has two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `school` level (level three), and the second is a random intercept at the `class` level (level two). The order in which these are specified (from left to right) is significant—`meoprobit` assumes that `class` is nested within `school`.

```
. meoprobit thk prethk cc##tv || school: || class:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood = -2127.8111
Iteration 2:  Log likelihood = -2127.7612
Iteration 3:  Log likelihood = -2127.7612
Refining starting values:
Grid node 0:  Log likelihood = -2195.6424
Fitting full model:
Iteration 0:  Log likelihood = -2195.6424  (not concave)
Iteration 1:  Log likelihood = -2167.9576  (not concave)
Iteration 2:  Log likelihood = -2140.2644  (not concave)
Iteration 3:  Log likelihood = -2128.6948  (not concave)
Iteration 4:  Log likelihood = -2119.9225
Iteration 5:  Log likelihood = -2117.0947
Iteration 6:  Log likelihood = -2116.7004
Iteration 7:  Log likelihood = -2116.6981
Iteration 8:  Log likelihood = -2116.6981
Mixed-effects oprobit regression               Number of obs     =      1,600
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
school	28	18	57.1	137
class	135	1	11.9	28

```
Integration method: mvaghermite                Integration pts.  =          7
                                               Wald chi2(4)      =     124.20
Log likelihood = -2116.6981                    Prob > chi2       =     0.0000
```

thk	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
prethk	.238841	.0231446	10.32	0.000	.1934784	.2842036
1.cc	.5254813	.1285816	4.09	0.000	.2734659	.7774967
1.tv	.1455573	.1255827	1.16	0.246	-.1005803	.3916949
cc#tv						
1 1	-.2426203	.1811999	-1.34	0.181	-.5977656	.1125251
/cut1	-.074617	.1029791			-.2764523	.1272184
/cut2	.6863046	.1034813			.4834849	.8891242
/cut3	1.413686	.1064889			1.204972	1.622401
school						
var(_cons)	.0186456	.0160226			.0034604	.1004695
school>class						
var(_cons)	.0519974	.0224014			.0223496	.1209745

```
LR test vs. oprobit model: chi2(2) = 22.13                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We see that we have 135 classes from 28 schools. The variance-component estimates are now organized and labeled according to level. The variance component for `class` is labeled `school>class` to emphasize that classes are nested within schools.

Compared with the two-level model from example 1, the estimate of the random intercept at the school level dropped from 0.03 to 0.02. This is not surprising because we now use two random components versus one random component to account for unobserved heterogeneity among students. We can use `lrtest` and our stored estimation result from example 1 to see which model provides a better fit:

```
. lrtest r_2 .
Likelihood-ratio test
Assumption: r_2 nested within .
 LR chi2(1) =  10.15
Prob > chi2 = 0.0014
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

The likelihood-ratio test favors the three-level model. For more information about the likelihood-ratio test in the context of mixed-effects models, see *Distribution theory for likelihood-ratio test* in [ME] **me**.

◁

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by `||`.

Stored results

`meoprobit` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_dv)`	number of dependent variables
`e(k_cat)`	number of categories
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_f)`	number of fixed-effects parameters
`e(k_r)`	number of random-effects parameters
`e(k_rs)`	number of variances
`e(k_rc)`	number of covariances
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(N_clust)`	number of clusters
`e(chi2)`	χ^2
`e(p)`	p-value for model test
`e(ll_c)`	log likelihood, comparison model
`e(chi2_c)`	χ^2, comparison test
`e(df_c)`	degrees of freedom, comparison test
`e(p_c)`	p-value for comparison test
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	`1` if converged, `0` otherwise

Macros

`e(cmd)`	`meglm`
`e(cmd2)`	`meoprobit`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(wtype)`	weight type
`e(wexp)`	weight expression (first-level weights)
`e(fweight`k`)`	`fweight` variable for kth highest level, if specified

e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	oprobit
e(title)	title in estimation output
e(link)	probit
e(family)	ordinal
e(clustvar)	name of cluster variable
e(offset)	offset
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(marginsdefault)	default predict() specification for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(cat)	category values
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

`meoprobit` is a convenience command for `meglm` with a `probit` link and an `ordinal` family; see [ME] **meglm**.

Without a loss of generality, consider a two-level ordered probit model. The probability of observing outcome k for response y_{ij} is then

$$\begin{aligned} p_{ij} = \Pr(y_{ij}=k|\boldsymbol{\kappa},\mathbf{u}_j) &= \Pr(\kappa_{k-1} < \boldsymbol{\eta}_{ij} + \epsilon_{it} \le \kappa_k) \\ &= \Phi(\kappa_k - \boldsymbol{\eta}_{ij}) - \Phi(\kappa_{k-1} - \boldsymbol{\eta}_{ij}) \end{aligned}$$

where $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \text{offset}_{ij}$, κ_0 is taken as $-\infty$, and κ_K is taken as $+\infty$. Here $\mathbf{x}_{ij}$ does not contain a constant term because its effect is absorbed into the cutpoints.

For cluster j, $j = 1,\ldots,M$, the conditional distribution of $\mathbf{y}_j = (y_{j1},\ldots,y_{jn_j})'$ given a set of cluster-level random effects $\mathbf{u}_j$ is

$$\begin{aligned} f(\mathbf{y}_j|\mathbf{u}_j) &= \prod_{i=1}^{n_j} p_{ij}^{I_k(y_{ij})} \\ &= \exp\sum_{i=1}^{n_j}\left\{ I_k(y_{ij})\,\log(p_{ij})\right\} \end{aligned}$$

where

$$I_k(y_{ij}) = \begin{cases} 1 & \text{if } y_{ij}=k \\ 0 & \text{otherwise} \end{cases}$$

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j,\mathbf{u}_j)$,

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta},\boldsymbol{\kappa},\boldsymbol{\Sigma}) &= (2\pi)^{-q/2}\,|\boldsymbol{\Sigma}|^{-1/2}\int f(\mathbf{y}_j|\boldsymbol{\kappa},\mathbf{u}_j)\exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right)d\mathbf{u}_j \\ &= (2\pi)^{-q/2}\,|\boldsymbol{\Sigma}|^{-1/2}\int \exp\left\{h\left(\boldsymbol{\beta},\boldsymbol{\kappa},\boldsymbol{\Sigma},\mathbf{u}_j\right)\right\}d\mathbf{u}_j \end{aligned} \tag{2}$$

where

$$h\left(\boldsymbol{\beta},\boldsymbol{\kappa},\boldsymbol{\Sigma},\mathbf{u}_j\right) = \sum_{i=1}^{n_j}\left\{ I_k(y_{ij})\,\log(p_{ij})\right\} - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j,\mathbf{r}_j,\mathbf{X}_j,\mathbf{Z}_j)$.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`meoprobit` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Flay, B. R., B. R. Brannon, C. A. Johnson, W. B. Hansen, A. L. Ulene, D. A. Whitney-Saltiel, L. R. Gleason, S. Sussman, M. D. Gavin, K. M. Glowacz, D. F. Sobol, and D. C. Spiegel. 1988. The television, school, and family smoking cessation and prevention project: I. Theoretical basis and program development. *Preventive Medicine* 17: 585–607. https://doi.org/10.1016/0091-7435(88)90053-9.

Harbord, R. M., and P. Whiting. 2009. metandi: Meta-analysis of diagnostic accuracy using hierarchical logistic regression. *Stata Journal* 9: 211–229.

Joe, H. 2008. Accuracy of Laplace approximation for discrete response mixed models. *Computational Statistics and Data Analysis* 52: 5066–5074. https://doi.org/10.1016/j.csda.2008.05.002.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

Lin, X., and N. E. Breslow. 1996. Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association* 91: 1007–1016. https://doi.org/10.2307/2291720.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models.* 2nd ed. Hoboken, NJ: Wiley.

McLachlan, G. J., and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering.* New York: Dekker.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Self, S. G., and K.-Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association* 82: 605–610. https://doi.org/10.2307/2289471.

Also see

[ME] **meoprobit postestimation** — Postestimation tools for meoprobit

[ME] **meologit** — Multilevel mixed-effects ordered logistic regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: meoprobit** — Bayesian multilevel ordered probit regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtoprobit** — Random-effects ordered probit model

[U] 20 Estimation and postestimation commands

meoprobit postestimation — Postestimation tools for meoprobit

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation command is of special interest after `meoprobit`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as probabilities, linear predictions, density and distribution functions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`pr`	predicted probabilities; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
`outcome(`*outcome*`)`	outcome category for predicted probabilities
Integration	
int_options	integration options

You specify one or k new variables in *newvars* with `pr`, where k is the number of outcomes. If you do not specify `outcome()`, these options assume `outcome(#1)`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`pr`, the default, calculates the predicted probabilities.

You specify one or k new variables, where k is the number of categories of the dependent variable. If you specify the `outcome()` option, the probabilities will be predicted for the requested outcome only, in which case you specify only one new variable. If you specify one new variable and do not specify `outcome()`, `outcome(#1)` is assumed.

`eta`, `xb`, `stdp`, `density`, `distribution`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`outcome(`*outcome*`)` specifies the outcome for which the predicted probabilities are to be calculated. `outcome()` should contain either one value of the dependent variable or one of `#1`, `#2`, ..., with `#1` meaning the first category of the dependent variable, `#2` meaning the second category, etc.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`, see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
default	probabilities for each outcome
`pr`	predicted probabilities for a specified outcome
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

`pr` defaults to the first outcome.

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.

Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting an ordered probit mixed-effects model using `meoprobit`. Here we show a short example of predicted probabilities and predicted random effects; refer to [ME] **meglm postestimation** for additional examples applicable to mixed-effects generalized linear models.

▷ Example 1: Obtaining predicted probabilities and random effects

In example 2 of [ME] **meoprobit**, we modeled the tobacco and health knowledge (`thk`) score—coded 1, 2, 3, 4—among students as a function of two treatments (`cc` and `tv`) using a three-level ordered probit model with random effects at the school and class levels.

```
. use https://www.stata-press.com/data/r19/tvsfpors
(Television, School, and Family Project)
. meoprobit thk prethk cc##tv || school: || class:
```

(*output omitted*)

We obtain predicted probabilities for all four outcomes based on the contribution of both fixed effects and random effects by typing

```
. predict pr*
(option pr assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

As the note says, the predicted values are based on the posterior means of random effects. You can use the `modes` option to obtain predictions based on the posterior modes of random effects.

Because we specified a stub name, Stata saved the predicted random effects in variables `pr1` through `pr4`. Here we list the predicted probabilities for the first two classes for school 515:

```
. list class thk pr? if school==515 & (class==515101 | class==515102),
> sepby(class)
```

	class	thk	pr1	pr2	pr3	pr4
1464.	515101	2	.1503512	.2416885	.2828209	.3251394
1465.	515101	2	.3750887	.2958534	.2080368	.121021
1466.	515101	1	.3750887	.2958534	.2080368	.121021
1467.	515101	4	.2886795	.2920168	.2433916	.1759121
1468.	515101	3	.2129906	.2729831	.2696254	.2444009
1469.	515101	3	.2886795	.2920168	.2433916	.1759121
1470.	515102	1	.3318574	.2959802	.2261095	.1460529
1471.	515102	2	.4223251	.2916287	.187929	.0981172
1472.	515102	2	.4223251	.2916287	.187929	.0981172
1473.	515102	2	.4223251	.2916287	.187929	.0981172
1474.	515102	2	.3318574	.2959802	.2261095	.1460529
1475.	515102	1	.4223251	.2916287	.187929	.0981172
1476.	515102	2	.3318574	.2959802	.2261095	.1460529

For each observation, our best guess for the predicted outcome is the one with the highest predicted probability. For example, for the very first observation in the table above, we would choose outcome 4 as the most likely to occur.

We obtain predictions of the posterior means themselves at the school and class levels by typing

```
. predict re_s re_c, reffects
(calculating posterior means of random effects)
(using 7 quadrature points)
```

Here we list the predicted random effects for the first two classes for school 515:

```
. list class re_s re_c if school==515 & (class==515101 | class==515102),
> sepby(class)
```

	class	re_s	re_c
1464.	515101	-.0340769	.0390243
1465.	515101	-.0340769	.0390243
1466.	515101	-.0340769	.0390243
1467.	515101	-.0340769	.0390243
1468.	515101	-.0340769	.0390243
1469.	515101	-.0340769	.0390243
1470.	515102	-.0340769	-.0834322
1471.	515102	-.0340769	-.0834322
1472.	515102	-.0340769	-.0834322
1473.	515102	-.0340769	-.0834322
1474.	515102	-.0340769	-.0834322
1475.	515102	-.0340769	-.0834322
1476.	515102	-.0340769	-.0834322

We can see that the predicted random effects at the school level (`re_s`) are the same for all classes and that the predicted random effects at the class level (`re_c`) are constant within each class.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

Also see

[ME] **meoprobit** — Multilevel mixed-effects ordered probit regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

mepoisson — Multilevel mixed-effects Poisson regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`mepoisson` fits mixed-effects models for count responses. The conditional distribution of the response given the random effects is assumed to be Poisson.

Quick start

Without weights

Two-level Poisson regression of `y` on `x` with random intercepts by `lev2`

```
mepoisson y x || lev2:
```

Add `evar` measuring exposure

```
mepoisson y x, exposure(evar) || lev2:
```

Same as above, but report incidence-rate ratios

```
mepoisson y x, exposure(evar) || lev2:, irr
```

Add indicators for levels of categorical variable `a` and random coefficients on `x`

```
mepoisson y x i.a || lev2: x, irr
```

Three-level random-intercept model of `y` on `x` with `lev2` nested within `lev3`

```
mepoisson y x || lev3: || lev2:
```

With weights

Two-level Poisson regression of `y` on `x` with random intercepts by `lev2` and observation-level frequency weights `wvar1`

```
mepoisson y x [fweight=wvar1] || lev2:
```

Two-level random-intercept model from a two-stage sampling design with PSUs identified by `psu` using PSU-level and observation-level sampling weights `wvar2` and `wvar1`, respectively

```
mepoisson y x [pweight=wvar1] || psu:, pweight(wvar2)
```

Add secondary sampling stage with units identified by `ssu` having weights `wvar2` and PSU-level weights `wvar3` for a three-level random-intercept model

```
mepoisson y x [pw=wvar1] || psu:, pw(wvar3) || ssu:, pw(wvar2)
```

Same as above, but `svyset` data first

```
svyset psu, weight(wvar3) || ssu, weight(wvar2) || _n, weight(wvar1)
svy: mepoisson y x || psu: || ssu:
```

Menu

Statistics > Multilevel mixed-effects models > Poisson regression

Syntax

```
mepoisson depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar: R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress the constant term from the fixed-effects equation
`exposure(`*varname*$_e$`)`	include $\ln($*varname*$_e)$ in model with coefficient constrained to 1
`offset(`*varname*$_o$`)`	include *varname*$_o$ in model with coefficient constrained to 1

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`irr`	report fixed-effects coefficients as incidence-rate ratios
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
pcaghermite	Pinheiro–Chao mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models
pclaplace	Pinheiro–Chao Laplacian approximation

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: mepoisson**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

`exposure(`*varname*$_e$`)` specifies a variable that reflects the amount of exposure over which the *depvar* events were observed for each observation; ln(*varname*$_e$) is included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`offset(`*varname*$_o$`)` specifies that *varname*$_o$ be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(#)`; see [R] **Estimation options**.

`irr` reports estimated fixed-effects coefficients transformed to incidence-rate ratios, that is, $\exp(\beta)$ rather than β. Standard errors and confidence intervals are similarly transformed. This option affects how results are displayed, not how they are estimated or stored. `irr` may be specified either at estimation or upon replay.

nocnsreport; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite and pcaghermite perform mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace and pclaplace perform the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point. Techniques pcaghermite and pclaplace obtain the random-effects mode and curvature using the efficient hierarchical decomposition algorithm described in Pinheiro and Chao (2006). For hierarchical models, this algorithm takes advantage of the design structure to minimize memory use and utilizes a series of orthogonal triangulations to compute the factored random-effects Hessian indirectly, avoiding the sparse full Hessian. Techniques mcaghermite and laplace use Cholesky factorization on the full Hessian. For four- and higher-level hierarchical designs, there can be dramatic computation-time differences.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for mepoisson are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with mepoisson but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

For a general introduction to me commands, see [ME] **me**.

Remarks are presented under the following headings:

Introduction
Two-level models
Higher-level models

Introduction

Mixed-effects Poisson regression is Poisson regression containing both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

mepoisson allows for many levels of random effects. However, for simplicity, for now we consider the two-level model, where for a series of M independent clusters, and conditional on a set of random effects $\mathbf{u}_j$,

$$\Pr(y_{ij} = y|\mathbf{x}_{ij}, \mathbf{u}_j) = \exp\left(-\mu_{ij}\right)\mu_{ij}^{y}/y! \tag{1}$$

for $\mu_{ij} = \exp(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j)$, $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The responses are counts y_{ij}. The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard Poisson regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. For notational convenience here and throughout this manual entry, we suppress the dependence of y_{ij} on $\mathbf{x}_{ij}$.

The $1 \times q$ vector $\mathbf{z}_{ij}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$ so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

As noted in section 13.7 of Rabe-Hesketh and Skrondal (2022), the inclusion of a random intercept causes the marginal variance of y_{ij} to be greater than the marginal mean, provided the variance of the random intercept is not 0. Thus the random intercept in a mixed-effects Poisson model produces overdispersion, a measure of variability above and beyond that allowed by a Poisson process; see [R] **nbreg** and [ME] **menbreg**.

Below we present examples of mixed-effects Poisson regression; refer to [ME] **me** and [ME] **meglm** for additional examples including crossed random-effects models. A two-level Poisson model can also be fit using xtpoisson with the re option; see [XT] **xtpoisson**. In the absence of random effects, mixed-effects Poisson regression reduces to standard Poisson regression; see [R] **poisson**.

Two-level models

▷ Example 1: Two-level random-intercept model

Breslow and Clayton (1993) fit a mixed-effects Poisson model to data from a randomized trial of the drug progabide for the treatment of epilepsy.

```
. use https://www.stata-press.com/data/r19/epilepsy
(Epilepsy data; progabide drug treatment)
. describe
Contains data from https://www.stata-press.com/data/r19/epilepsy.dta
 Observations:           236                  Epilepsy data; progabide drug
                                                treatment
    Variables:             8                  31 May 2024 14:09
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
subject         byte    %9.0g                 Subject ID: 1-59
seizures        int     %9.0g                 No. of seizures
treat           byte    %9.0g      treat      Treatment
visit           float   %9.0g                 Doctor's visit
lage            float   %9.0g                 log(age), mean-centered
lbas            float   %9.0g                 log(0.25*baseline seizures),
                                                mean-centered
lbas_trt        float   %9.0g                 lbas/treat interaction
v4              byte    %8.0g                 Fourth visit indicator
------------------------------------------------------------------------------
Sorted by: subject
```

Originally from Thall and Vail (1990), data were collected on 59 subjects (31 on progabide, 28 on placebo). The number of epileptic seizures (`seizures`) was recorded during the two weeks prior to each of four doctor visits (`visit`). The treatment group is identified by the indicator variable `treat`. Data were also collected on the logarithm of age (`lage`) and the logarithm of one-quarter the number of seizures during the eight weeks prior to the study (`lbas`). The variable `lbas_trt` represents the interaction between `lbas` and treatment. `lage`, `lbas`, and `lbas_trt` are mean centered. Because the study originally noted a substantial decrease in seizures prior to the fourth doctor visit, an indicator, `v4`, for the fourth visit was also recorded.

Breslow and Clayton (1993) fit a random-effects Poisson model for the number of observed seizures

$$\log(\mu_{ij}) = \beta_0 + \beta_1 \texttt{treat}_{ij} + \beta_2 \texttt{lbas}_{ij} + \beta_3 \texttt{lbas_trt}_{ij} + \beta_4 \texttt{lage}_{ij} + \beta_5 \texttt{v4}_{ij} + u_j$$

for $j = 1, \ldots, 59$ subjects and $i = 1, \ldots, 4$ visits. The random effects u_j are assumed to be normally distributed with mean 0 and variance σ_u^2.

```
. mepoisson seizures treat lbas lbas_trt lage v4 || subject:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -1016.4106
Iteration 1:  Log likelihood = -819.20112
Iteration 2:  Log likelihood = -817.66006
Iteration 3:  Log likelihood = -817.65925
Iteration 4:  Log likelihood = -817.65925
Refining starting values:
Grid node 0:  Log likelihood = -680.40523
Refining starting values (unscaled likelihoods):
Grid node 0:  Log likelihood = -680.40523
Fitting full model:
Iteration 0:  Log likelihood = -680.40523  (not concave)
Iteration 1:  Log likelihood = -672.95766  (not concave)
Iteration 2:  Log likelihood = -667.14039
Iteration 3:  Log likelihood = -665.51823
Iteration 4:  Log likelihood = -665.29165
Iteration 5:  Log likelihood = -665.29067
Iteration 6:  Log likelihood = -665.29067

Mixed-effects Poisson regression              Number of obs     =        236
Group variable: subject                       Number of groups  =         59
                                              Obs per group:
                                                            min =          4
                                                            avg =        4.0
                                                            max =          4
Integration method: mvaghermite               Integration pts.  =          7
                                              Wald chi2(5)      =     121.70
Log likelihood = -665.29067                   Prob > chi2       =     0.0000
------------------------------------------------------------------------------
    seizures | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
       treat |  -.9330306   .4007512    -2.33   0.020    -1.718489   -.1475727
        lbas |   .8844225   .1312033     6.74   0.000     .6272689    1.141576
    lbas_trt |   .3382561   .2033021     1.66   0.096    -.0602087     .736721
       lage |   .4842226   .3471905     1.39   0.163    -.1962582    1.164703
         v4 |  -.1610871   .0545758    -2.95   0.003    -.2680536   -.0541206
      _cons |   2.154578   .2199928     9.79   0.000       1.7234    2.585756
-------------+----------------------------------------------------------------
subject      |
  var(_cons) |   .2528664   .0589844                      .1600801     .399434
------------------------------------------------------------------------------
LR test vs. Poisson model: chibar2(01) = 304.74       Prob >= chibar2 = 0.0000
```

The number of seizures before the fourth visit does exhibit a significant drop, and the patients on progabide demonstrate a decrease in frequency of seizures compared with the placebo group. The subject-specific random effects also appear significant: $\hat{\sigma}_u^2 = 0.25$ with standard error 0.06.

Because this is a simple random-intercept model, you can obtain equivalent results by using `xtpoisson` with the `re` and `normal` options.

◁

▷ Example 2: Two-level random-slope model

In their study of PQL, Breslow and Clayton (1993) also fit a model where they dropped the fixed effect on v4 and replaced it with a random subject-specific linear trend over the four doctor visits. The model they fit is

$$\log(\mu_{ij}) = \beta_0 + \beta_1 \texttt{treat}_{ij} + \beta_2 \texttt{lbas}_{ij} + \beta_3 \texttt{lbas_trt}_{ij} + \beta_4 \texttt{lage}_{ij} + \beta_5 \texttt{visit}_{ij} + u_j + v_j \texttt{visit}_{ij}$$

where (u_j, v_j) are bivariate normal with 0 mean and variance–covariance matrix

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} \sigma_u^2 & \sigma_{uv} \\ \sigma_{uv} & \sigma_v^2 \end{bmatrix}$$

```
. mepoisson seizures treat lbas lbas_trt lage visit || subject: visit,
> covariance(unstructured) intpoints(9) nolog
```

```
Mixed-effects Poisson regression                Number of obs     =        236
Group variable: subject                         Number of groups  =         59

                                                Obs per group:
                                                              min =          4
                                                              avg =        4.0
                                                              max =          4

Integration method: mvaghermite                 Integration pts.  =          9

                                                Wald chi2(5)      =     115.56
Log likelihood = -655.68103                     Prob > chi2       =     0.0000
```

seizures	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
treat	-.9286592	.4021715	-2.31	0.021	-1.716901	-.1404175
lbas	.8849762	.1312535	6.74	0.000	.627724	1.142228
lbas_trt	.3379759	.2044471	1.65	0.098	-.062733	.7386849
lage	.4767192	.3536276	1.35	0.178	-.2163781	1.169817
visit	-.2664098	.1647098	-1.62	0.106	-.5892352	.0564156
_cons	2.099555	.2203749	9.53	0.000	1.667629	2.531482
subject						
var(visit)	.5314803	.229385			.2280928	1.238405
var(_cons)	.2514923	.0587902			.1590534	.3976549
subject						
cov(visit, _cons)	.0028715	.0887037	0.03	0.974	-.1709846	.1767276

```
LR test vs. Poisson model: chi2(3) = 324.54                   Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

In the above, we specified the `covariance(unstructured)` option to allow correlation between u_j and v_j, although on the basis of the above output it probably was not necessary—the default `independent` structure would have sufficed. In the interest of getting more accurate estimates, we also increased the number of quadrature points to nine, although the estimates do not change much when compared with estimates based on the default seven quadrature points.

The essence of the above-fitted model is that after adjusting for other covariates, the log trend in seizures is modeled as a random subject-specific line, with intercept distributed as $N(\beta_0, \sigma_u^2)$ and slope distributed as $N(\beta_5, \sigma_v^2)$. From the above output, $\hat{\beta}_0 = 2.10$, $\hat{\sigma}_u^2 = 0.25$, $\hat{\beta}_5 = -0.27$, and $\hat{\sigma}_v^2 = 0.53$.

You can predict the random effects u_j and v_j by using `predict` after `mepoisson`; see [ME] **mepoisson postestimation**. Better still, you can obtain a predicted number of seizures that takes these random effects into account.

◁

Higher-level models

▷ Example 3: Three- and four-level random-intercept model

Rabe-Hesketh and Skrondal (2022, exercise 13.7) describe data from the *Atlas of Cancer Mortality in the European Economic Community* (EEC) (Smans, Mair, and Boyle 1992). The data were analyzed in Langford, Bentham, and McDonald (1998) and record the number of deaths among males due to malignant melanoma during 1971–1980.

```
. use https://www.stata-press.com/data/r19/melanoma
(Skin cancer (melanoma) data)

. describe
Contains data from https://www.stata-press.com/data/r19/melanoma.dta
 Observations:           354                  Skin cancer (melanoma) data
    Variables:             6                  30 May 2024 17:10
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
nation          byte    %11.0g     n          Nation ID
region          byte    %9.0g                 Region ID: EEC level-I areas
county          int     %9.0g                 County ID: EEC level-II/level-III
                                                areas
deaths          int     %9.0g                 No. deaths during 1971-1980
expected        float   %9.0g                 No. expected deaths
uv              float   %9.0g                 UV dose, mean-centered
------------------------------------------------------------------------------
Sorted by:
```

Nine European nations (variable `nation`) are represented, and data were collected over geographical regions defined by EEC statistical services as level I areas (variable `region`), with deaths being recorded for each of 354 counties, which are level II or level III EEC-defined areas (variable `county`, which identifies the observations). Counties are nested within regions, and regions are nested within nations.

The variable `deaths` records the number of deaths for each county, and `expected` records the expected number of deaths (the exposure) on the basis of crude rates for the combined countries. Finally, the variable `uv` is a measure of exposure to ultraviolet (UV) radiation.

In modeling the number of deaths, one possibility is to include dummy variables for the nine nations as fixed effects. Another is to treat these as random effects and fit the three-level random-intercept Poisson model,

$$\log(\mu_{ijk}) = \log(\texttt{expected}_{ijk}) + \beta_0 + \beta_1 \texttt{uv}_{ijk} + u_k + v_{jk}$$

for nation k, region j, and county i. The model includes an exposure term for expected deaths.

```
. mepoisson deaths uv, exposure(expected) || nation: || region:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -2136.5847
Iteration 1:  Log likelihood = -1723.8955
Iteration 2:  Log likelihood = -1723.7727
Iteration 3:  Log likelihood = -1723.7727
Refining starting values:
Grid node 0:  Log likelihood = -1166.6536
Refining starting values (unscaled likelihoods):
Grid node 0:  Log likelihood = -1166.6536
Fitting full model:
Iteration 0:  Log likelihood = -1166.6536  (not concave)
Iteration 1:  Log likelihood = -1152.2741  (not concave)
Iteration 2:  Log likelihood = -1146.3094  (not concave)
Iteration 3:  Log likelihood = -1119.8479  (not concave)
Iteration 4:  Log likelihood = -1108.0129  (not concave)
Iteration 5:  Log likelihood = -1098.8067
Iteration 6:  Log likelihood = -1095.7563
Iteration 7:  Log likelihood = -1095.3164
Iteration 8:  Log likelihood =   -1095.31
Iteration 9:  Log likelihood =   -1095.31
Mixed-effects Poisson regression               Number of obs     =        354
        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
         nation |          9          3       39.3         95
         region |         78          1        4.5         13
------------------------------------------------------------------------------
Integration method: mvaghermite                Integration pts.  =          7
                                               Wald chi2(1)      =       6.12
Log likelihood = -1095.31                      Prob > chi2       =     0.0134
------------------------------------------------------------------------------
      deaths | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
          uv |  -.0282041   .0113998    -2.47   0.013    -.0505473   -.0058608
       _cons |  -.0639672   .1335515    -0.48   0.632    -.3257234    .197789
ln(expected) |          1  (exposure)
-------------+----------------------------------------------------------------
nation       |
   var(_cons)|   .1371732   .0723303                      .048802    .3855676
-------------+----------------------------------------------------------------
nation>      |
region       |
   var(_cons)|   .0483483   .0109079                     .0310699    .0752353
------------------------------------------------------------------------------
LR test vs. Poisson model: chi2(2) = 1256.93                  Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

By including an exposure variable that is an expected rate, we are in effect specifying a linear model for the log of the standardized mortality ratio, the ratio of observed deaths to expected deaths that is based on a reference population. Here the reference population is all nine nations.

Looking at the estimated variance components, we can see there is more unobserved variability between nations than between regions within each nation. This may be due to, for example, country-specific informational campaigns on the risks of sun exposure.

We now add a random intercept for counties nested within regions, making this a four-level model. Because counties also identify the observations, the corresponding variance component can be interpreted as a measure of overdispersion, variability above and beyond that allowed by a Poisson process; see [R] **nbreg** and [ME] **menbreg**.

```
. mepoisson deaths uv, exposure(expected) || nation: || region: || county:,
> intmethod(mcaghermite)
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -2136.5847
Iteration 1:  Log likelihood = -1723.8955
Iteration 2:  Log likelihood = -1723.7727
Iteration 3:  Log likelihood = -1723.7727
Refining starting values:
Grid node 0:  Log likelihood = -1379.3466
Refining starting values (unscaled likelihoods):
Grid node 0:  Log likelihood = -1379.3466
Fitting full model:
Iteration 0:  Log likelihood = -1379.3466  (not concave)
Iteration 1:  Log likelihood = -1310.4947  (not concave)
Iteration 2:  Log likelihood =  -1245.534  (not concave)
Iteration 3:  Log likelihood = -1218.5474  (not concave)
Iteration 4:  Log likelihood =  -1207.881  (not concave)
Iteration 5:  Log likelihood = -1122.0585  (not concave)
Iteration 6:  Log likelihood = -1092.4049
Iteration 7:  Log likelihood = -1088.0486
Iteration 8:  Log likelihood = -1086.7175
Iteration 9:  Log likelihood = -1086.6756
Iteration 10: Log likelihood = -1086.6754
Iteration 11: Log likelihood = -1086.6754
Mixed-effects Poisson regression               Number of obs     =        354
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
nation	9	3	39.3	95
region	78	1	4.5	13
county	354	1	1.0	1

```
Integration method: mcaghermite                 Integration pts.  =          7
                                                Wald chi2(1)      =       8.62
Log likelihood = -1086.6754                     Prob > chi2       =     0.0033
```

deaths	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
uv	-.0334702	.0113968	-2.94	0.003	-.0558075	-.0111329
_cons	-.0864583	.1299275	-0.67	0.506	-.3411115	.168195
ln(expected)	1	(exposure)				
nation						
var(_cons)	.1288627	.0681643			.0456949	.3634011
nation> region						
var(_cons)	.0406279	.0105154			.0244633	.0674735
nation> region> county						
var(_cons)	.0146672	.0050979			.0074215	.0289867

```
LR test vs. Poisson model: chi2(3) = 1274.19                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

In the above, we used `intmethod(mcaghermite)`, which is not only faster but also produces estimates that closely agree with those obtained with the default `mvaghermite` integration method.

◁

Stored results

`mepoisson` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_dv)`	number of dependent variables
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_f)`	number of fixed-effects parameters
`e(k_r)`	number of random-effects parameters
`e(k_rs)`	number of variances
`e(k_rc)`	number of covariances
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(N_clust)`	number of clusters
`e(chi2)`	χ^2
`e(p)`	p-value for model test
`e(ll_c)`	log likelihood, comparison model
`e(chi2_c)`	χ^2, comparison test
`e(df_c)`	degrees of freedom, comparison test
`e(p_c)`	p-value for comparison test
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	mepoisson
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	poisson
e(title)	title in estimation output
e(link)	log
e(family)	poisson
e(clustvar)	name of cluster variable
e(offset)	offset
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, *p*-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

`mepoisson` is a convenience command for `meglm` with a `log` link and an `poisson` family; see [ME] **meglm**.

In a two-level Poisson model, for cluster j, $j = 1, \ldots, M$, the conditional distribution of $\mathbf{y}_j = (y_{j1}, \ldots, y_{jn_j})'$, given a set of cluster-level random effects $\mathbf{u}_j$, is

$$\begin{aligned} f(\mathbf{y}_j|\mathbf{u}_j) &= \prod_{i=1}^{n_j} \left[\left\{ \exp\left(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j\right) \right\}^{y_{ij}} \exp\left\{ -\exp\left(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j\right) \right\} / y_{ij}! \right] \\ &= \exp\left[\sum_{i=1}^{n_j} \left\{ y_{ij}\left(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j\right) - \exp\left(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j\right) - \log(y_{ij}!) \right\} \right] \end{aligned}$$

Defining $c\left(\mathbf{y}_j\right) = \sum_{i=1}^{n_j} \log(y_{ij}!)$, where $c(\mathbf{y}_j)$ does not depend on the model parameters, we can express the above compactly in matrix notation,

$$f(\mathbf{y}_j|\mathbf{u}_j) = \exp\left\{ \mathbf{y}_j'\left(\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j\right) - \mathbf{1}'\exp\left(\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j\right) - c\left(\mathbf{y}_j\right) \right\}$$

where $\mathbf{X}_j$ is formed by stacking the row vectors $\mathbf{x}_{ij}$ and $\mathbf{Z}_j$ is formed by stacking the row vectors $\mathbf{z}_{ij}$. We extend the definition of $\exp(\cdot)$ to be a vector function where necessary.

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j, \mathbf{u}_j)$,

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2} |\boldsymbol{\Sigma}|^{-1/2} \int f(\mathbf{y}_j|\mathbf{u}_j) \exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right) d\mathbf{u}_j \\ &= \exp\left\{-c\left(\mathbf{y}_j\right)\right\} (2\pi)^{-q/2} |\boldsymbol{\Sigma}|^{-1/2} \int \exp\left\{h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right)\right\} d\mathbf{u}_j \end{aligned} \tag{2}$$

where

$$h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right) = \mathbf{y}_j'\left(\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j\right) - \mathbf{1}'\exp\left(\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j\right) - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j, \mathbf{X}_j, \mathbf{Z}_j)$.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`mepoisson` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Breslow, N. E., and D. G. Clayton. 1993. Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association* 88: 9–25. https://doi.org/10.2307/2290687.

Joe, H. 2008. Accuracy of Laplace approximation for discrete response mixed models. *Computational Statistics and Data Analysis* 52: 5066–5074. https://doi.org/10.1016/j.csda.2008.05.002.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

Langford, I. H., G. Bentham, and A. McDonald. 1998. Multi-level modelling of geographically aggregated health data: A case study on malignant melanoma mortality and UV exposure in the European community. *Statistics in Medicine* 17: 41–57. https://doi.org/10.1002/(SICI)1097-0258(19980115)17:1<41::AID-SIM712>3.0.CO;2-0.

Leyland, A. H., and H. Goldstein, eds. 2001. *Multilevel Modelling of Health Statistics*. New York: Wiley.

Lin, X., and N. E. Breslow. 1996. Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association* 91: 1007–1016. https://doi.org/10.2307/2291720.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models*. 2nd ed. Hoboken, NJ: Wiley.

McLachlan, G. J., and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. New York: Dekker.

Pinheiro, J. C., and E. C. Chao. 2006. Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models. *Journal of Computational and Graphical Statistics* 15: 58–81. https://doi.org/10.1198/106186006X96962.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2005. Maximum likelihood estimation of limited and discrete dependent variable models with nested random effects. *Journal of Econometrics* 128: 301–323. https://doi.org/10.1016/j.jeconom.2004.08.017.

Raudenbush, S. W., and A. S. Bryk. 2002. *Hierarchical Linear Models: Applications and Data Analysis Methods*. 2nd ed. Thousand Oaks, CA: Sage.

Self, S. G., and K.-Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association* 82: 605–610. https://doi.org/10.2307/2289471.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman and Hall/CRC.

Smans, M., C. S. Mair, and P. Boyle, eds. 1992. *Atlas of Cancer Mortality in the European Economic Community*. Lyon: IARC.

Thall, P. F., and S. C. Vail. 1990. Some covariance models for longitudinal count data with overdispersion. *Biometrics* 46: 657–671. https://doi.org/10.2307/2532086.

Also see

[ME] **mepoisson postestimation** — Postestimation tools for mepoisson

[ME] **menbreg** — Multilevel mixed-effects negative binomial regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: mepoisson** — Bayesian multilevel Poisson regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtpoisson** — Fixed-effects, random-effects, and population-averaged Poisson models

[U] 20 Estimation and postestimation commands

mepoisson postestimation — Postestimation tools for mepoisson

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation command is of special interest after `mepoisson`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as mean responses; linear predictions; density and distribution functions; standard errors; and Pearson, deviance, and Anscombe residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function
`pearson`	Pearson residuals
`deviance`	deviance residuals
`anscombe`	Anscombe residuals

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

`pearson`, `deviance`, `anscombe` may not be combined with `marginal`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`mu`, the default, calculates the predicted mean, that is, the predicted number of events.

`eta`, `xb`, `stdp`, `density`, `distribution`, `pearson`, `deviance`, `anscombe`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, and `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for mean responses and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* `...)` [`predict(`*statistic* `...)` `...`] [*options*]

statistic	Description
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`pearson`	not allowed with `margins`
`deviance`	not allowed with `margins`
`anscombe`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.

Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects Poisson model with `mepoisson`. For the most part, calculation centers around obtaining estimates of the subject/group-specific random effects. Random effects are not estimated when the model is fit but instead need to be predicted after estimation.

Here we show a short example of predicted counts and predicted random effects; refer to [ME] **meglm postestimation** for additional examples applicable to mixed-effects generalized linear models.

Example 1: Predicting counts and random effects

In example 2 of [ME] **mepoisson**, we modeled the number of observed epileptic seizures as a function of treatment with the drug progabide and other covariates,

$$\log(\mu_{ij}) = \beta_0 + \beta_1 \texttt{treat}_{ij} + \beta_2 \texttt{lbas}_{ij} + \beta_3 \texttt{lbas_trt}_{ij} + \beta_4 \texttt{lage}_{ij} + \beta_5 \texttt{visit}_{ij} + u_j + v_j \texttt{visit}_{ij}$$

where (u_j, v_j) are bivariate normal with 0 mean and variance–covariance matrix

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} \sigma_u^2 & \sigma_{uv} \\ \sigma_{uv} & \sigma_v^2 \end{bmatrix}$$

```
. use https://www.stata-press.com/data/r19/epilepsy
(Epilepsy data; progabide drug treatment)
. mepoisson seizures treat lbas lbas_trt lage visit || subject: visit,
> cov(unstructured) intpoints(9)
```

(*iteration log omitted*)

```
Mixed-effects Poisson regression                Number of obs     =        236
Group variable: subject                         Number of groups  =         59
                                                Obs per group:
                                                              min =          4
                                                              avg =        4.0
                                                              max =          4
Integration method: mvaghermite                 Integration pts.  =          9
                                                Wald chi2(5)      =     115.56
Log likelihood = -655.68103                     Prob > chi2       =     0.0000
```

seizures	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
treat	-.9286592	.4021715	-2.31	0.021	-1.716901	-.1404175
lbas	.8849762	.1312535	6.74	0.000	.627724	1.142228
lbas_trt	.3379759	.2044471	1.65	0.098	-.062733	.7386849
lage	.4767192	.3536276	1.35	0.178	-.2163781	1.169817
visit	-.2664098	.1647098	-1.62	0.106	-.5892352	.0564156
_cons	2.099555	.2203749	9.53	0.000	1.667629	2.531482
subject						
var(visit)	.5314803	.229385			.2280928	1.238405
var(_cons)	.2514923	.0587902			.1590534	.3976549
subject						
cov(visit, _cons)	.0028715	.0887037	0.03	0.974	-.1709846	.1767276

```
LR test vs. Poisson model: chi2(3) = 324.54                   Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

The purpose of this model was to allow subject-specific linear log trends over each subject's four doctor visits, after adjusting for the other covariates. The intercepts of these lines are distributed $N(\beta_0, \sigma_u^2)$, and the slopes are distributed $N(\beta_5, \sigma_v^2)$, based on the fixed effects and assumed distribution of the random effects.

We can use `predict` to obtain estimates of the random effects u_j and v_j and combine these with our estimates of β_0 and β_5 to obtain the intercepts and slopes of the linear log trends.

```
. predict re_visit re_cons, reffects
(calculating posterior means of random effects)
(using 9 quadrature points)
. generate b1 = _b[visit] + re_visit
. generate b0 = _b[_cons] + re_cons
. by subject, sort: generate tolist = _n==1
. list subject treat b1 b0 if tolist & (subject <=5 | subject >=55)
```

	subject	treat	b1	b0
1.	1	Placebo	-.428854	2.13539
5.	2	Placebo	-.2731013	2.149744
9.	3	Placebo	.0022089	2.417506
13.	4	Placebo	-.3197094	2.238224
17.	5	Placebo	.6082718	2.110739
217.	55	Progabide	-.2308834	2.282539
221.	56	Progabide	.2912798	3.19678
225.	57	Progabide	-.4828764	1.423153
229.	58	Progabide	-.2519466	1.131373
233.	59	Progabide	-.1269573	2.171541

We list these slopes (`b1`) and intercepts (`b0`) for five control subjects and five subjects on the treatment.

```
. count if tolist & treat
  31
. count if tolist & treat & b1 < 0
  25
. count if tolist & !treat
  28
. count if tolist & !treat & b1 < 0
  20
```

We also find that 25 of the 31 subjects taking progabide were estimated to have a downward trend in seizures over their four doctor visits, compared with 20 of the 28 control subjects.

We also obtain predictions for number of seizures, and unless we specify the `conditional(fixedonly)` option, these predictions will incorporate the estimated subject-specific random effects.

```
. predict n
(option mu assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 9 quadrature points)
```

```
. list subject treat visit seizures n if subject <= 2 | subject >= 58, sep(0)
```

	subject	treat	visit	seizures	n
1.	1	Placebo	-.3	5	3.775774
2.	1	Placebo	-.1	3	3.465422
3.	1	Placebo	.1	3	3.18058
4.	1	Placebo	.3	3	2.919151
5.	2	Placebo	-.3	3	3.598805
6.	2	Placebo	-.1	5	3.40751
7.	2	Placebo	.1	3	3.226382
8.	2	Placebo	.3	3	3.054883
229.	58	Progabide	-.3	0	.9611137
230.	58	Progabide	-.1	0	.9138838
231.	58	Progabide	.1	0	.8689747
232.	58	Progabide	.3	0	.8262726
233.	59	Progabide	-.3	1	2.40652
234.	59	Progabide	-.1	4	2.346184
235.	59	Progabide	.1	3	2.287361
236.	59	Progabide	.3	2	2.230013

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

Also see

[ME] **mepoisson** — Multilevel mixed-effects Poisson regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

meprobit — Multilevel mixed-effects probit regression

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`meprobit` fits mixed-effects models for binary or binomial responses. The conditional distribution of the response given the random effects is assumed to be Bernoulli, with success probability determined by the standard normal cumulative distribution function.

Quick start

Two-level probit model of `y` and covariate `x` and random intercepts by `lev2`

```
meprobit y x || lev2:
```

Add random coefficients for `x`

```
meprobit y x || lev2: x
```

Same as above, but specify that `y` records the number of successes from 10 trials

```
meprobit y x || lev2: x, binomial(10)
```

Same as above, but with the number of trials stored in variable `n`

```
meprobit y x || lev2: x, binomial(n)
```

Three-level random-intercept model of `y` and covariate `x` with `lev2` nested within `lev3`

```
meprobit y x || lev3: || lev2:
```

Two-way crossed random effects by factors `a` and `b`

```
meprobit y x || _all:R.a || b:
```

Menu

Statistics > Multilevel mixed-effects models > Probit regression

Syntax

```
meprobit depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress constant term from the fixed-effects equation
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1
`asis`	retain perfect predictor variables

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`binomial(`*varname* | #`)`	set binomial trials if data are in binomial form
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid[(`*gridspec*`)]`	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: meprobit**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

`offset(`*varname*`)` specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`asis` forces retention of perfect predictor variables and their associated, perfectly predicted observations and may produce instabilities in maximization; see [R] **probit**.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of

matname are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if *matname*$[i, j]$ = *matname*$[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`binomial(`*varname* | #`)` specifies that the data are in binomial form; that is, *depvar* records the number of successes from a series of binomial trials. This number of trials is given either as *varname*, which allows this number to vary over the observations, or as the constant #. If `binomial()` is not specified (the default), *depvar* is treated as Bernoulli, with any nonzero, nonmissing values indicating positive responses.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(#)`, `nocnsreport`; see [R] **Estimation options**.

`notable` suppresses the estimation table, either at estimation or upon replay.

`noheader` suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite performs mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for meprobit are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with meprobit but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

Mixed-effects probit regression is probit regression containing both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

`meprobit` allows for many levels of random effects. However, for simplicity, we here consider the two-level model, where for a series of M independent clusters, and conditional on a set of fixed effects $\mathbf{x}_{ij}$ and a set of random effects $\mathbf{u}_j$,

$$\Pr(y_{ij} = 1|\mathbf{x}_{ij}, \mathbf{u}_j) = H(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j) \tag{1}$$

for $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The responses are the binary-valued y_{ij}, and we follow the standard Stata convention of treating $y_{ij} = 1$ if $depvar_{ij} \neq 0$ and treating $y_{ij} = 0$ otherwise. The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard probit regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. For notational convenience here and throughout this manual entry, we suppress the dependence of y_{ij} on $\mathbf{x}_{ij}$.

The $1 \times q$ vector $\mathbf{z}_{ij}$ are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ij}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{z}_{ij} = \mathbf{x}_{ij}$, so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

Finally, because this is probit regression, $H(\cdot)$ is the standard normal cumulative distribution function, which maps the linear predictor to the probability of a success ($y_{ij} = 1$) with $H(v) = \Phi(v)$.

Model (1) may also be stated in terms of a latent linear response, where only $y_{ij} = I(y^*_{ij} > 0)$ is observed for the latent

$$y^*_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij}$$

The errors ϵ_{ij} are distributed as a standard normal with mean 0 and variance 1 and are independent of $\mathbf{u}_j$.

Below we present two short examples of mixed-effects probit regression; refer to [ME] **me** and [ME] **meglm** for examples of other random-effects models. A two-level probit model can also be fit using `xtprobit` with the `re` option; see [XT] **xtprobit**. In the absence of random effects, mixed-effects probit regression reduces to standard probit regression; see [R] **probit**.

▷ Example 1: Two-level random-intercept model

Ng et al. (2006) analyzed a subsample of data from the 1989 Bangladesh fertility survey (Huq and Cleland 1990), which polled 1,934 Bangladeshi women on their use of contraception. The women sampled were from 60 districts, identified by the variable `district`. Each district contained either urban or rural areas (variable `urban`) or both. The variable `c_use` is the binary response, with a value of 1 indicating contraceptive use. Other covariates include mean-centered `age` and three indicator variables recording number of children.

```
. use https://www.stata-press.com/data/r19/bangladesh
(Bangladesh Fertility Survey, 1989)
. meprobit c_use i.urban age i.children || district:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -1228.8313
Iteration 1:  Log likelihood = -1228.2466
Iteration 2:  Log likelihood = -1228.2466
Refining starting values:
Grid node 0:  Log likelihood = -1237.3973
Fitting full model:
Iteration 0:  Log likelihood = -1237.3973  (not concave)
Iteration 1:  Log likelihood = -1221.2111  (not concave)
Iteration 2:  Log likelihood = -1207.4451
Iteration 3:  Log likelihood = -1206.7002
Iteration 4:  Log likelihood = -1206.5346
Iteration 5:  Log likelihood = -1206.5336
Iteration 6:  Log likelihood = -1206.5336
Mixed-effects probit regression                 Number of obs     =      1,934
Group variable: district                        Number of groups  =         60
                                                Obs per group:
                                                              min =          2
                                                              avg =       32.2
                                                              max =        118
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(5)      =     115.36
Log likelihood = -1206.5336                     Prob > chi2       =     0.0000
------------------------------------------------------------------------------
       c_use | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
       urban |
      Urban  |   .4490191   .0727176     6.17   0.000     .3064953    .5915429
         age |  -.0162203   .0048005    -3.38   0.001    -.0256291   -.0068114
             |
    children |
    1 child  |    .674377   .0947829     7.11   0.000      .488606    .8601481
 2 children  |   .8281581   .1048136     7.90   0.000      .6227272   1.033589
3 or more..  |   .8137876   .1073951     7.58   0.000      .6032972   1.024278
             |
       _cons |   -1.02799   .0870307   -11.81   0.000    -1.198567   -.8574132
-------------+----------------------------------------------------------------
district     |
   var(_cons)|   .0798719    .026886                      .0412921    .1544972
------------------------------------------------------------------------------
LR test vs. probit model: chibar2(01) = 43.43         Prob >= chibar2 = 0.0000
```

Probit regression coefficients are most commonly interpreted in terms of partial effects, as we demonstrate in example 1 of [ME] **meprobit postestimation**. For now, we only note that urban women and women with more children are more likely to use contraceptives and that contraceptive use decreases with age. The estimated variance of the random intercept at the district level, $\widehat{\sigma}^2$, is 0.08 with standard error 0.03. The reported likelihood-ratio test shows that there is enough variability between districts to favor a mixed-effects probit regression over an ordinary probit regression; see *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

◁

▷ Example 2: Three-level random-intercept model

Rabe-Hesketh, Toulopoulou, and Murray (2001) analyzed data from a study that measured the cognitive ability of patients with schizophrenia compared with their relatives and control subjects. Cognitive ability was measured as the successful completion of the "Tower of London", a computerized task, measured at three levels of difficulty. For all but one of the 226 subjects, there were three measurements (one for each difficulty level). Because patients' relatives were also tested, a family identifier, `family`, was also recorded.

We fit a probit model with response `dtlm`, the indicator of cognitive function, and with covariates `difficulty` and a set of indicator variables for `group`, with the controls (`group==1`) being the base category. We also allow for random effects due to families and due to subjects within families. The first is a random intercept (constant only) at the `family` level, and the second is a random intercept at the `subject` level. The order in which these are specified (from left to right) is significant—`meprobit` assumes that `subject` is nested within `family`. The equations are separated by `||`.

```
. use https://www.stata-press.com/data/r19/towerlondon
(Tower of London data)
. meprobit dtlm difficulty i.group || family: || subject:
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -317.11238
Iteration 1:  Log likelihood = -314.50535
Iteration 2:  Log likelihood = -314.50121
Iteration 3:  Log likelihood = -314.50121
Refining starting values:
Grid node 0:  Log likelihood = -326.18533
Fitting full model:
Iteration 0:  Log likelihood = -326.18533  (not concave)
Iteration 1:  Log likelihood = -313.16256  (not concave)
Iteration 2:  Log likelihood = -308.47528
Iteration 3:  Log likelihood = -305.02228
Iteration 4:  Log likelihood = -304.88972
Iteration 5:  Log likelihood = -304.88845
Iteration 6:  Log likelihood = -304.88845
Mixed-effects probit regression                 Number of obs     =        677
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
family	118	2	5.7	27
subject	226	2	3.0	3

```
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(3)      =      83.28
Log likelihood = -304.88845                     Prob > chi2       =     0.0000
```

dtlm	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
difficulty	-.9329891	.1037376	-8.99	0.000	-1.136311	-.7296672
group						
2	-.1632243	.204265	-0.80	0.424	-.5635763	.2371276
3	-.6220196	.228063	-2.73	0.006	-1.069015	-.1750244
_cons	-.8405154	.1597223	-5.26	0.000	-1.153565	-.5274654
family						
var(_cons)	.2120948	.1736281			.0426292	1.055244
family>subject						
var(_cons)	.3559141	.219331			.106364	1.190956

```
LR test vs. probit model: chi2(2) = 19.23                 Prob > chi2 = 0.0001
Note: LR test is conservative and provided only for reference.
```

We see that we have 226 subjects from 118 families. After adjusting for the random-effects structure, the probability of successful completion of the Tower of London decreases dramatically as the level of difficulty increases. Also, people with schizophrenia (`group==3`) tended not to perform as well as the control subjects.

◁

The above extends to models with more than two levels of nesting by adding more random-effects equations, each separated by ||.

Stored results

meprobit stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_dv)	number of dependent variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(N_clust)	number of clusters
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	meprobit
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	probit
e(title)	title in estimation output
e(link)	probit
e(family)	bernoulli or binomial
e(clustvar)	name of cluster variable
e(offset)	offset
e(binomial)	binomial number of trials
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method

e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved
Matrices	
e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
Functions	
e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices	
r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

meprobit is a convenience command for meglm with a probit link and a bernoulli or binomial family; see [ME] **meglm**.

Model (1) assumes Bernoulli data, a special case of the binomial. Because binomial data are also supported by meprobit (option binomial()), the methods presented below are in terms of the more general binomial mixed-effects model.

For a two-level binomial model, consider the response y_{ij} as the number of successes from a series of r_{ij} Bernoulli trials (replications). For cluster j, $j = 1, \ldots, M$, the conditional distribution of $\mathbf{y}_j = (y_{j1}, \ldots, y_{jn_j})'$, given a set of cluster-level random effects $\mathbf{u}_j$, is

$$
\begin{aligned}
f(\mathbf{y}_j|\mathbf{u}_j) &= \prod_{i=1}^{n_j} \left[\binom{r_{ij}}{y_{ij}} \left\{\Phi(\boldsymbol{\eta}_{ij})\right\}^{y_{ij}} \left\{1-\Phi(\boldsymbol{\eta}_{ij})\right\}^{r_{ij}-y_{ij}} \right] \\
&= \exp\left(\sum_{i=1}^{n_j} \left[y_{ij} \log\left\{\Phi(\boldsymbol{\eta}_{ij})\right\} - (r_{ij}-y_{ij}) \log\left\{\Phi(-\boldsymbol{\eta}_{ij})\right\} + \log\binom{r_{ij}}{y_{ij}} \right] \right)
\end{aligned}
$$

for $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \text{offset}_{ij}$.

Defining $\mathbf{r}_j = (r_{j1}, \ldots, r_{jn_j})'$ and

$$c\left(\mathbf{y}_j, \mathbf{r}_j\right) = \sum_{i=1}^{n_j} \log \binom{r_{ij}}{y_{ij}}$$

where $c(\mathbf{y}_j, \mathbf{r}_j)$ does not depend on the model parameters, we can express the above compactly in matrix notation,

$$f(\mathbf{y}_j|\mathbf{u}_j) = \exp\left[\mathbf{y}_j' \log\left\{\Phi(\boldsymbol{\eta}_j)\right\} - (\mathbf{r}_j - \mathbf{y}_j)' \log\left\{\Phi(-\boldsymbol{\eta}_j)\right\} + c\left(\mathbf{y}_j, \mathbf{r}_j\right)\right]$$

where $\boldsymbol{\eta}_j$ is formed by stacking the row vectors $\boldsymbol{\eta}_{ij}$. We extend the definitions of $\Phi(\cdot)$, $\log(\cdot)$, and $\exp(\cdot)$ to be vector functions where necessary.

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{y}_j, \mathbf{u}_j)$,

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2} \left|\boldsymbol{\Sigma}\right|^{-1/2} \int f(\mathbf{y}_j|\mathbf{u}_j) \exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right) d\mathbf{u}_j \\ &= \exp\left\{c\left(\mathbf{y}_j, \mathbf{r}_j\right)\right\} (2\pi)^{-q/2} \left|\boldsymbol{\Sigma}\right|^{-1/2} \int \exp\left\{h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right)\right\} d\mathbf{u}_j \end{aligned} \tag{2}$$

where

$$h\left(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j\right) = \mathbf{y}_j' \log\left\{\Phi(\boldsymbol{\eta}_j)\right\} - (\mathbf{r}_j - \mathbf{y}_j)' \log\left\{\Phi(-\boldsymbol{\eta}_j)\right\} - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2$$

and for convenience, in the arguments of $h(\cdot)$ we suppress the dependence on the observable data $(\mathbf{y}_j, \mathbf{r}_j, \mathbf{X}_j, \mathbf{Z}_j)$.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`meprobit` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

References

Huq, N. M., and J. Cleland. 1990. *Bangladesh Fertility Survey 1989 (Main Report)*. National Institute of Population Research and Training.

Ng, E. S.-W., J. R. Carpenter, H. Goldstein, and J. Rasbash. 2006. Estimation in generalised linear mixed models with binary outcomes by simulated maximum likelihood. *Statistical Modelling* 6: 23–42. https://doi.org/10.1191/1471082X06st106oa.

Rabe-Hesketh, S., T. Toulopoulou, and R. M. Murray. 2001. Multilevel modeling of cognitive function in schizophrenic patients and their first degree relatives. *Multivariate Behavioral Research* 36: 279–298. https://doi.org/10.1207/S15327906MBR3602_07.

Also see

[ME] **meprobit postestimation** — Postestimation tools for meprobit

[ME] **mecloglog** — Multilevel mixed-effects complementary log–log regression

[ME] **melogit** — Multilevel mixed-effects logistic regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: meprobit** — Bayesian multilevel probit regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtprobit** — Random-effects and population-averaged probit models

[U] 20 Estimation and postestimation commands

meprobit postestimation — Postestimation tools for meprobit

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation commands are of special interest after `meprobit`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as mean responses; linear predictions; density and distribution functions; standard errors; and Pearson, deviance, and Anscombe residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`density`	predicted density function
`distribution`	predicted distribution function
`pearson`	Pearson residuals
`deviance`	deviance residuals
`anscombe`	Anscombe residuals

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

`pearson`, `deviance`, `anscombe` may not be combined with `marginal`.

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`mu`, the default, calculates the predicted mean, that is, the probability of a positive outcome.

`eta`, `xb`, `stdp`, `density`, `distribution`, `pearson`, `deviance`, `anscombe`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, and `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for mean responses and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* `...)` [`predict(`*statistic* `...)` `...`] [*options*]

statistic	Description
`mu`	mean response; the default
`eta`	fitted linear predictor
`xb`	linear predictor for the fixed portion of the model only
`stdp`	not allowed with `margins`
`density`	not allowed with `margins`
`distribution`	not allowed with `margins`
`pearson`	not allowed with `margins`
`deviance`	not allowed with `margins`
`anscombe`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Options `conditional(ebmeans)` and `conditional(ebmodes)` are not allowed with `margins`.

Option `marginal` is assumed where applicable if `conditional(fixedonly)` is not specified.

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects probit model using meprobit. Here we show a short example of predicted probabilities and predicted random effects; refer to [ME] **meglm postestimation** for additional examples.

Example 1: Predicting random effects and estimating intraclass correlations

In example 2 of [ME] **meprobit**, we analyzed the cognitive ability (dtlm) of patients with schizophrenia compared with their relatives and control subjects, by using a three-level probit model with random effects at the family and subject levels. Cognitive ability was measured as the successful completion of the "Tower of London", a computerized task, measured at three levels of difficulty.

```
. use https://www.stata-press.com/data/r19/towerlondon
(Tower of London data)
. meprobit dtlm difficulty i.group || family: || subject:
  (output omitted)
```

We obtain predicted probabilities based on the contribution of both fixed effects and random effects by typing

```
. predict pr
(option mu assumed)
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

As the note says, the predicted values are based on the posterior means of random effects. You can use the modes option to obtain predictions based on the posterior modes of random effects.

We obtain predictions of the posterior means themselves by typing

```
. predict re*, reffects
(calculating posterior means of random effects)
(using 7 quadrature points)
```

Because we have one random effect at the family level and another random effect at the subject level, Stata saved the predicted posterior means in the variables re1 and re2, respectively. If you are not sure which prediction corresponds to which level, you can use the describe command to show the variable labels.

Here we list the data for family 16:

```
. list family subject dtlm pr re1 re2 if family==16, sepby(subject)
```

	family	subject	dtlm	pr	re1	re2
208.	16	5	1	.5301687	.5051272	.1001124
209.	16	5	0	.1956408	.5051272	.1001124
210.	16	5	0	.0367041	.5051272	.1001124
211.	16	34	1	.8876646	.5051272	.7798247
212.	16	34	1	.6107262	.5051272	.7798247
213.	16	34	1	.2572725	.5051272	.7798247
214.	16	35	0	.6561904	.5051272	-.0322885
215.	16	35	1	.2977437	.5051272	-.0322885
216.	16	35	0	.071612	.5051272	-.0322885

The predicted random effects at the family level (re1) are the same for all members of the family. Similarly, the predicted random effects at the individual level (re2) are constant within each individual. The predicted probabilities (pr) for this family seem to be in fair agreement with the response (dtlm) based on a cutoff of 0.5.

We can use estat icc to estimate the residual intraclass correlation (conditional on the difficulty level and the individual's category) between the latent responses of subjects within the same family or between the latent responses of the same subject and family:

```
. estat icc
Residual intraclass correlation
```

Level	ICC	Std. err.	[95% conf.	interval]
family	.1352637	.1050492	.0261998	.4762821
subject|family	.3622485	.0877459	.2124808	.5445812

estat icc reports two intraclass correlations for this three-level nested model. The first is the level-3 intraclass correlation at the family level, the correlation between latent measurements of the cognitive ability in the same family. The second is the level-2 intraclass correlation at the subject-within-family level, the correlation between the latent measurements of cognitive ability in the same subject and family.

There is not a strong correlation between individual realizations of the latent response, even within the same subject.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**.

Also see

[ME] **meprobit** — Multilevel mixed-effects probit regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

mestreg — Multilevel mixed-effects parametric survival models

Description | Quick start | Menu | Syntax | Options
Remarks and examples | Stored results | Methods and formulas | References | Also see

Description

`mestreg` fits a mixed-effects parametric survival-time model. The conditional distribution of the response given the random effects is assumed to be an exponential, Weibull, lognormal, loglogistic, or gamma distribution. `mestreg` can be used with single- or multiple-record st data.

Quick start

Without weights

Two-level Weibull survival model with covariates `x1` and `x2` and random intercepts by `lev2` using `stset` data

```
mestreg x1 x2 || lev2:, distribution(weibull)
```

Mixed-effects model adding random coefficients for `x1`

```
mestreg x1 x2 || lev2:x1, distribution(weibull)
```

Three-level random-intercept model with `lev2` nested within `lev3`

```
mestreg x1 x2 || lev3: || lev2:, distribution(weibull)
```

With weights

Two-level Weibull survival model with covariates `x1` and `x2`, random intercepts by `lev2`, and observation-level frequency weights `wvar1` using `stset` data

```
mestreg x1 x2 [fweight=wvar1] || lev2:, distribution(weibull)
```

Two-level random-intercept model from a two-stage sampling design with PSUs identified by `psu` using PSU-level and observation-level sampling weights `wvar2` and `wvar1`

```
mestreg x1 x2 [pweight=wvar1] || psu:, pweight(wvar2)
```

Same as above, but `svyset` the data first

```
svyset psu, weight(wvar2) || _n, weight(wvar1)
svy: mestreg x1 x2 || psu:, distribution(weibull)
```

Note: Any supported parametric survival distribution may be specified in place of `weibull` above.

Menu

Statistics > Multilevel mixed-effects models > Parametric survival regression

Syntax

```
mestreg fe_equation [ || re_equation ] [ || re_equation ... ],
    distribution(distname) [ options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress constant term from the fixed-effects equation
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
* distribution(*distname*)	specify survival distribution
time	use accelerated failure-time metric
constraints(*constraints*)	apply specified linear constraints
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, opg, robust, or cluster *clustvar*
Reporting	
level(#)	set confidence level; default is level(95)
nohr	do not report hazard ratios
tratio	report time ratios
noshow	do not show st setting information
nocnsreport	do not display constraints
notable	suppress coefficient table
noheader	suppress output header
nogroup	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
intmethod(*intmethod*)	integration method
intpoints(#)	set the number of integration (quadrature) points for all levels; default is intpoints(7)
Maximization	
maximize_options	control the maximization process; seldom used
startvalues(*svmethod*)	method for obtaining starting values
startgrid[(*gridspec*)]	perform a grid search to improve starting values
noestimate	do not fit the model; show starting values instead
dnumerical	use numerical derivative techniques
collinear	keep collinear variables
coeflegend	display legend instead of statistics

*distribution(*distname*) is required.

vartype	Description
independent	one unique variance parameter per random effect and all covariances 0; the default unless the R. notation is used
exchangeable	equal variances for random effects and one common pairwise covariance
identity	equal variances for random effects and all covariances 0; the default if the R. notation is used
unstructured	all variances and covariances to be distinctly estimated
fixed(*matname*)	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
pattern(*matname*)	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

distname	Description
exponential	exponential survival distribution
loglogistic	loglogistic survival distribution
llogistic	synonym for loglogistic
weibull	Weibull survival distribution
lognormal	lognormal survival distribution
lnormal	synonym for lognormal
gamma	gamma survival distribution

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

You must stset your data before using mestreg; see [ST] **stset**.

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

bayes, by, collect, and svy are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: mestreg**.

vce() and weights are not allowed with the svy prefix; see [SVY] **svy**.

fweights, iweights, and pweights are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

startvalues(), startgrid, noestimate, dnumerical, collinear, and coeflegend do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

noconstant suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

offset(*varname*) specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

covariance(*vartype*) specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: independent, exchangeable, identity, unstructured, fixed(*matname*), or pattern(*matname*).

covariance(independent) covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is covariance(independent) unless a crossed random-effects model is fit, in which case the default is covariance(identity).

covariance(exchangeable) structure specifies one common variance for all random effects and one common pairwise covariance.

covariance(identity) is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`distribution(`*distname*`)` specifies the survival model to be fit. *distname* is one of the following: `exponential`, `loglogistic`, `llogistic`, `weibull`, `lognormal`, `lnormal`, or `gamma`. This option is required.

`time` specifies that the model be fit in the accelerated failure-time metric rather than in the log relative-hazard metric. This option is valid only for the exponential and Weibull models because these are the only models that have both a proportional hazards and an accelerated failure-time parameterization. Regardless of metric, the likelihood function is the same, and models are equally appropriate in either metric; it is just a matter of changing interpretation.

`time` must be specified at estimation.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim, opg), that are robust to some kinds of misspecification (robust), and that allow for intragroup correlation (cluster *clustvar*); see [R] ***vce_option***. If vce(robust) is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

level(#); see [R] **Estimation options**.

nohr, which may be specified at estimation or upon redisplaying results, specifies that coefficients rather than exponentiated coefficients be displayed, that is, that coefficients rather than hazard ratios be displayed. This option affects only how coefficients are displayed, not how they are estimated.

This option is valid only for the exponential and Weibull models because they have a natural proportional hazards parameterization. These two models, by default, report hazards ratios (exponentiated coefficients).

tratio specifies that exponentiated coefficients, which are interpreted as time ratios, be displayed. tratio is appropriate only for the loglogistic, lognormal, and gamma models or for the exponential and Weibull models when fit in the accelerated failure-time metric.

tratio may be specified at estimation or upon replay.

noshow prevents mestreg from showing the key st variables. This option is rarely used because most users type stset, show or stset, noshow to set once and for all whether they want to see these variables mentioned at the top of the output of every st command; see [ST] **stset**.

nocnsreport; see [R] **Estimation options**.

notable suppresses the estimation table, either at estimation or upon replay.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

`intmethod(`*intmethod*`)` specifies the integration method to be used for the random-effects model. `mvaghermite` performs mean–variance adaptive Gauss–Hermite quadrature; `mcaghermite` performs mode-curvature adaptive Gauss–Hermite quadrature; `ghermite` performs nonadaptive Gauss–Hermite quadrature; and `laplace` performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is `mvaghermite` unless a crossed random-effects model is fit, in which case the default integration method is `laplace`. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

`intpoints(`#`)` sets the number of integration points for quadrature. The default is `intpoints(7)`, which means that seven quadrature points are used for each level of random effects. This option is not allowed with `intmethod(laplace)`.

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(`#`)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(`#`)`, `ltolerance(`#`)`, `nrtolerance(`#`)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **Maximize**. Those that require special mention for `mestreg` are listed below.

`from()` accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with `mestreg` but are not shown in the dialog box:

`startvalues(`*svmethod*`)`, `startgrid[(`*gridspec*`)]`, `noestimate`, and `dnumerical`; see [ME] **meglm**.

`collinear`, `coeflegend`; see [R] **Estimation options**.

Remarks and examples

For a general introduction to `me` commands, see [ME] **me**.

Remarks are presented under the following headings:

Introduction
Two-level models
Three-level models

Introduction

Mixed-effects survival models contain both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

mestreg allows for many levels of random effects. However, for simplicity, we now consider two-level models, where we have a series of M independent clusters and a set of random effects $\mathbf{u}_j$ corresponding to those clusters. Two often-used models for adjusting survivor functions for the effects of covariates are the accelerated failure-time (AFT) model and the multiplicative or proportional hazards (PH) model.

In the AFT model, the natural logarithm of the survival time, $\log t$, is expressed as a linear function of the covariates; when we incorporate random-effects, this yields the model

$$\log t_{ji} = \mathbf{x}_{ji}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j + v_{ji}$$

for $j = 1, \ldots, M$ clusters, with cluster j consisting of $i = 1, \ldots, n_j$ observations. The $1 \times p$ row vector $\mathbf{x}_{ji}$ contains the covariates for the fixed effects, with regression coefficients (fixed effects) $\boldsymbol{\beta}$.

The $1 \times q$ vector $\mathbf{z}_{ji}$ contains the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercept model, $\mathbf{z}_{ji}$ is simply the scalar 1. The random effects $\mathbf{u}_j$ are M realizations from a multivariate normal distribution with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components.

Finally, v_{ji} are the observation-level errors with density $\varphi(\cdot)$. The distributional form of the error term determines the regression model. Five regression models are implemented in mestreg using the AFT parameterization: exponential, gamma, loglogistic, lognormal, and Weibull. The lognormal regression model is obtained by letting $\varphi(\cdot)$ be the normal density. Similarly, by letting $\varphi(\cdot)$ be the logistic density, one obtains the loglogistic regression. Setting $\varphi(\cdot)$ equal to the extreme-value density yields the exponential and the Weibull regression models.

In the PH models fit by mestreg, the covariates have a multiplicative effect on the hazard function

$$h(t_{ji}) = h_0(t_{ji}) \exp(\mathbf{x}_{ji}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j)$$

for some baseline hazard function $h_0(t)$. For the mestreg command, $h_0(t)$ is assumed to be parametric. The exponential and Weibull models are implemented in mestreg for the PH parameterization. These two models are implemented using both the AFT and PH parameterizations.

mestreg is suitable only for data that have been stset. By using stset on your data, you define the variables _t0, _t, and _d, which serve as the trivariate response variable (t_0, t, d). Each response corresponds to a period under observation, $(t_0, t]$, resulting in either failure ($d = 1$) or right-censoring ($d = 0$) at time t.

mestreg does not allow delayed entry or gaps. However, mestreg is appropriate for data exhibiting multiple records per subject and time-varying covariates. mestreg requires subjects to be nested within clusters.

stset weights are not used; instead, weights must be specified at estimation. Weights are not allowed with crossed models or the Laplacian approximation. See *Survey estimation* in *Methods and formulas* for details.

Two-level models

▷ Example 1: Two-level random-intercept PH model

In example 11 of [ST] **streg**, we fit a Weibull model with an inverse-Gaussian shared frailty to the recurrence times for catheter-insertion point infection for 38 kidney dialysis patients. In this example, the subjects are the catheter insertions, not the patients themselves. This is a function of how the data were recorded—the onset of risk occurs at the time the catheter is inserted and not, say, at the time of admission of the patient into the study. Thus we have two subjects (insertions) within each group (patient). Each catheter insertion results in either infection (`infect==1`) or right-censoring (`infect==0`). The `stset` results are shown below.

```
. use https://www.stata-press.com/data/r19/catheter
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)
. stset
-> stset time, failure(infect)
Survival-time data settings
         Failure event: infect!=0 & infect<.
Observed time interval: (0, time]
     Exit on or before: failure
------------------------------------------------------------------------------
         76  total observations
          0  exclusions
------------------------------------------------------------------------------
         76  observations remaining, representing
         58  failures in single-record/single-failure data
      7,424  total analysis time at risk and under observation
                                            At risk from t =         0
                                 Earliest observed entry t =         0
                                      Last observed exit t =       562
```

While it is reasonable to assume independence of patients, we would not want to assume that recurrence times within each patient are independent. The model used in [ST] **streg** allowed us to model the correlation by assuming that it was the result of a latent patient-level effect, or frailty.

The random-effects approach used by `mestreg` is more flexible because it allows you to experiment with several levels of random effects, including random coefficients, or both. You can then choose the model that best suits your data. Here we use `mestreg` to fit a random-effects Weibull model with normally distributed random effects. This model can be viewed as a shared frailty model with lognormal frailty.

```
. mestreg age female || patient:, distribution(weibull)
        Failure _d: infect
  Analysis time _t: time
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -1700989.9
Iteration 1:  Log likelihood =  -440.1998
Iteration 2:  Log likelihood = -336.62162
Iteration 3:  Log likelihood = -334.64937
Iteration 4:  Log likelihood = -334.57959
Iteration 5:  Log likelihood = -334.57944
Iteration 6:  Log likelihood = -334.57944
Refining starting values:
Grid node 0:  Log likelihood = -336.03604
Fitting full model:
Iteration 0:  Log likelihood = -336.03604  (not concave)
Iteration 1:  Log likelihood = -333.14043
Iteration 2:  Log likelihood = -330.40952
Iteration 3:  Log likelihood = -329.89242
Iteration 4:  Log likelihood = -329.87847
Iteration 5:  Log likelihood = -329.87832
Iteration 6:  Log likelihood = -329.87832
Mixed-effects Weibull PH regression             Number of obs     =         76
Group variable: patient                         Number of groups  =         38
                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(2)      =      10.12
Log likelihood = -329.87832                     Prob > chi2       =     0.0063
------------------------------------------------------------------------------
          _t | Haz. ratio   Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         age |   1.007348   .013788     0.53   0.593     .9806828    1.034737
      female |   .1904727   .099992    -3.16   0.002     .0680737    .5329493
       _cons |   .0072901   .0072274   -4.96   0.000     .0010444    .0508881
-------------+----------------------------------------------------------------
       /ln_p |   .2243233   .1402795                     -.0506195    .4992661
-------------+----------------------------------------------------------------
patient      |
   var(_cons) |   .8308495   .4978425                      .256735    2.688808
------------------------------------------------------------------------------
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard (conditional on zero random effects).
LR test vs. Weibull model: chibar2(01) = 9.40         Prob >= chibar2 = 0.0011
```

The results are similar to those in [ST] **streg**. The likelihood-ratio test compares the random-effects model with a survival model with fixed-effects only. The results support the random-effects model.

By default, when fitting a model with the PH parameterization, `mestreg` displays exponentiated coefficients, labeled as hazard ratios. These hazard ratios should be interpreted as "conditional hazard ratios", that is, conditional on the random effects.

For example, the hazard ratio for `age` is 1.01. This means that according to the model, for a given patient, the hazard would increase 1% with each year of age. However, at the population level, marginal hazards corresponding to different levels of the covariates are not necessarily proportional. Example 5 in [ME] **mestreg postestimation** illustrates this point with simulated data.

The exponentiated coefficients of covariates that usually remain constant within a group do not have a natural interpretation as conditional hazard ratios. However, the magnitude of the exponentiated coefficients always gives an idea of the effect of the covariates. In this example, `female` is constant within the group. The estimated hazard ratio for `female` is 0.19, which indicates that hazard functions for females tend to be smaller than hazard functions for males. Both conditional and unconditional predictions can be obtained with `predict`. Unconditional predictions can be visualized by using `stcurve`. Unconditional effects can be tested and visualized by using `margins` and `marginsplot`. See example 1 in [ME] **mestreg postestimation** for an example using `predict`, `margins`, and `marginsplot`.

◁

▷ Example 2: Two-level random-intercept AFT model

Although the PH parameterization is more popular in the literature because the output is easier to interpret, the AFT parameterization is useful when we need to make comparisons with other models that have only an AFT parameterization. For example, we might want to compare the Weibull results from example 1 with the results from a gamma model.

Let's redisplay the results of a Weibull PH model from example 1 as coefficients:

```
. mestreg, nohr
Mixed-effects Weibull PH regression             Number of obs     =         76
Group variable: patient                         Number of groups  =         38
                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(2)      =      10.12
Log likelihood = -329.87832                     Prob > chi2       =     0.0063
------------------------------------------------------------------------------
          _t | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         age |   .0073207   .0136874     0.53   0.593    -.0195062    .0341476
      female |  -1.658247   .5249676    -3.16   0.002    -2.687164    -.629329
       _cons |  -4.921236   .9914009    -4.96   0.000    -6.864346   -2.978126
-------------+----------------------------------------------------------------
       /ln_p |   .2243233   .1402795                     -.0506195    .4992661
-------------+----------------------------------------------------------------
patient      |
   var(_cons) |   .8308495   .4978425                      .256735    2.688808
------------------------------------------------------------------------------
LR test vs. Weibull model: chibar2(01) = 9.40         Prob >= chibar2 = 0.0011
```

We can refit the Weibull model using the AFT parameterization by specifying option `time`.

```
. mestreg age female || patient:, distribution(weibull) time
        Failure _d: infect
  Analysis time _t: time
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -346.46486
Iteration 1:  Log likelihood = -343.29515
Iteration 2:  Log likelihood =  -335.0513
Iteration 3:  Log likelihood = -334.58308
Iteration 4:  Log likelihood = -334.57944
Iteration 5:  Log likelihood = -334.57944
Refining starting values:
Grid node 0:  Log likelihood = -335.10428
Fitting full model:
Iteration 0:  Log likelihood = -335.10428
Iteration 1:  Log likelihood = -332.13546
Iteration 2:  Log likelihood = -330.01623
Iteration 3:  Log likelihood = -329.88013
Iteration 4:  Log likelihood = -329.87832
Iteration 5:  Log likelihood = -329.87832
Mixed-effects Weibull AFT regression            Number of obs     =         76
Group variable: patient                         Number of groups  =         38
                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(2)      =      13.00
Log likelihood = -329.87832                     Prob > chi2       =     0.0015
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	-.0058496	.010872	-0.54	0.591	-.0271585	.0154592
female	1.325034	.3719102	3.56	0.000	.596103	2.053964
_cons	3.932346	.5663757	6.94	0.000	2.82227	5.042422
/ln_p	.2243237	.1402794			-.0506189	.4992663
patient						
var(_cons)	.5304902	.2343675			.2231626	1.261053

```
LR test vs. Weibull model: chibar2(01) = 9.40         Prob >= chibar2 = 0.0011
```

The estimates of coefficients and variance components are different between the two models. In fact, the coefficients have the opposite signs. This is expected because the two models have different parameterizations. The relationship between the coefficients and variances of the two parameterizations for the Weibull model is

$$\beta_{\text{PH}} = -p \times \beta_{\text{AFT}}$$

$$\text{var}_{\text{PH}} = p^2 \times \text{var}_{\text{AFT}}$$

where p denotes the ancillary parameter. It is estimated in the logarithmic metric and is displayed in the output as `/ln_p`.

For example, we could calculate β_{PH} for `female` as approximately $-\exp(0.22) \times 1.33 = -1.66$. If we exponentiate this to obtain the hazard ratio that was reported in example 1, we obtain the same reported result, 0.19.

For a discussion of the differences between the PH and AFT parameterizations, see, for example, Cleves, Gould, and Marchenko (2016).

Now, we can compare the results from our Weibull specification with the results from a gamma specification.

```
. mestreg age female || patient:, distribution(gamma)
        Failure _d: infect
  Analysis time _t: time
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -351.17349
Iteration 1:  Log likelihood = -337.04571
Iteration 2:  Log likelihood = -335.10167
Iteration 3:  Log likelihood = -335.09115
Iteration 4:  Log likelihood = -335.09115
Refining starting values:
Grid node 0:  Log likelihood = -334.49759
Fitting full model:
Iteration 0:  Log likelihood = -334.49759
Iteration 1:  Log likelihood = -331.87827
Iteration 2:  Log likelihood = -329.64795
Iteration 3:  Log likelihood = -329.52682
Iteration 4:  Log likelihood = -329.52635
Iteration 5:  Log likelihood = -329.52634
Mixed-effects gamma AFT regression              Number of obs     =         76
Group variable: patient                         Number of groups  =         38
                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(2)      =      13.23
Log likelihood = -329.52634                     Prob > chi2       =     0.0013
------------------------------------------------------------------------------
          _t | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         age |  -.0060276   .0108267    -0.56   0.578    -.0272475    .0151924
      female |   1.324745   .3685132     3.59   0.000     .6024726    2.047018
       _cons |   3.873854   .5628993     6.88   0.000     2.770592    4.977117
-------------+----------------------------------------------------------------
       /logs |  -.1835075   .1008892                     -.3812467    .0142317
-------------+----------------------------------------------------------------
patient      |
   var(_cons) |   .5071823   .2241959                      .213254    1.206232
------------------------------------------------------------------------------
LR test vs. gamma model: chibar2(01) = 11.13          Prob >= chibar2 = 0.0004
```

The coefficients and the random-effects variance are very similar for the two AFT models.

We can compare the marginal distributions or hazard functions for the two models by using `stcurve`; see example 2 in [ME] **mestreg postestimation**.

◁

▷ Example 3: Two-level random-slope model

In this example, we use a modified form of the dataset from Rabe-Hesketh and Skrondal (2022, sec. 15.7), previously published in Danahy et al. (1977) and analyzed by Pickles and Crouchley (1994, 1995) and Rabe-Hesketh, Skrondal, and Pickles (2004).

`angina.dta` includes data on 21 patients with coronary heart disease who participated in a randomized crossover trial comparing a drug to prevent angina (chest pain) with a placebo. The participants are identified by `pid`.

Before receiving the drug (or placebo), participants were asked to exercise on exercise bikes to the onset of angina or, if angina did not occur, to exhaustion. The exercise time, `seconds`, and the result of the exercise, `angina`—angina (`angina=1`) or exhaustion (`angina=0`)—were recorded. The drug (`treat=1`) or placebo (`treat=0`) was then taken orally, and the exercise test was repeated one, three, and five hours (variable `occasion`) after drug or placebo administration. Because each exercise test can have a failure (the occurrence of angina), the test is the subject. Each test is identified by `tid`. Failure is indicated by the variable `angina`. In this case, we have eight repeated measures per study participant.

Before fitting the model, we `stset` our data:

```
. use https://www.stata-press.com/data/r19/angina
(Angina drug data, Rabe-Hesketh and Skrondal (2021, ch. 15.7))
. stset seconds, failure(angina) id(tid)
Survival-time data settings
           ID variable: tid
         Failure event: angina!=0 & angina<.
Observed time interval: (seconds[_n-1], seconds]
     Exit on or before: failure
------------------------------------------------------------------------------
        168  total observations
          0  exclusions
------------------------------------------------------------------------------
        168  observations remaining, representing
        168  subjects
        155  failures in single-failure-per-subject data
     47,267  total analysis time at risk and under observation
                                                At risk from t =         0
                                     Earliest observed entry t =         0
                                          Last observed exit t =       743
```

To reiterate, we specify `seconds` as the time variable, `angina` as the failure variable, and `tid` as the variable identifying multiple observations per test.

Rabe-Hesketh and Skrondal (2022) apply several models to this dataset, including a lognormal model and a Cox model with random effects. We fit a Weibull model with covariates `occasion` and `treat` and interaction between `occasion` and `treat`. We include a random effect at the subject level.

```
. mestreg occasion##treat || pid:, distribution(weibull) nofvlabel
        Failure _d: angina
  Analysis time _t: seconds
       ID variable: tid
note: 1.occasion#1.treat identifies no observations in the sample.
note: 4.occasion#1.treat omitted because of collinearity.
  (output omitted)
Mixed-effects Weibull PH regression             Number of obs     =        168
Group variable: pid                             Number of groups  =         21
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.0
                                                              max =          8
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =      78.14
Log likelihood = -885.67135                     Prob > chi2       =     0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf.	interval]
occasion						
2	.719456	.2031744	-1.17	0.244	.4136423	1.251364
3	.902988	.2542476	-0.36	0.717	.5200146	1.568009
4	1.264262	.3516347	0.84	0.399	.7329746	2.180648
1.treat	.3825531	.128784	-2.85	0.004	.1977608	.7400195
occasion#treat						
1 1	1	(empty)				
2 1	.1576401	.0804767	-3.62	0.000	.0579589	.4287586
3 1	.4512793	.2127706	-1.69	0.091	.1791093	1.137032
4 1	1	(omitted)				
_cons	4.90e-13	9.98e-13	-13.91	0.000	9.03e-15	2.66e-11
/ln_p	1.640297	.0689544			1.505149	1.775445
pid						
var(_cons)	4.529641	1.544175			2.322124	8.835725

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard (conditional on zero random effects).
LR test vs. Weibull model: chibar2(01) = 177.40       Prob >= chibar2 = 0.0000
```

Because individuals were exercising without the administration of a placebo or treatment at the first occasion (`occasion==1`), the category for interaction between `occasion==1` and `treat==1` is empty.

The estimated variance at the individual level (that is, the variance between individuals) is equal to 4.53. The likelihood-ratio test shows evidence in favor of the random-effects model versus the fixed-effects model.

The parameter p is $\exp(1.640297) = 5.16$, which is larger than 1. This means that the estimated hazard (conditional on the covariates and on the random effects) is a monotonically increasing function if we assume a Weibull distribution.

The model contains interaction terms for occasion and treat. Interpretation of interaction terms is usually less straightforward. Briefly, to interpret the exponentiated coefficients as conditional hazard ratios, we need to examine all the covariates in the interaction. The hazard function for pid $= j$, when we set occasion $= k$ and treat $= l$, will be

$$h(t) = h_0(t) \times \exp(\beta_{\text{occ}_k} + \beta_{\text{treat}_l} + \beta_{\text{occ}_k \times \text{treat}_l} + \texttt{_cons} + u_j)$$

where β_{occ_k}, β_{treat_l}, and $\beta_{\text{occ}_k \times \text{treat}_l}$ are, respectively, the coefficients for the dummies for occasion $= k$ and treat $= l$ and the interaction (occasion $= k \times$ treatment $= l$).

For example, when treat $= 0$, the hazard function is

$$h(t|\texttt{treat} = 0, \texttt{occasion} = k, \texttt{pid} = j) = h_0(t) \times \exp(\beta_{\text{occ}_k} + \texttt{_cons} + u_j)$$

where β_{occ_1} is equal to 0 because occasion $= 1$ is the base category. This means that for a given pid,

$$\frac{h(t|\texttt{treat} = 0, \texttt{occ} = k, \texttt{pid} = j)}{h(t|\texttt{treat} = 0, \texttt{occ} = 1, \texttt{pid} = j)} = \exp(\beta_{\text{occ}_k})$$

Notice that this is only true within pid, because different participants have different u_js.

The coefficients have already been exponentiated, so we can see clearly that according to this model, when there is no treatment, the hazard for occasion 2 is smaller than the hazard for occasion 1. The increasing ratios indicate that the hazard increases with the occasion. Similar calculations could be performed for other interaction terms.

The easiest way to interpret models with interactions is by using margins and marginsplot, which allow us to compute and then visualize unconditional predictions and marginal effects. See [R] **margins** for more information.

Above we assumed a constant treatment effect for all individuals for each occasion. However, we may instead believe that the treatment effect varies also with individuals. This can be modeled by adding a random coefficient for the treatment, i.treat, at the individual level; we also include the covariance(unstructured) option to estimate a covariance term between the random intercept and the random slope for 1.treat.

```
. mestreg occasion##treat || pid: i.treat, distribution(weibull)
> covariance(unstructured) nofvlabel
        Failure _d: angina
  Analysis time _t: seconds
       ID variable: tid
note: 1.occasion#1.treat identifies no observations in the sample.
note: 4.occasion#1.treat omitted because of collinearity.
```

(*output omitted*)

```
Mixed-effects Weibull PH regression             Number of obs     =        168
Group variable: pid                             Number of groups  =         21
                                                Obs per group:
                                                              min =          8
                                                              avg =        8.0
                                                              max =          8
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(6)      =      50.18
Log likelihood = -859.50038                     Prob > chi2       =     0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf.	interval]
occasion						
2	.5993591	.1861745	-1.65	0.099	.3260503	1.101766
3	.8643306	.2560242	-0.49	0.623	.483665	1.544597
4	1.333201	.3843218	1.00	0.318	.7577392	2.345694
1.treat	.2147751	.1280091	-2.58	0.010	.0667814	.6907365
occasion#treat						
1 1	1	(empty)				
2 1	.1594337	.0885644	-3.31	0.001	.0536714	.4736058
3 1	.4632936	.2273925	-1.57	0.117	.1770402	1.212385
4 1	1	(omitted)				
_cons	6.21e-17	1.75e-16	-13.20	0.000	2.44e-19	1.58e-14
/ln_p	1.91931	.0736166			1.775024	2.063596
pid						
var(1.treat)	4.682507	1.956897			2.064178	10.62208
var(_cons)	6.939041	2.372975			3.549852	13.56403
pid						
cov(1.treat, _cons)	1.73782	1.313054	1.32	0.186	-.8357182	4.311357

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard (conditional on zero random effects).
LR test vs. Weibull model: chi2(3) = 229.74                 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

We obtain somewhat different estimates of hazard ratios, but our inferential conclusions remain the same. We now observe two variances in the output, the variance for the intercept at the individual level and the variance for the coefficient for treatment at the individual level. The variance for the intercept is smaller because some of the variability is now explained by varying coefficients for treatment. The covariance is positive, meaning that the random slope tends to be larger for individuals who have a larger random intercept. See example 4 in [ME] **mestreg postestimation** for an application of `predict` that presents a graphical analysis of this relationship.

◁

Three-level models

▷ Example 4: Three-level random-slope model

Blossfeld, Golsch, and Rohwer (2007) analyze a dataset based on the German Life History Study of Mayer and Brückner (1989), collected in the years 1981–1983. (This dataset is also available in Blossfeld, Rohwer, and Schneider (2019), a second edition of the 2007 reference.) The `jobhistory` dataset contains a modified version of Blossfeld, Golsch, and Rohwer's anonymization of a random sample of 201 respondents from the original data. Each of the 600 observations in the dataset corresponds to a job episode. Variable `id` contains identification of the individual, `tstart` contains the starting point of the job (in months from the beginning of the century), `tend` is the end of the job episode, and `failure` indicates whether the date in `tend` corresponds to the actual end of the employment in a certain job or whether it is a censored observation.

We first `stset` the data. As explained in Cleves (1999) and Therneau and Grambsch (2000), when analyzing multiple-failure data, we can consider two main approaches. One approach is to define the study time from the first time that an individual starts being at risk. The second approach is to define the study time from the last failure. We will take the second approach, which means that we treat each job episode as the subject.

Therefore, the origin is defined as the start of each job episode, and the study time will be the time from the start of each episode until the jobs end or the episode is censored.

```
. use https://www.stata-press.com/data/r19/jobhistory
(Job history data, Event History Analysis with Stata, Blossfeld et al. 2007)
. stset tend, origin(tstart) failure(failure)
Survival-time data settings
         Failure event: failure!=0 & failure<.
Observed time interval: (origin, tend]
     Exit on or before: failure
     Time for analysis: (time-origin)
               Origin: time tstart
------------------------------------------------------------------------------
        600  total observations
          0  exclusions
------------------------------------------------------------------------------
        600  observations remaining, representing
        458  failures in single-record/single-failure data
     40,782  total analysis time at risk and under observation
                                                 At risk from t =         0
                                      Earliest observed entry t =         0
                                           Last observed exit t =       428
```

We want to fit a Weibull model using the education level, the number of previous jobs, the prestige of the current job, and gender as explanatory variables. `education` records the highest education level before entering the labor market, `njobs` contains the number of previous jobs for each individual, and `prestige` is an index for the prestige of the current job. The `birthyear` variable indicates the year of birth. `female` is 1 for women, 0 for men. To account for individual heterogeneity, we include a random effect at the individual level.

```
. mestreg education njobs prestige i.female || id:, distribution(weibull)
        Failure _d: failure
  Analysis time _t: (tend-origin)
            Origin: time tstart
Fitting fixed-effects model:
Iteration 0:  Log likelihood = -5736904.5
Iteration 1:  Log likelihood = -2664.7487
Iteration 2:  Log likelihood = -2484.7829
Iteration 3:  Log likelihood = -2477.4358
Iteration 4:  Log likelihood = -2477.3338
Iteration 5:  Log likelihood = -2477.3337
Refining starting values:
Grid node 0:  Log likelihood = -2491.2191
Fitting full model:
Iteration 0:  Log likelihood = -2491.2191  (not concave)
Iteration 1:  Log likelihood = -2468.3995
Iteration 2:  Log likelihood = -2450.0938
Iteration 3:  Log likelihood = -2443.0739
Iteration 4:  Log likelihood =  -2442.875
Iteration 5:  Log likelihood = -2442.8747
Iteration 6:  Log likelihood = -2442.8746

Mixed-effects Weibull PH regression             Number of obs     =        600
Group variable: id                              Number of groups  =        201
                                                Obs per group:
                                                              min =          1
                                                              avg =        3.0
                                                              max =          9
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(4)      =      87.38
Log likelihood = -2442.8746                     Prob > chi2       =     0.0000
------------------------------------------------------------------------------
          _t | Haz. ratio   Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
   education |    1.11897   .0463468     2.71   0.007     1.031722    1.213597
       njobs |   .7071195   .0357624    -6.85   0.000     .6403884    .7808043
    prestige |   .9677567   .0069576    -4.56   0.000     .9542157      .98149
    1.female |    1.75651   .3185526     3.11   0.002     1.231063    2.506228
       _cons |   .0053352   .0029015    -9.62   0.000     .0018376    .0154904
-------------+----------------------------------------------------------------
       /ln_p |   .1695545   .0453649                      .0806409    .2584681
-------------+----------------------------------------------------------------
id           |
   var(_cons) |   1.016459   .2149037                      .671623    1.538347
------------------------------------------------------------------------------
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard (conditional on zero random effects).
LR test vs. Weibull model: chibar2(01) = 68.92        Prob >= chibar2 = 0.0000
```

The estimated variance of the random intercept is equal to 1.02.

According to this model, an increase in the number of previous jobs is negatively associated with job mobility; the same is true for an increase in the prestige of the current job. By contrast, an increase in the years of education is positively associated with job mobility. Also, women seem to be more mobile than men.

We now store our estimates for later use:

```
. estimates store randint
```

The dataset has only two natural levels. However, for illustration purposes, let's consider the following situation. Assume that we want to account for unobserved variables associated with the date of birth, such as life experience, level of familiarity with new technologies, and family situation. We therefore add a random effect for the year of birth. Now, individuals will be nested within birth years.

```
. mestreg education njobs prestige i.female || birthyear: || id:,
> distribution(weibull)
        Failure _d: failure
  Analysis time _t: (tend-origin)
            Origin: time tstart
```

(*output omitted*)

```
Mixed-effects Weibull PH regression             Number of obs     =        600
        Grouping information
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
birthyear	12	3	50.0	99
id	201	1	3.0	9

```
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(4)      =      83.20
Log likelihood = -2439.9066                     Prob > chi2       =     0.0000
```

_t	Haz. ratio	Std. err.	z	P>|z|	[95% conf.	interval]
education	1.120373	.045203	2.82	0.005	1.035189	1.212566
njobs	.7181197	.0372039	-6.39	0.000	.6487813	.7948686
prestige	.966567	.0069189	-4.75	0.000	.9531009	.9802234
1.female	1.734236	.3022479	3.16	0.002	1.232419	2.440384
_cons	.0059091	.0031758	-9.55	0.000	.0020609	.0169429
/ln_p	.1685641	.0454824			.0794203	.257708
birthyear var(_cons)	.0950371	.0741445			.0205976	.4385006
birthyear>id var(_cons)	.8728384	.2020938			.5544339	1.374099

```
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard (conditional on zero random effects).
LR test vs. Weibull model: chi2(2) = 74.85                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

The results for the fixed part of the model are similar to the ones in the previous model.

Now, we have two estimated variances—one estimate for the random intercept at the individual level and one estimate for the random intercept at the birth-year level.

The variance component for the individual level is smaller for this model, and it looks as if the first model might have been trying to explain a variance component at the birth-year level by incorporating it into the individual-level variance. We can perform a likelihood-ratio test to compare the stored model `randint` with the current model:

```
. lrtest randint .

Likelihood-ratio test
Assumption: randint nested within .

 LR chi2(1) =   5.94
Prob > chi2 = 0.0148

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

The test is conservative because we are testing on the boundary of the parameter space; see *Distribution theory for likelihood-ratio test* in [ME] **me** for details. Provided that we are testing only one variance component, we can adjust the p-value accordingly by dividing the reported value by two, which results in an adjusted p-value equal to 0.0074.

The test is significant at the 0.05 level. It supports the three-level model with the additional variance component at the birth-year level.

◁

Stored results

`mestreg` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(k)`	number of parameters
`e(k_eq)`	number of equations in `e(b)`
`e(k_eq_model)`	number of equations in overall model test
`e(k_dv)`	number of dependent variables
`e(k_f)`	number of fixed-effects parameters
`e(k_r)`	number of random-effects parameters
`e(k_rs)`	number of variances
`e(k_rc)`	number of covariances
`e(df_m)`	model degrees of freedom
`e(ll)`	log likelihood
`e(chi2)`	χ^2
`e(p)`	p-value for model test
`e(ll_c)`	log likelihood, comparison model
`e(chi2_c)`	χ^2, comparison test
`e(df_c)`	degrees of freedom, comparison test
`e(p_c)`	p-value for comparison test
`e(N_clust)`	number of clusters
`e(rank)`	rank of `e(V)`
`e(ic)`	number of iterations
`e(rc)`	return code
`e(converged)`	1 if converged, 0 otherwise

Macros

`e(cmd)`	`gsem`
`e(cmd2)`	`mestreg`
`e(cmdline)`	command as typed
`e(depvar)`	name of dependent variable
`e(wtype)`	weight type

e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	model name
e(title)	title in estimation output
e(distribution)	distribution
e(clustvar)	name of cluster variable
e(offset)	offset
e(intmethod)	integration method
e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(frm2)	hazard or time
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

Survival models
Survey data

Survival models

Survival models have a trivariate response (t_0, t, d):

t_0 is the starting time under observation $t_0 \geq 0$;

t is the ending time under observation $t \geq t_0$; and

d is an indicator for failure $d \in \{0, 1\}$.

The survival function for a given family is the complement of the cumulative distribution function, $S(t) = 1 - F(t)$. The unconditional density for a failure at time t is given by

$$g(t) = \frac{\partial F(t)}{\partial t} = -\frac{\partial S(t)}{\partial t}$$

Some distributions contain ancillary parameters that are not denoted here.

The conditional density for a failure at time t is

$$g(t|t \geq t_0, d = 1) = g(t)/S(t_0)$$

and the conditional probability of survival without failure up to time t is

$$P(T \geq t|t \geq t_0, d = 0) = S(t)/S(t_0)$$

The conditional likelihood is given by

$$L(t, t_0, d) = \left\{\frac{g(t)}{S(t_0)}\right\}^d \left\{\frac{S(t)}{S(t_0)}\right\}^{1-d}$$

See *Survival distributions* in [SEM] **Methods and formulas for gsem** for the specific density function corresponding to each distribution.

Given a set of cluster-level random effects $\mathbf{u}_j$ for $j = 1, \ldots, M$, the conditional distribution of $\mathbf{t}_j = (t_{j1}, \ldots, t_{jn_j})'$ on $\boldsymbol{\eta}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j = (\mathbf{x}_{j1}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j, \ldots, \mathbf{x}_{jn_j}\boldsymbol{\beta} + \mathbf{z}_{jn_j}\mathbf{u}_j)$ for cluster j is

$$f(\mathbf{t}_j|\boldsymbol{\eta}_j) = \prod_{i=1}^{n_j} f(t_{ji}|\eta_{ji})$$

where $f(t_{ji}|\eta_{ji})$ is the contribution to the likelihood from observation ji; that is,

$$f(t_{ji}|\eta_{ji}) = \left\{\frac{g(t_{ji}|\mathbf{x}_{ji}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j)}{S(t_{0ji}|\mathbf{x}_{ji}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j)}\right\}^{d_{ji}} \left\{\frac{S(t_{ji}|\mathbf{x}_{ji}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j)}{S(t_{0ji}|\mathbf{x}_{ji}\boldsymbol{\beta} + \mathbf{z}_{ji}\mathbf{u}_j)}\right\}^{1-d_{ji}} \tag{1}$$

where $g(t|\eta)$ and $S(t|\eta)$ are, respectively, the density and the survivor function conditional on the linear prediction η.

As mentioned in *Introduction* under *Remarks and examples*, `mestreg` does not allow delayed entry or gaps. Therefore, the first observation for a given subject will have a value of $t_0 = 0$, and subsequent spells for the subject must start at the end of the previous spell. That is, if observations ji and $j, i+1$ belong to the same subject, then $t_{0j,i+1} = t_{ji}$.

Because the prior distribution of $\mathbf{u}_j$ is multivariate normal with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$, the likelihood contribution for the jth cluster is obtained by integrating $\mathbf{u}_j$ out of the joint density $f(\mathbf{t}_j, \mathbf{u}_j)$,

$$\mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = (2\pi)^{-q/2} \, |\boldsymbol{\Sigma}|^{-1/2} \int f(\mathbf{t}_j|\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j) \exp\left(-\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2\right) d\mathbf{u}_j \tag{2}$$

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

Survey data

In the presence of sampling weights, following Rabe-Hesketh and Skrondal (2006), the weighted log pseudolikelihood for a two-level model is given as

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \sum_{j=1}^{M} w_j \log \int_{-\infty}^{\infty} \exp\left\{ \sum_{i=1}^{n_j} w_{i|j} \log f(t_{ji}|\eta_{ji}) \right\} \phi(\mathbf{v}_{j1}) \, d\mathbf{v}_{j1}$$

where w_j is the inverse of the probability of selection for the jth cluster; $w_{i|j}$ is the inverse of the conditional probability of selection of individual i, given the selection of cluster j; $f(t_{ji}|\eta_{ji})$ is as in (1); and η_{ji}, $\phi(\cdot)$, $\mathbf{v}_{j1}$ are defined as in *Methods and formulas* in [ME] **meglm**.

Weighted estimation is achieved through the direct application of w_j and $w_{i|j}$ into the likelihood calculations as detailed above to reflect replicated clusters for w_j and replicated observations within clusters for $w_{i|j}$. Because this estimation is based on replicated clusters and observations, frequency weights are handled similarly.

References

Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Blossfeld, H.-P., K. Golsch, and G. Rohwer. 2007. *Event History Analysis with Stata*. Mahwah, NJ: Erlbaum.

Blossfeld, H.-P., G. Rohwer, and T. Schneider. 2019. *Event History Analysis with Stata*. 2nd ed. New York: Routledge.

Cleves, M. A. 1999. FAQ: How do I analyze multiple failure-time data using Stata? https://www.stata.com/support/faqs/statistics/multiple-failure-time-data.

Cleves, M. A., W. W. Gould, and Y. V. Marchenko. 2016. *An Introduction to Survival Analysis Using Stata*. Rev. 3rd ed. College Station, TX: Stata Press.

Crowther, M. J. 2019. Multilevel mixed-effects parametric survival analysis: Estimation, simulation, and application. *Stata Journal* 19: 931–949.

Danahy, D. T., D. T. Burwell, W. S. Aronow, and R. Prakash. 1977. Sustained hemodynamic and antianginal effect of high dose oral isosorbide dinitrate. *Circulation* 55: 381–387. https://doi.org/10.1161/01.cir.55.2.381.

Joe, H. 2008. Accuracy of Laplace approximation for discrete response mixed models. *Computational Statistics and Data Analysis* 52: 5066–5074. https://doi.org/10.1016/j.csda.2008.05.002.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

Leyland, A. H., and H. Goldstein, eds. 2001. *Multilevel Modelling of Health Statistics*. New York: Wiley.

Lin, X., and N. E. Breslow. 1996. Bias correction in generalized linear mixed models with multiple components of dispersion. *Journal of the American Statistical Association* 91: 1007–1016. https://doi.org/10.2307/2291720.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

Mayer, K. U., and E. Brückner. 1989. Lebensverläufe und Wohlfahrtsentwicklung: Konzeption, Design und Methodik der Erhebung von Lebensverläufen der Geburtsjahrgänge 1929–1931, 1939–1941, 1949–1951. Materialien aus der Bildungsforschung 35, Max-Planck-Institut für Bildungsforschung, Berlin.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models*. 2nd ed. Hoboken, NJ: Wiley.

McLachlan, G. J., and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. New York: Dekker.

Pickles, A., and R. Crouchley. 1994. Generalizations and applications of frailty models for survival and event data. *Statistical Methods in Medical Research* 3: 263–278. https://doi.org/10.1177/096228029400300305.

———. 1995. A comparison of frailty models for multivariate survival data. *Statistics in Medicine* 14: 1447–1461. https://doi.org/10.1002/sim.4780141305.

Rabe-Hesketh, S., and A. Skrondal. 2006. Multilevel modelling of complex survey data. *Journal of the Royal Statistical Society*, A ser., 169: 805–827. https://doi.org/10.1111/j.1467-985X.2006.00426.x.

———. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Rabe-Hesketh, S., A. Skrondal, and A. Pickles. 2004. Generalized multilevel structural equation modeling. *Psychometrika* 69: 167–190. https://doi.org/10.1007/BF02295939.

———. 2005. Maximum likelihood estimation of limited and discrete dependent variable models with nested random effects. *Journal of Econometrics* 128: 301–323. https://doi.org/10.1016/j.jeconom.2004.08.017.

Raudenbush, S. W., and A. S. Bryk. 2002. *Hierarchical Linear Models: Applications and Data Analysis Methods*. 2nd ed. Thousand Oaks, CA: Sage.

Self, S. G., and K.-Y. Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association* 82: 605–610. https://doi.org/10.2307/2289471.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models*. Boca Raton, FL: Chapman and Hall/CRC.

Therneau, T. M., and P. M. Grambsch. 2000. *Modeling Survival Data: Extending the Cox Model*. New York: Springer.

Also see

[ME] **mestreg postestimation** — Postestimation tools for mestreg

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: mestreg** — Bayesian multilevel parametric survival models

[ST] **streg** — Parametric survival models

[ST] **Glossary**

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xtstreg** — Random-effects parametric survival models

[U] 20 Estimation and postestimation commands

mestreg postestimation — Postestimation tools for mestreg

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation commands are of special interest after `mestreg`:

Command	Description
`stcurve`	plot the survivor, hazard, and cumulative hazard functions
`estat group`	summarize the composition of the nested groups
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, medians, hazards, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

predict creates a new variable containing predictions such as mean and median survival times, hazards, survivor functions, linear predictions, and standard errors.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

predict [*type*] { *stub** | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

predict [*type*] { *stub** | *newvars* } [*if*] [*in*], reffects [*re_options*]

Syntax for obtaining ML scores

predict [*type*] { *stub** | *newvars* } [*if*] [*in*], scores

statistic	Description
Main	
mean	mean survival time; the default
median	median survival time
hazard	hazard
eta	fitted linear predictor
xb	linear predictor for the fixed portion of the model only
stdp	standard error of the fixed-portion linear prediction
surv	predicted survivor function
density	predicted density function
distribution	predicted distribution function

These statistics are available both in and out of sample; type predict ... if e(sample) ... if wanted only for the estimation sample.

options	Description
Main	
conditional(*ctype*)	compute *statistic* conditional on estimated random effects; default is conditional(ebmeans)
marginal	compute *statistic* marginally with respect to the random effects
nooffset	make calculation ignoring offset or exposure
Integration	
int_options	integration options

median may not be combined with marginal.

ctype	Description
ebmeans	empirical Bayes means of random effects; the default
ebmodes	empirical Bayes modes of random effects
fixedonly	prediction for the fixed portion of the model only

re_options	Description
Main	
ebmeans	use empirical Bayes means of random effects; the default
ebmodes	use empirical Bayes modes of random effects
reses(*stub** | *newvars*)	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
intpoints(#)	use # quadrature points to compute marginal predictions and empirical Bayes means
iterate(#)	set maximum number of iterations in computing statistics involving empirical Bayes estimators
tolerance(#)	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

mean, the default, calculates the mean survival time.

median calculates the median survival time.

hazard calculates the hazard. When marginal is specified, marginal hazard is calculated as a ratio of the marginal density to the marginal survivor function.

surv calculates the predicted survivor function.

eta, xb, stdp, density, distribution, scores, conditional(), marginal, and nooffset; see [ME] **meglm postestimation**. marginal may not be specified with median.

reffects, ebmeans, ebmodes, and reses(); see [ME] **meglm postestimation**.

Integration

intpoints(), iterate(), and tolerance(); see [ME] **meglm postestimation**.

margins

Description for margins

margins estimates margins of response for mean and median survival times and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

margins [*marginlist*] [, *options*]

margins [*marginlist*] , predict(*statistic* ...) [predict(*statistic* ...) ...] [*options*]

statistic	Description
mean	mean survival time; the default
median	median survival time
xb	linear predictor for the fixed portion of the model only
hazard	not allowed with margins
eta	not allowed with margins
stdp	not allowed with margins
surv	not allowed with margins
density	not allowed with margins
distribution	not allowed with margins
reffects	not allowed with margins
scores	not allowed with margins

Options conditional(ebmeans) and conditional(ebmodes) are not allowed with margins.

Option marginal is assumed where applicable if conditional(fixedonly) is not specified.

Statistics not allowed with margins are functions of stochastic quantities other than e(b).

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects parametric survival model with mestreg. For the most part, predictions center on obtaining estimates of the survival times or hazard functions. Conditional predictions are based on the computation of the group-specific random effects, and marginal predictions are obtained by numerically integrating out the random effects.

▷ Example 1 : Predicting conditional and marginal mean survival time

In example 1 of [ME] **mestreg**, we analyzed the time to infection of the catheter insertion point for 38 kidney dialysis patients. We fit the following model:

```
. use https://www.stata-press.com/data/r19/catheter
(Kidney data, McGilchrist and Aisbett, Biometrics, 1991)
. stset time, failure(infect)
  (output omitted)
. mestreg age female || patient:, distribution(weibull)
  (output omitted)
```

The `predict` command allows us to compute marginal and conditional predictions. Unless stated differently, we use the word "conditional" to mean "conditional on the empirical Bayes predictions of the random effects". Below we compute marginal and conditional means for the mean survival time.

```
. predict m_marg, mean marginal
. predict m_cond, mean conditional
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
```

Now, we can display the predictions for some of the patients.

```
. sort female age patient
. list patient female age m_* in 15/20, sepby(patient)
```

	patient	female	age	m_marg	m_cond
15.	29	0	53	52.79355	22.36027
16.	29	0	53	52.79355	22.36027
17.	16	0	60	50.67546	28.01295
18.	16	0	60	50.67546	28.01295
19.	38	0	60	50.67546	49.47013
20.	38	0	60	50.67546	49.47013

We see in the output that the predicted expected conditional mean for patient 29 is equal to 22.36 (shown in `m_cond`). This is the expected time to infection for this patient. However, the predicted marginal mean for this patient is 52.79 (shown in `m_marg`). This is the expected time to infection for a patient from the population who is male and is 53 years old. This particular patient seems to be more prone to infection than would be expected based on his age and gender.

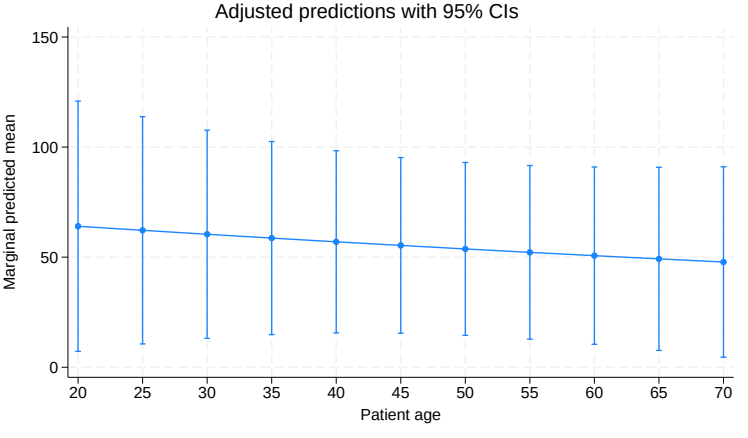
Conditional predictions are specific to each group, while marginal predictions are the same within each covariate pattern through the data. Patients 16 and 38 have the same covariate patterns; therefore, their marginal predicted means are the same. However, conditional predicted means differ.

margins and marginsplot show the changes in the marginal means for different ages.

```
. margins, predict(mean marginal) at(female=0 age=(20(5)70)) noatlegend
Adjusted predictions                                      Number of obs = 76
Model VCE: OIM
Expression: Marginal predicted mean, predict(mean marginal)
```

	Margin	Delta-method std. err.	z	P>|z|	[95% conf.	interval]
_at						
1	64.03481	28.99882	2.21	0.027	7.19816	120.8715
2	62.18903	26.33284	2.36	0.018	10.57761	113.8005
3	60.39646	24.11456	2.50	0.012	13.13279	107.6601
4	58.65556	22.37001	2.62	0.009	14.81116	102.5
5	56.96484	21.11488	2.70	0.007	15.58043	98.34925
6	55.32285	20.34538	2.72	0.007	15.44663	95.19908
7	53.7282	20.03192	2.68	0.007	14.46635	92.99004
8	52.17951	20.12	2.59	0.010	12.74503	91.61398
9	50.67546	20.53852	2.47	0.014	10.42071	90.93021
10	49.21476	21.21134	2.32	0.020	7.64129	90.78823
11	47.79617	22.06715	2.17	0.030	4.545348	91.04698

```
. marginsplot
Variables that uniquely identify margins: age
```



We see that the predicted marginal mean decreases with age; older patients are expected to have an event earlier. This is consistent with the findings from example 1 of [ME] **mestreg** that the hazard is increasing with age.

◁

▷ Example 2 : Predicting survivor functions

Continuing with example 1, we now predict survivor functions.

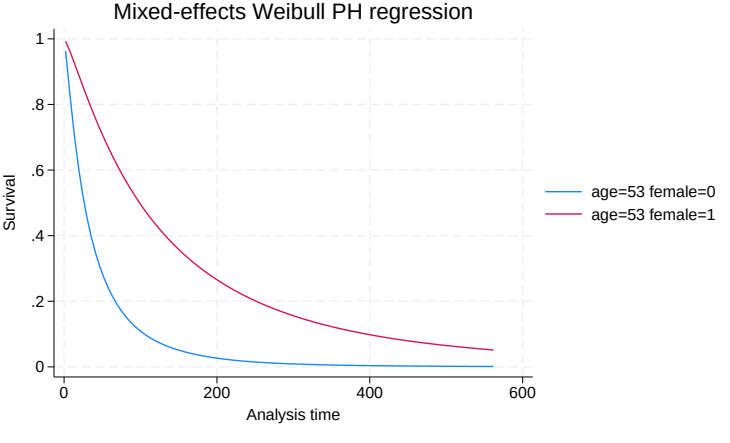
```
. predict S_marg, surv marginal
(using 7 quadrature points)
. predict S_cond, surv conditional
(predictions based on fixed effects and posterior means of random effects)
(using 7 quadrature points)
. sort female age patient _t
. list patient female age _t S_* in 15/20, sepby(patient)
```

	patient	female	age	_t	S_marg	S_cond
15.	29	0	53	2	.9628581	.9564017
16.	29	0	53	25	.5165027	.3493623
17.	16	0	60	4	.9122225	.9230723
18.	16	0	60	17	.6273606	.6129264
19.	38	0	60	8	.8141544	.9107039
20.	38	0	60	63	.20487	.2900458

Survival predictions vary with the value of the study time variable because they are predictions of the survivor function at the study time `_t`. For example, patient 29 has a 0.96 probability that a new insertion remains at least 2 days without infection and a 0.35 probability that a new insertion remains at least 25 days without infection. For a randomly chosen 53-year-old male patient from the population, the probabilities to remain at least 2 or 25 days without infection are, respectively, 0.96 and 0.52.

We can use `stcurve` to plot these predictions simultaneously for males and females of the same age.

```
. stcurve, surv at1(female=0 age=53) at2(female=1 age=53)
(option unconditional assumed)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
```



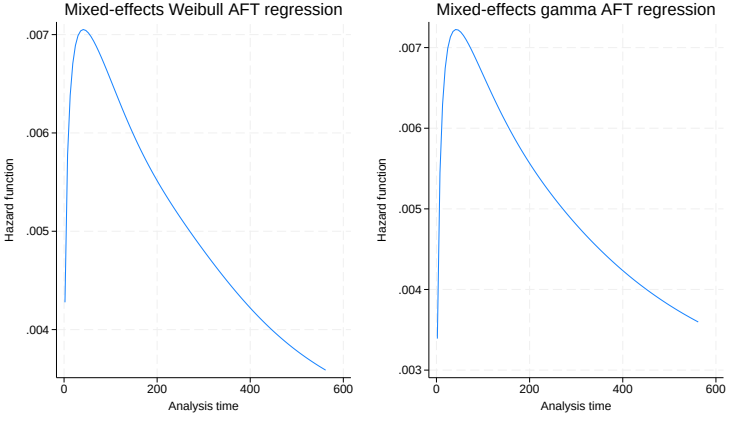
We see that the survivor function for females is above the survivor function for males, which means that females have a greater probability of not having an episode by study time _t.

◁

▷ Example 3 : Comparing marginal hazards

In example 2 of [ME] **mestreg**, we estimated two different distributions with random effects on patient and covariates age and female. Here we compare the marginal hazards using `stcurve`. By default, `stcurve` plots predictions at the mean of the covariates, computed over the whole estimation sample. We plot the predictions for `female==1`.

```
. mestreg age female || patient:, dist(weibull) time
  (output omitted)
. stcurve, hazard at(female=1)
(option unconditional assumed)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
. graph save g1
file g1.gph saved
. mestreg age female || patient:, dist(gamma)
  (output omitted)
. stcurve, hazard at(female=1)
(option unconditional assumed)
note: function evaluated at specified values of selected covariates and
      overall means of other covariates (if any).
. graph save g2
file g2.gph saved
. graph combine g1.gph g2.gph
```



The two estimated marginal hazards are similar. The marginal hazard has a very different shape from the conditional hazards. The conditional hazard function for a Weibull or a gamma distribution are both monotonic (increasing, constant, or decreasing, depending on the parameters).

◁

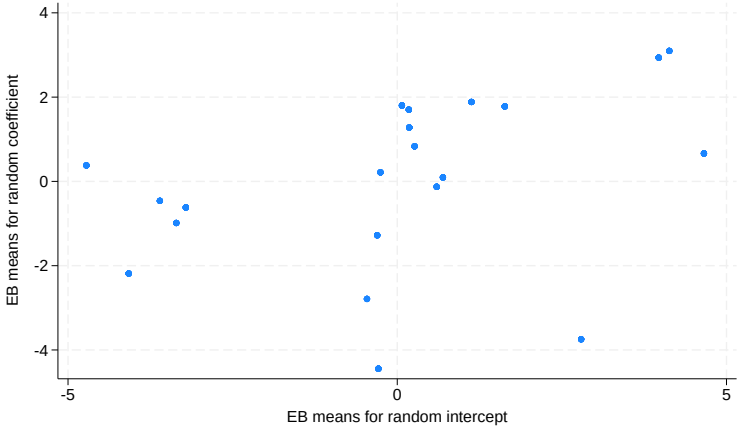
▷ Example 4 : Obtaining predictions of random effects

In example 3 of [ME] **mestreg**, we fit a Weibull model with random intercepts and random coefficients at the subject level. We obtained a positive covariance between the random effects. We refit the model here and then use `predict` with the option `reffects` to obtain predictions of the random effects based on the empirical Bayes posterior means.

```
. use https://www.stata-press.com/data/r19/angina, clear
(Angina drug data, Rabe-Hesketh and Skrondal (2021, ch. 15.7))
. mestreg occasion##treat || pid: i.treat, distribution(weibull)
> covariance(unstructured) nofvlabel
  (output omitted)
. predict re*, reffects
(calculating posterior means of random effects)
(using 7 quadrature points)
```

Plotting the predictions of the predicted random coefficient versus the random intercept shows the pattern we discussed in the main section: individuals with a larger random slope tend also to have a larger random intercept.

```
. twoway scatter re1 re2, ytitle(EB means for random coefficient)
> xtitle(EB means for random intercept)
```



Individuals with large random intercepts have individual hazards that are larger than those of other individuals with the same covariate patterns. Also, individuals with large random coefficients have individual conditional hazard ratios for treatment that are larger than those of other individuals with the same covariate pattern.

In other words, if the aim of the treatment is to decrease the hazard, then the positive correlation means that the treatment tends to be less effective for individuals who have a higher individual hazard (within the same occasion number).

◁

▷ Example 5 : Conditional and marginal hazards

In example 1 of [ME] **mestreg**, we mentioned that hazard ratios should be interpreted as conditional on the random effects. Here we use `predict` to illustrate this concept. We use a simulated dataset for a Weibull model with random effects for `group` and a binary covariate `x`.

We show that for a given group, the conditional hazard function satisfies the proportional hazards (PH) assumption. That is, for a given group j,

$$h(t|\mathtt{x}=1,\mathtt{group}=j) = \exp(\beta_x) \times h(t|\mathtt{x}=0,\mathtt{group}=j)$$

is equivalent to

$$\log\{h(t|\mathtt{x}=1,\mathtt{group}=j)\} = \beta_x + \log\{h(t|\mathtt{x}=0,\mathtt{group}=j)\}$$

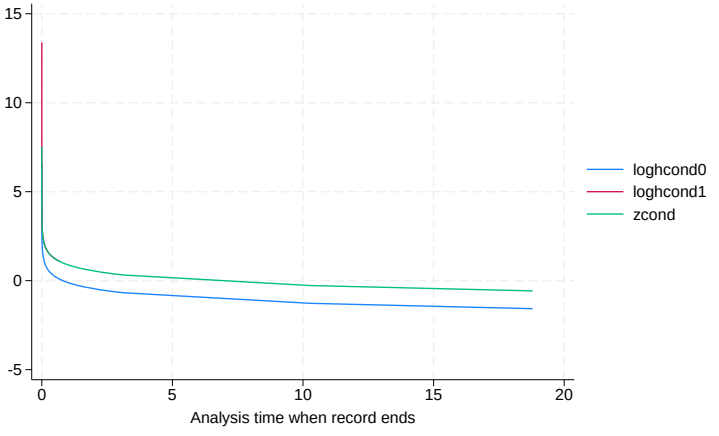
This property of the log hazard-function translates to one curve being a shifted version of the other, which is easier to see than the proportionality of the (untransformed) hazard function.

After fitting the model, we use `predict` to compute the conditional prediction of the hazard function for group 1; we create the variables `hcond0` and `hcond1`. `hcond0` will contain the conditional hazard for group 1 when `x==0`; `hcond1` will contain the conditional hazard for group 1 when `x==1`.

We also create `zcond` = `loghcond0` + β_x. If the PH assumption is satisfied, then the plotted values of `zcond` will be superimposed on those of `loghcond1`.

```
. use https://www.stata-press.com/data/r19/weibre, clear
. mestreg i.x || group:, distribution(weibull) nolog
        Failure _d: 1 (meaning all fail)
  Analysis time _t: t
Mixed-effects Weibull PH regression             Number of obs     =    100,000
Group variable: group                           Number of groups  =        500
                                                Obs per group:
                                                              min =        200
                                                              avg =      200.0
                                                              max =        200
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(1)      =   21447.86
Log likelihood = 175196.47                      Prob > chi2       =     0.0000
------------------------------------------------------------------------------
          _t | Haz. ratio   Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         1.x |   2.713138   .0184908   146.45   0.000     2.677137    2.749622
       _cons |   2.564135   .0797385    30.28   0.000     2.412518    2.725281
-------------+----------------------------------------------------------------
       /ln_p |  -.6925791   .0024746                     -.6974291    -.687729
-------------+----------------------------------------------------------------
group        |
   var(_cons) |    .472804   .0303096                      .4169789     .536103
------------------------------------------------------------------------------
Note: Estimates are transformed only in the first equation to hazard ratios.
Note: _cons estimates baseline hazard (conditional on zero random effects).
LR test vs. Weibull model: chibar2(01) = 35800.39     Prob >= chibar2 = 0.0000
. predict hcond, hazard conditional(ebmeans)
(predictions based on fixed effects and posterior means of random effects)
. gen loghcond0 = log(hcond) if x==0
(49,991 missing values generated)
. gen loghcond1 = log(hcond) if x==1
(50,009 missing values generated)
. gen zcond = loghcond0 + _b[_t:1.x]
(49,991 missing values generated)
. sort _t group
```

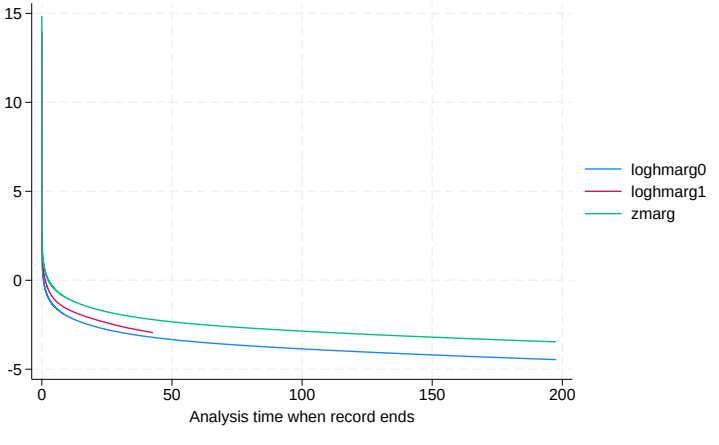
```
. twoway line loghcond0 loghcond1 zcond _t if group==1
```



In the graph above, the line for `loghcond1` cannot be distinguished from the line for `zcond` for most of the distribution. This illustrates that the PH assumption is satisfied for the conditional hazard. Notice that you can still see a part of `loghcond1` near the origin. This is because the two variables correspond to different values of `_t` and only `loghcond1` happens to be defined at the early values.

Now, we make the same computation for the marginal hazard.

```
. predict hmarg, hazard marginal
. gen loghmarg0 = log(hmarg) if x==0
(49,991 missing values generated)
. gen loghmarg1 = log(hmarg) if x==1
(50,009 missing values generated)
. gen zmarg = loghmarg0 + _b[_t:1.x]
(49,991 missing values generated)
. sort _t group
. twoway line loghmarg0 loghmarg1 zmarg _t
```



The curve for zmarg is clearly different from the curve for loghmarg1, demonstrating that the marginal distribution does not meet the PH assumption. Notice that the line for loghmarg1 is shorter than the others. This is because predictions are obtained at the values of _t in the dataset. These values of _t were simulated based on the model, which determines that observations with x==1 fail earlier.

◁

Methods and formulas

Methods and formulas for predicting random effects and other statistics are given in *Methods and formulas* of [ME] **meglm postestimation**. Statistics of special interest for survival analysis are described below.

predict *newvar* with the conditional() option computes the following predictions:

median:

$$newvar_{ji} = \{t : \hat{S}(t|\mathbf{x}_{ji}, \hat{u}_{ji}) = 1/2\}$$

where $\hat{S}(t|\mathbf{x}_{ji}, \hat{u}_{ji})$ is $S(t|\mathbf{x}_{ji}\widehat{\boldsymbol{\beta}} + \hat{u}_{ji})$, where $\hat{u}_{ji}$ are the empirical Bayes predictions for u_{ji}. If conditional(fixedonly) is specified, then 0 is substituted for $\hat{u}_{ji}$.

mean:

$$newvar_{ji} = \int_0^\infty \hat{S}(t|\mathbf{x}_{ji}, u_{ji})dt$$

surv:

$$newvar_{ji} = \hat{S}(t_{ji}|\mathbf{x}_{ji}, \hat{u}_{ji})$$

hazard:

$$newvar_{ji} = \hat{g}(t_{ji}|\mathbf{x}_{ji}, \hat{u}_{ji})/\hat{S}(t_{ji}|\mathbf{x}_{ji}, \hat{u}_{ji})$$

where $\hat{g}(t|\mathbf{x}_{ji}, u_{ji})$ is the density $g(t|\mathbf{x}_{ji}\widehat{\boldsymbol{\beta}} + \hat{u}_{ji})$.

When the marginal option is used with mean or surv, the prediction is computed marginally with respect to the random effects. That is, the prediction is integrated over the random-effects distributions. When the marginal option is used with hazard, the hazard for the marginal distribution is computed. That is, the predicted hazard is computed as the quotient of the marginal hazard and the marginal survivor function.

Also see

[ME] **mestreg** — Multilevel mixed-effects parametric survival models

[ME] **meglm postestimation** — Postestimation tools for meglm

[ME] **mixed postestimation** — Postestimation tools for mixed

[ST] **stcurve** — Plot the survivor or related function after streg, stcox, and more

[U] **20 Estimation and postestimation commands**

metobit — Multilevel mixed-effects tobit regression

Description Quick start Menu Syntax Options
Remarks and examples Stored results Methods and formulas Also see

Description

`metobit` fits mixed-effects models for continuous responses where the outcome variable is censored. Censoring limits may be fixed for all observations or vary across observations.

Quick start

Without weights

Two-level tobit regression of `y` on `x` with random intercepts by `lev2` where `y` is censored at a lower limit of 5

```
metobit y x || lev2:, ll(5)
```

Same as above, but specify that left-censoring occurs at 5 and right-censoring occurs at 25

```
metobit y x || lev2:, ll(5) ul(25)
```

Same as above, but where `lower` and `upper` are variables containing the censoring limits

```
metobit y x || lev2:, ll(lower) ul(upper)
```

Mixed-effects model adding random coefficients for `x`

```
metobit y x || lev2: x, ll(5)
```

Three-level random-intercept model of `y` on `x` with `lev2` nested within `lev3`

```
metobit y x || lev3: || lev2:, ll(5)
```

Crossed-effects model of `y` on `x` with two-way crossed random effects by factors `a` and `b`

```
metobit y x || _all:R.a || b:, ll(5)
```

With weights

Two-level tobit regression of `y` on `x` with random intercepts by `lev2` and observation-level frequency weights `wvar1`

```
metobit y x [fweight=wvar1] || lev2:, ll(5)
```

Two-level random-intercept model from a two-stage sampling design with PSUs identified by `psu` using PSU-level and observation-level sampling weights `wvar2` and `wvar1`, respectively

```
metobit y x [pweight=wvar1] || psu:, pweight(wvar2) ll(5)
```

Same as above, but `svyset` data first

```
svyset psu, weight(wvar2) || _n, weight(wvar1)
svy: metobit y x || psu:, ll(5)
```

Menu

Statistics > Multilevel mixed-effects models > Tobit regression

Syntax

```
metobit depvar fe_equation [|| re_equation] [|| re_equation ...] [, options]
```

where the syntax of *fe_equation* is

```
[indepvars] [if] [in] [weight] [, fe_options]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar: [varlist] [, re_options]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar: R.varname
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress constant term from the fixed-effects equation
`offset(`*varname*`)`	include *varname* in model with coefficient constrained to 1

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*varname*`)`	frequency weights at higher levels
`iweight(`*varname*`)`	importance weights at higher levels
`pweight(`*varname*`)`	sampling weights at higher levels

options	Description
Model	
`ll`[`(`*varname*`|`#`)`]	left-censoring variable or limit
`ul`[`(`*varname*`|`#`)`]	right-censoring variable or limit
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `opg`, `robust`, or `cluster` *clustvar*
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
`notable`	suppress coefficient table
`noheader`	suppress output header
`nogroup`	suppress table summarizing groups
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Integration	
`intmethod(`*intmethod*`)`	integration method
`intpoints(`#`)`	set the number of integration (quadrature) points for all levels; default is `intpoints(7)`
Maximization	
maximize_options	control the maximization process; seldom used
`startvalues(`*svmethod*`)`	method for obtaining starting values
`startgrid`[`(`*gridspec*`)`]	perform a grid search to improve starting values
`noestimate`	do not fit the model; show starting values instead
`dnumerical`	use numerical derivative techniques
`collinear`	keep collinear variables
`coeflegend`	display legend instead of statistics

vartype	Description
`independent`	one unique variance parameter per random effect and all covariances 0; the default unless the `R.` notation is used
`exchangeable`	equal variances for random effects and one common pairwise covariance
`identity`	equal variances for random effects and all covariances 0; the default if the `R.` notation is used
`unstructured`	all variances and covariances to be distinctly estimated
`fixed(`*matname*`)`	user-selected variances and covariances constrained to specified values; the remaining variances and covariances unrestricted
`pattern(`*matname*`)`	user-selected variances and covariances constrained to be equal; the remaining variances and covariances unrestricted

intmethod	Description
mvaghermite	mean–variance adaptive Gauss–Hermite quadrature; the default unless a crossed random-effects model is fit
mcaghermite	mode-curvature adaptive Gauss–Hermite quadrature
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `by`, `collect`, and `svy` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: metobit**.

`vce()` and weights are not allowed with the `svy` prefix; see [SVY] **svy**.

`fweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**. Only one type of weight may be specified. Weights are not supported under the Laplacian approximation or for crossed models.

`startvalues()`, `startgrid`, `noestimate`, `dnumerical`, `collinear`, and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

`ll`[`(`*varname*`|`#`)`] and `ul`[`(`*varname*`|`#`)`] indicate the lower and upper limits for censoring, respectively. Observations with *depvar* $\leq$ `ll()` are left-censored; observations with *depvar* $\geq$ `ul()` are right-censored; and remaining observations are not censored. You do not have to specify the censoring values. If you specify `ll`, the lower limit is the minimum of *depvar*. If you specify `ul`, the upper limit is the maximum of *depvar*.

`offset(`*varname*`)` specifies that *varname* be included in the fixed-effects portion of the model with the coefficient constrained to be 1.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, `unstructured`, `fixed(`*matname*`)`, or `pattern(`*matname*`)`.

`covariance(independent)` covariance structure allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0. The default is `covariance(independent)` unless a crossed random-effects model is fit, in which case the default is `covariance(identity)`.

`covariance(exchangeable)` structure specifies one common variance for all random effects and one common pairwise covariance.

`covariance(identity)` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`covariance(unstructured)` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(fixed(`*matname*`))` and `covariance(pattern(`*matname*`))` covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a `fixed(`*matname*`)` covariance structure, (co)variance (i, j) is constrained to equal the value specified in the i, jth entry of *matname*. In a `pattern(`*matname*`)` covariance structure, (co)variances (i, j) and (k, l) are constrained to be equal if $matname[i, j] = matname[k, l]$.

`fweight(`*varname*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [fw = wt1] || school: ... , fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`iweight(`*varname*`)` specifies importance weights at higher levels in a multilevel model, whereas importance weights at the first level (the observation level) are specified in the usual manner, for example, `[iw=`*iwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `iweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [iw = wt1] || school: ... , iweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) importance weights, and `wt2` would hold the second-level (the school-level) importance weights.

`pweight(`*varname*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *varname* can be any valid Stata variable name, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mecmd fixed_portion [pw = wt1] || school: ... , pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

`vce(`*vcetype*`)` specifies the type of standard error reported, which includes types that are derived from asymptotic theory (`oim`, `opg`), that are robust to some kinds of misspecification (`robust`), and that allow for intragroup correlation (`cluster` *clustvar*); see [R] ***vce_option***. If `vce(robust)` is specified, robust variances are clustered at the highest level in the multilevel model.

Reporting

`level(`#`)`, `nocnsreport`; see [R] **Estimation options**.

`notable` suppresses the estimation table, either at estimation or upon replay.

`noheader` suppresses the output header, either at estimation or upon replay.

`nogroup` suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Integration

intmethod(*intmethod*) specifies the integration method to be used for the random-effects model. mvaghermite performs mean–variance adaptive Gauss–Hermite quadrature; mcaghermite performs mode-curvature adaptive Gauss–Hermite quadrature; ghermite performs nonadaptive Gauss–Hermite quadrature; and laplace performs the Laplacian approximation, equivalent to mode-curvature adaptive Gaussian quadrature with one integration point.

The default integration method is mvaghermite unless a crossed random-effects model is fit, in which case the default integration method is laplace. The Laplacian approximation has been known to produce biased parameter estimates; however, the bias tends to be more prominent in the estimates of the variance components rather than in the estimates of the fixed effects.

For crossed random-effects models, estimation with more than one quadrature point may be prohibitively intensive even for a small number of levels. For this reason, the integration method defaults to the Laplacian approximation. You may override this behavior by specifying a different integration method.

intpoints(#) sets the number of integration points for quadrature. The default is intpoints(7), which means that seven quadrature points are used for each level of random effects. This option is not allowed with intmethod(laplace).

The more integration points, the more accurate the approximation to the log likelihood. However, computation time increases as a function of the number of quadrature points raised to a power equaling the dimension of the random-effects specification. In crossed random-effects models and in models with many levels or many random coefficients, this increase can be substantial.

Maximization

maximize_options: difficult, technique(*algorithm_spec*), iterate(#), [no]log, trace, gradient, showstep, hessian, showtolerance, tolerance(#), ltolerance(#), nrtolerance(#), nonrtolerance, and from(*init_specs*); see [R] **Maximize**. Those that require special mention for metobit are listed below.

from() accepts a properly labeled vector of initial values or a list of coefficient names with values. A list of values is not allowed.

The following options are available with metobit but are not shown in the dialog box:

startvalues(*svmethod*), startgrid[(*gridspec*)], noestimate, and dnumerical; see [ME] **meglm**.

collinear, coeflegend; see [R] **Estimation options**.

Remarks and examples

Mixed-effects tobit regression is tobit regression containing both fixed effects and random effects. In longitudinal data and panel data, random effects are useful for modeling intracluster correlation; that is, observations in the same cluster are correlated because they share common cluster-level random effects.

In a mixed-effects tobit regression, the values of the outcome variable may be observed, unobserved but known to fall below a given limit (left-censored data), or unobserved but known to fall above a given limit (right-censored data). That is, the observed data, y^*_{ij}, represent possibly censored versions of y_{ij} for the ith observation within the jth cluster.

The observed outcome is therefore defined as

$$y^*_{ij} = \begin{cases} y_{ij} & \text{if } a < y_{ij} < b \\ a & \text{if } y_{ij} \leq a \\ b & \text{if } y_{ij} \geq b \end{cases}$$

where a is the lower-censoring limit and b is the upper-censoring limit. If the data are uncensored, $y^*_{ij} = y_{ij}$, and the value is determined by the value of the outcome variable. If they are left-censored, all that is known is that $y_{ij} \leq a$ and y^*_{ij} is determined by `ll()`. If they are right-censored, all that is known is that $y_{ij} \geq b$ and y^*_{ij} is determined by `ul()`. The censoring limits specified in `ll()` and `ul()` can be the same for all observations or can vary from observation to observation.

Regardless of the type of censoring, the expected value of the underlying dependent variable—say, **y**—is modeled using the following linear prediction:

$$E(\mathbf{y}|\mathbf{X}, \mathbf{u}) = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} \tag{1}$$

X is an $n \times p$ design/covariate matrix, analogous to the covariates you would find in a standard linear regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. **Z** is the $n \times q$ design/covariate matrix for the random effects **u**. This linear prediction also contains the offset when `offset()` is specified.

The columns of matrix **Z** are the covariates corresponding to the random effects and can be used to represent both random intercepts and random coefficients. For example, in a random-intercepts model, **Z** is simply the scalar 1. The random effects **u** are realizations from a multivariate normal distribution with mean **0** and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The random effects are not directly estimated as model parameters but are instead summarized according to the unique elements of $\boldsymbol{\Sigma}$, known as variance components. One special case of (1) places $\mathbf{Z} = \mathbf{X}$ so that all covariate effects are essentially random and distributed as multivariate normal with mean $\boldsymbol{\beta}$ and variance $\boldsymbol{\Sigma}$.

Below we present a short example of mixed-effects tobit regression; refer to [ME] **me** and [ME] **meglm** for additional examples of random-effects models. A two-level tobit model can also be fit using `xttobit`; see [XT] **xttobit**. In the absence of random effects, mixed-effects tobit regression reduces to standard tobit regression; see [R] **tobit**.

▷ Example 1: Random-intercept model

We have wage data on young women who were between ages 14 and 24 in 1968 and who were surveyed over the period 1968–1988; see [XT] **xt** for a more detailed discussion of the data. We are interested in the effect of completed years of schooling, current age, union membership, and residence in the South on wages.

```
. use https://www.stata-press.com/data/r19/nlswork
(National Longitudinal Survey of Young Women, 14-24 years old in 1968)
```

We fit a mixed-effects tobit model of the log of inflation-adjusted wages (`ln_wage`). For illustration purposes, we use the `ul()` option to impose an artificial upper limit at 1.96, the 75th percentile of the recorded log wages.

```
. metobit ln_wage i.union age south##c.grade || idcode:, ul(1.96)
Fitting fixed-effects model:

Iteration 0:  Log likelihood = -11628.188
Iteration 1:  Log likelihood = -10617.455
Iteration 2:  Log likelihood = -10555.304
Iteration 3:  Log likelihood =  -10554.78
Iteration 4:  Log likelihood =  -10554.78

Refining starting values:

Grid node 0:  Log likelihood = -10225.917

Fitting full model:

Iteration 0:  Log likelihood = -10225.917  (not concave)
Iteration 1:  Log likelihood = -8728.9674  (not concave)
Iteration 2:  Log likelihood = -7827.6894  (not concave)
Iteration 3:  Log likelihood = -7112.0272
Iteration 4:  Log likelihood = -6894.0253
Iteration 5:  Log likelihood = -6821.7055
Iteration 6:  Log likelihood = -6818.5592
Iteration 7:  Log likelihood = -6818.5512
Iteration 8:  Log likelihood = -6818.5512

Mixed-effects tobit regression                  Number of obs     =     19,224
                                                   Uncensored     =     13,188
Limits: Lower = -inf                               Left-censored  =          0
        Upper = 1.96                               Right-censored =      6,036

Group variable: idcode                          Number of groups  =      4,148
                                                Obs per group:
                                                              min =          1
                                                              avg =        4.6
                                                              max =         12

Integration method: mvaghermite                 Integration pts.  =          7

                                                Wald chi2(5)      =    2812.43
Log likelihood = -6818.5512                     Prob > chi2       =     0.0000
```

ln_wage	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
1.union	.1418088	.0068398	20.73	0.000	.1284029	.1552146
age	.0107585	.0004068	26.45	0.000	.0099612	.0115559
1.south	-.2373995	.048346	-4.91	0.000	-.3321559	-.1426431
grade	.0763865	.0029104	26.25	0.000	.0706822	.0820909
south# c.grade 1	.0099306	.0037452	2.65	0.008	.0025902	.0172709
_cons	.4146363	.0396691	10.45	0.000	.3368864	.4923862
idcode var(_cons)	.0985482	.003018			.0928071	.1046444
var(e.ln_w~e)	.0619327	.000876			.0602394	.0636736

```
LR test vs. tobit model: chibar2(01) = 7472.46        Prob >= chibar2 = 0.0000
```

The estimation table reports the fixed effects, which are interpreted just as you would the output from `tobit`, and the estimated variance components. Because the dependent variable is log transformed, the fixed-effects coefficients can be interpreted in terms of a percent change. For example, we see that on average, union members make 14.2% more than nonunion members and that each additional year of age is associated with a 1.1% increase in wages.

The random-effects equation is labeled idcode. The estimated variance of the subject-specific random intercept is 0.099 with standard error 0.003. A likelihood-ratio test comparing the model with a tobit model without random effects is provided under the table and indicates that the two-level tobit model is preferred.

◁

Stored results

metobit stores the following in e():

Scalars

e(N)	number of observations
e(N_unc)	number of uncensored observations
e(N_lc)	number of left-censored observations
e(N_rc)	number of right-censored observations
e(k)	number of parameters
e(k_dv)	number of dependent variables
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(N_clust)	number of clusters
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	meglm
e(cmd2)	metobit
e(cmdline)	command as typed
e(depvar)	names of dependent variables
e(llopt)	minimum of *depvar* or contents of ll()
e(ulopt)	maximum of *depvar* or contents of ul()
e(wtype)	weight type
e(wexp)	weight expression (first-level weights)
e(fweight*k*)	fweight variable for *k*th highest level, if specified
e(iweight*k*)	iweight variable for *k*th highest level, if specified
e(pweight*k*)	pweight variable for *k*th highest level, if specified
e(covariates)	list of covariates
e(ivars)	grouping variables
e(model)	tobit
e(title)	title in estimation output
e(link)	identity
e(family)	gaussian
e(clustvar)	name of cluster variable
e(offset)	offset
e(intmethod)	integration method

e(n_quad)	number of integration points
e(chi2type)	Wald; type of model χ^2
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(which)	max or min; whether optimizer is to perform maximization or minimization
e(ml_method)	type of ml method
e(user)	name of likelihood-evaluator program
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins
e(marginswtype)	weight type for margins
e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved

Matrices

e(b)	coefficient vector
e(Cns)	constraints matrix
e(ilog)	iteration log (up to 20 iterations)
e(gradient)	gradient vector
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices

r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Without a loss of generality, consider a two-level regression model

$$E(\mathbf{y}_j|\mathbf{X}_j,\mathbf{u}_j) = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j \qquad \mathbf{y} \sim \text{normal}$$

for $j = 1,\ldots,M$ clusters, with the jth cluster consisting of n_j observations, where, for the jth cluster, $\mathbf{y}_j$ is the $n_j \times 1$ censored response vector, $\mathbf{X}_j$ is the $n_j \times p$ matrix of fixed predictors, $\mathbf{Z}_j$ is the $n_j \times q$ matrix of random predictors, $\mathbf{u}_j$ is the $q \times 1$ vector of random effects, and $\boldsymbol{\beta}$ is the $p \times 1$ vector of regression coefficients on the fixed predictors. The random effects, $\mathbf{u}_j$, are assumed to be multivariate normal with mean $\mathbf{0}$ and variance $\boldsymbol{\Sigma}$.

Let $\boldsymbol{\eta}_j$ be the linear predictor, $\boldsymbol{\eta}_j = \mathbf{X}_j\boldsymbol{\beta}+\mathbf{Z}_j\mathbf{u}_j$, that also includes the offset variable when offset() is specified. y_{ij} and η_{ij} are the ith individual elements of $\mathbf{y}_j$ and $\boldsymbol{\eta}_j$, $i = 1,\ldots,n_j$. a_{ij} refers to the lower limit for observation ij, and b_{ij} refers to the upper limit for observation ij. The conditional density

function for the response at observation ij is then

$$f(y^*_{ij}|\eta_{ij}) = \begin{cases} \left(\sqrt{2\pi}\sigma_\epsilon\right)^{-1} \exp^{-(y_{ij}-\eta_{ij})^2/(2\sigma^2_\epsilon)} & \text{if } y_{ij} = y^*_{ij} \\ \Phi\left(\frac{a_{ij}-\eta_{ij}}{\sigma_\epsilon}\right) & \text{if } y_{ij} \leq y^*_{ij} \\ 1 - \Phi\left(\frac{b_{ij}-\eta_{ij}}{\sigma_\epsilon}\right) & \text{if } y_{ij} \geq y^*_{ij} \end{cases}$$

where $\Phi(\cdot)$ is the cumulative normal distribution.

Because the observations are assumed to be conditionally independent, the conditional log density function for cluster j is

$$\log f(\mathbf{y}^*_j|\boldsymbol{\eta}_j) = \sum_{j=1}^{n_i} \log f(y^*_{ij}|\eta_{ij})$$

and the likelihood function for cluster j is given by

$$\begin{aligned} \mathcal{L}_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= (2\pi)^{-q/2}|\boldsymbol{\Sigma}|^{-1/2} \int_{\Re^q} f(\mathbf{y}^*_j|\boldsymbol{\eta}_j) \exp\left(-\frac{1}{2}\mathbf{u}'_j\boldsymbol{\Sigma}^{-1}\mathbf{u}_j\right) d\mathbf{u}_j \\ &= (2\pi)^{-q/2}|\boldsymbol{\Sigma}|^{-1/2} \int_{\Re^q} \exp\left\{\log f(\mathbf{y}^*_j|\boldsymbol{\eta}_j) - \frac{1}{2}\mathbf{u}'_j\boldsymbol{\Sigma}^{-1}\mathbf{u}_j\right\} d\mathbf{u}_j \end{aligned} \tag{2}$$

where $\Re$ denotes the set of values on the real line and $\Re^q$ is the analog in q-dimensional space.

The integration in (2) has no closed form and thus must be approximated; see *Methods and formulas* in [ME] **meglm** for details.

`metobit` supports multilevel weights and survey data; see *Methods and formulas* in [ME] **meglm** for details.

Also see

[ME] **metobit postestimation** — Postestimation tools for metobit

[ME] **meintreg** — Multilevel mixed-effects interval regression

[ME] **me** — Introduction to multilevel mixed-effects models

[BAYES] **bayes: metobit** — Bayesian multilevel tobit regression

[R] **tobit** — Tobit regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[SVY] **svy estimation** — Estimation commands for survey data

[XT] **xttobit** — Random-effects tobit model

[U] 20 Estimation and postestimation commands

metobit postestimation — Postestimation tools for metobit

Postestimation commands | predict | margins
Remarks and examples | Methods and formulas | Also see

Postestimation commands

The following postestimation commands are of special interest after `metobit`:

Command	Description
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat sd`	display variance components as standard deviations and correlations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estat (svy)`	postestimation statistics for survey data
`estimates`	cataloging estimation results
`etable`	table of estimation results
* `hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
* `lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	means, probabilities, densities, REs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

*`hausman` and `lrtest` are not appropriate with `svy` estimation results.

predict

Description for predict

`predict` creates a new variable containing predictions such as linear predictions, standard errors, probabilities, and expected values.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*] [, *statistic options*]

Syntax for obtaining estimated random effects and their standard errors

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `reffects` [*re_options*]

Syntax for obtaining ML scores

`predict` [*type*] { *stub*`*` | *newvars* } [*if*] [*in*], `scores`

statistic	Description
Main	
`eta`	fitted linear predictor; the default
`xb`	linear predictor for the fixed portion of the model only
`stdp`	standard error of the fixed-portion linear prediction
`pr(`*a*`,`*b*`)`	$\Pr(a < y < b)$
`e(`*a*`,`*b*`)`	$E(y \mid a < y < b)$
`ystar(`*a*`,`*b*`)`	$E(y^*)$, $y^* = \max\{a, \min(y, b)\}$

These statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample.

where *a* and *b* may be numbers or variables; *a* missing ($a \geq .$) means $-\infty$, and *b* missing ($b \geq .$) means $+\infty$; see **[U] 12.2.1 Missing values**.

options	Description
Main	
`conditional(`*ctype*`)`	compute *statistic* conditional on estimated random effects; default is `conditional(ebmeans)`
`marginal`	compute *statistic* marginally with respect to the random effects
`nooffset`	make calculation ignoring offset or exposure
Integration	
int_options	integration options

ctype	Description
`ebmeans`	empirical Bayes means of random effects; the default
`ebmodes`	empirical Bayes modes of random effects
`fixedonly`	prediction for the fixed portion of the model only

re_options	Description
Main	
`ebmeans`	use empirical Bayes means of random effects; the default
`ebmodes`	use empirical Bayes modes of random effects
`reses(`*stub*`*` | *newvars*`)`	calculate standard errors of empirical Bayes estimates
Integration	
int_options	integration options

int_options	Description
`intpoints(#)`	use # quadrature points to compute marginal predictions and empirical Bayes means
`iterate(#)`	set maximum number of iterations in computing statistics involving empirical Bayes estimators
`tolerance(#)`	set convergence tolerance for computing statistics involving empirical Bayes estimators

Options for predict

Main

`eta`, the default, calculates the fitted linear prediction.

`pr(`*a*`,`*b*`)` calculates estimates of $\Pr(a < y < b)$, which is the probability that y would be observed in the interval (a, b).

a and *b* may be specified as numbers or variable names; *lb* and *ub* are variable names;
`pr(20,30)` calculates $\Pr(20 < y < 30)$;
`pr(`*lb*`,`*ub*`)` calculates $\Pr(lb < y < ub)$; and
`pr(20,`*ub*`)` calculates $\Pr(20 < y < ub)$.

a missing ($a \geq .$) means $-\infty$; `pr(.,30)` calculates $\Pr(-\infty < y < 30)$;
`pr(`*lb*`,30)` calculates $\Pr(-\infty < y < 30)$ in observations for which $lb \geq .$
(and calculates $\Pr(lb < y < 30)$ elsewhere).

b missing ($b \geq .$) means $+\infty$; `pr(20,.)` calculates $\Pr(+\infty > y > 20)$;
`pr(20,`*ub*`)` calculates $\Pr(+\infty > y > 20)$ in observations for which $ub \geq .$
(and calculates $\Pr(20 < y < ub)$ elsewhere).

`e(`*a*`,`*b*`)` calculates estimates of $E(y \mid a < y < b)$, which is the expected value of y conditional on y being in the interval (a, b), meaning that y is truncated. *a* and *b* are specified as they are for `pr()`.

`ystar(`*a*`,`*b*`)` calculates estimates of $E(y^*)$, where $y^* = a$ if $y \leq a$, $y^* = b$ if $y \geq b$, and $y^* = y$ otherwise, meaning that y^* is the censored version of y. *a* and *b* are specified as they are for `pr()`.

`xb`, `stdp`, `scores`, `conditional()`, `marginal`, and `nooffset`; see [ME] **meglm postestimation**.

`reffects`, `ebmeans`, `ebmodes`, and `reses()`; see [ME] **meglm postestimation**.

Integration

`intpoints()`, `iterate()`, `tolerance()`; see [ME] **meglm postestimation**.

margins

Description for margins

`margins` estimates margins of response for linear predictions, probabilities, and expected values.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [, *options*]

`margins` [*marginlist*] , `predict(`*statistic* ...`)` [`predict(`*statistic* ...`)` ...] [*options*]

statistic	Description
`eta`	fitted linear predictor; the default
`xb`	linear predictor for the fixed portion of the model only
`pr(`*a*`,`*b*`)`	$\Pr(a < y < b)$
`e(`*a*`,`*b*`)`	$E(y \mid a < y < b)$
`ystar(`*a*`,`*b*`)`	$E(y^*)$, $y^* = \max\{a, \min(y, b)\}$
`stdp`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed-effects tobit model with `metobit`.

The `predict` command allows us to compute marginal and conditional predictions. Unless stated differently, we use the word "conditional" to mean "conditional on the empirical Bayes predictions of the random effects." The default prediction is the linear prediction, `eta`, which is the expected value of the unobserved censored variable. Predictions of expected values for censored and truncated versions of the response are also available.

▷ Example 1: Predicting censored and uncensored means

In example 1 of [ME] **metobit**, we analyzed wages for a subpopulation from the National Longitudinal Survey. The dependent variable is the logarithm of wage, and we fit a model that assumes that the data are right-censored at 1.9.

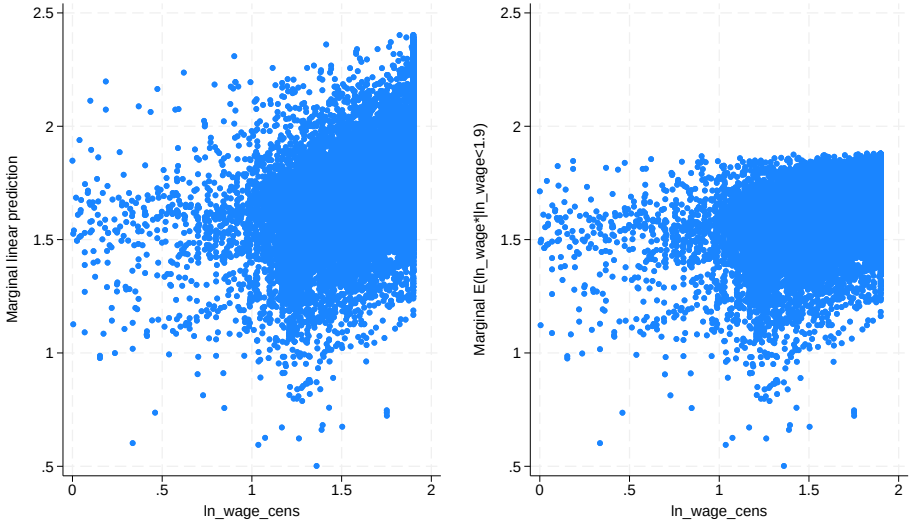
```
. use https://www.stata-press.com/data/r19/nlswork3
(National Longitudinal Survey of Young Women, 14-24 years old in 1968)
. metobit ln_wage union age south##c.grade || idcode:, ul(1.9)
  (output omitted)
```

Below, we use `predict` to predict both the mean for the (unobserved) uncensored variable and the (censored) observed values. We also manually generate the censored version of `ln_wage`.

```
. predict uncens_pred, eta marginal
(9310 missing values generated)
. predict cens_pred, ystar(.,1.9) marginal
. generate double ln_wage_cens = min(ln_wage,1.9)
```

To see how the two predictions differ, we can plot them side by side against the censored wage (ln_wage_cens).

```
. scatter uncens_pred ln_wage_cens, name(gr1) xsize(4) ysize(4)
. scatter cens_pred ln_wage_cens, name(gr2) xsize(4) ysize(4)
. graph combine gr1 gr2, ycommon
```



We see that many of the predictions for the uncensored variable exceed the censoring point, while the predictions for the censored variable never fall above the upper-censoring limit.

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction
Conditional predictions
Marginal predictions
Marginal variance of the linear predictor

Introduction

This postestimation entry presents the methods and formulas used to calculate the `pr()`, `e()`, and `ystar()` statistics. See *Methods and formulas* of [ME] **estat icc** for a discussion of intraclass correlations. See *Methods and formulas* of [ME] **meglm postestimation** for a discussion of the remaining postestimation features.

Recall that in a two-level model, the linear predictor for any ith observation in the jth cluster is defined as $\eta_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j$. Let $\ell\ell_{ij}$ represent a lower bound for y_{ij} and $u\ell_{ij}$ represent an upper bound.

Conditional predictions

The probability that $y_{ij}|\hat{\eta}_{ij}$ is observed in the interval $(\ell\ell_{ij}, u\ell_{ij})$—the `pr(a,b)` option—is calculated as

$$\text{pr}(\ell\ell_{ij}, u\ell_{ij}) = \Pr(\ell\ell_{ij} < \hat{\eta}_{ij} + \epsilon_{ij} < u\ell_{ij}) = \Phi\left(\frac{u\ell_{ij} - \hat{\eta}_{ij}}{\hat{\sigma}_\epsilon}\right) - \Phi\left(\frac{\ell\ell_{ij} - \hat{\eta}_{ij}}{\hat{\sigma}_\epsilon}\right)$$

where $\hat{\sigma}_\epsilon$ is the estimated error standard deviation.

The `e(a,b)` option computes the expected value of $y_{ij}|\hat{\eta}_{ij}$ conditional on $y_{ij}|\hat{\eta}_{ij}$ being in the interval $(\ell\ell_{ij}, u\ell_{ij})$, that is, when $y_{ij}|\hat{\eta}_{ij}$ is truncated. The expected value is calculated as

$$\begin{aligned}\text{e}(\ell\ell_{ij}, u\ell_{ij}) &= E(\hat{\eta}_{ij} + \epsilon_{ij} \mid \ell\ell_{ij} < \hat{\eta}_{ij} + \epsilon_{ij} < u\ell_{ij}) \\ &= \hat{\eta}_{ij} - \hat{\sigma}_\epsilon \frac{\phi\left(\frac{u\ell_{ij} - \hat{\eta}_{ij}}{\hat{\sigma}_\epsilon}\right) - \phi\left(\frac{\ell\ell_{ij} - \hat{\eta}_{ij}}{\hat{\sigma}_\epsilon}\right)}{\Phi\left(\frac{u\ell_{ij} - \hat{\eta}_{ij}}{\hat{\sigma}_\epsilon}\right) - \Phi\left(\frac{\ell\ell_{ij} - \hat{\eta}_{ij}}{\hat{\sigma}_\epsilon}\right)}\end{aligned}$$

where ϕ is the normal density and Φ is the cumulative normal distribution.

You can also compute `ystar(a,b)`—the expected value of $y_{ij}|\hat{\eta}_{ij}$, where y_{ij} is assumed censored at $\ell\ell_{ij}$ and $u\ell_{ij}$:

$$y^*_{ij} = \begin{cases} \ell\ell_{ij} & \text{if } y_{ij} \le \ell\ell_{ij} \\ \eta_{ij} + \epsilon_{ij} & \text{if } \ell\ell_{ij} < y_{ij} < u\ell_{ij} \\ u\ell_{ij} & \text{if } y_{ij} \ge u\ell_{ij} \end{cases}$$

This computation can be expressed in several ways, but the most intuitive formulation involves a combination of the two statistics just defined:

$$E(y^*_{ij}) = \text{pr}(-\infty, \ell\ell_{ij})\ell\ell_{ij} + \text{pr}(\ell\ell_{ij}, u\ell_{ij})\text{e}(\ell\ell_{ij}, u\ell_{ij}) + \text{pr}(u\ell_{ij}, +\infty)u\ell_{ij}$$

Marginal predictions

When the `marginal` option is specified, the `pr()` statistic is calculated as

$$\text{pr}(\ell\ell_{ij}, u\ell_{ij}) = \Phi\left(\frac{u\ell_{ij} - \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}}{\hat{s}_{ij}}\right) - \Phi\left(\frac{\ell\ell_{ij} - \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}}{\hat{s}_{ij}}\right)$$

where $\hat{s}_{ij}$ is the square root of the estimated marginal variance of the linear predictor, defined in detail below.

The marginal `e()` statistic is calculated as

$$\text{e}(\ell\ell_{ij}, u\ell_{ij}) = \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}} - \hat{s}_{ij}\frac{\phi\left(\frac{u\ell_{ij} - \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}}{\hat{s}_{ij}}\right) - \phi\left(\frac{\ell\ell_{ij} - \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}}{\hat{s}_{ij}}\right)}{\Phi\left(\frac{u\ell_{ij} - \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}}{\hat{s}_{ij}}\right) - \Phi\left(\frac{\ell\ell_{ij} - \mathbf{x}_{ij}\widehat{\boldsymbol{\beta}}}{\hat{s}_{ij}}\right)}$$

and the marginal `ystar()` statistic is calculated as above with marginal predictions used in place of the conditional ones.

Marginal variance of the linear predictor

In a two-level model, the marginal variance for observation ij is given by

$$\sigma^2_{ij} = \sigma^2_\epsilon + \mathbf{z}_{ij}\boldsymbol{\Sigma}_2\mathbf{z}'_{ij}$$

where σ^2_ϵ is the error variance at level 1 and $\boldsymbol{\Sigma}_2$ is the variance matrix of the random effects at level 2. The marginal standard deviation is $s_{ij} = \sqrt{\sigma^2_{ij}}$.

In general, for a G-level random-effects model, the marginal variance for one observation is given by

$$\sigma^2 = \sigma^2_\epsilon + \sum_{g=2}^{G} \mathbf{z}_g\boldsymbol{\Sigma}_g\mathbf{z}'_g$$

where $\mathbf{z}_g$ is a row vector of the covariates at level g for that observation and $\boldsymbol{\Sigma}_g$ is the variance matrix of the random effects at level g.

Also see

[ME] **metobit** — Multilevel mixed-effects tobit regression

[ME] **meglm postestimation** — Postestimation tools for meglm

[U] 20 Estimation and postestimation commands

mixed — Multilevel mixed-effects linear regression[+]

[+] This command includes features that are part of StataNow.

Description | Quick start | Menu | Syntax
Options | Remarks and examples | Stored results | Methods and formulas
Acknowledgments | References | Also see

Description

`mixed` fits linear mixed-effects models. These models are also known as multilevel models or hierarchical linear models. The overall error distribution of the linear mixed-effects model is assumed to be Gaussian, and heteroskedasticity and correlations within lowest-level groups also may be modeled.

Quick start

Linear mixed-effects model of `y` on `x` with random intercepts by `lev2`

```
mixed y x || lev2:
```

Same as above, but perform restricted maximum-likelihood (REML) estimation instead of the default maximum likelihood (ML) estimation

```
mixed y x || lev2:, reml
```

Same as above, but perform small-sample inference on `x` using the Kenward–Roger degrees of freedom (DF) method

```
mixed y x || lev2:, reml dfmethod(kroger)
```

Add random coefficients on `x`

```
mixed y x || lev2: x
```

Same as above, but allow correlation between the random slopes and intercepts

```
mixed y x || lev2: x, covariance(unstructured)
```

Three-level model with random intercepts by `lev2` and `lev3` for `lev2` nested within `lev3`

```
mixed y x || lev3: || lev2:
```

Crossed-effects model with two-way crossed effects by factors `a` and `b`

```
mixed y x || _all:R.a || b:
```

Menu

Statistics > Multilevel mixed-effects models > Linear regression

Syntax

```
mixed depvar fe_equation [ || re_equation ] [ || re_equation ... ] [ , options ]
```

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ weight ] [ , fe_options ]
```

and the syntax of *re_equation* is one of the following:

for random coefficients and intercepts

```
levelvar : [ varlist ] [ , re_options ]
```

for random effects among the values of a factor variable in a crossed-effects model

```
levelvar : R.varname [ , re_options ]
```

levelvar is a variable identifying the group structure for the random effects at that level or is `_all` representing one group comprising all observations.

fe_options	Description
Model	
`noconstant`	suppress constant term from the fixed-effects equation
`constraints(`*constraints*`)`	apply specified linear constraints

re_options	Description
Model	
`covariance(`*vartype*`)`	variance–covariance structure of the random effects
`noconstant`	suppress constant term from the random-effects equation
`fweight(`*exp*`)`	frequency weights at higher levels
`pweight(`*exp*`)`	sampling weights at higher levels
`collinear`	keep collinear variables

options	Description
Model	
mle	fit model via maximum likelihood; the default
reml	fit model via restricted maximum likelihood
dfmethod(*df_method*)	specify method for computing DF of a t distribution
residuals(*restype*[, *resopts*])	structure of residual errors
pwscale(*scale_method*)	control scaling of sampling weights in two-level models
SE/Robust	
vce(*vcetype*)	*vcetype* may be oim, robust, or cluster *clustvar*; types other than oim may not be combined with dfmethod()
Reporting	
level(#)	set confidence level; default is level(95)
variance	show random-effects and error parameter estimates as variances and covariances; the default
stddeviations	show random-effects and error parameter estimates as standard deviations and correlations
dftable(*dftable*)	specify contents of fixed-effects table; requires dfmethod() at estimation
noretable	suppress random-effects table
nofetable	suppress fixed-effects table
estmetric	show parameter estimates as stored in e(b)
noheader	suppress output header
nogroup	suppress table summarizing groups
nostderr	do not estimate standard errors of random-effects parameters
nocnsreport	do not display constraints
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
EM options	
emiterate(#)	number of EM iterations; default is emiterate(20)
emtolerance(#)	EM convergence tolerance; default is emtolerance(1e-10)
emonly	fit model exclusively using EM
emlog	show EM iteration log
emdots	show EM iterations as dots
Maximization	
maximize_options	control the maximization process; seldom used
matsqrt	parameterize variance components using matrix square roots; the default
matlog	parameterize variance components using matrix logarithms
small	replay small-sample inference results
coeflegend	display legend instead of statistics

vartype	Description
independent	one unique variance parameter per random effect, all covariances 0; the default unless the R. notation is used
exchangeable	equal variances for random effects, and one common pairwise covariance
identity	equal variances for random effects, all covariances 0; the default if the R. notation is used
unstructured	all variances and covariances to be distinctly estimated

df_method	Description
residual	residual degrees of freedom, $n - \text{rank}(X)$
repeated	repeated-measures ANOVA
anova	ANOVA
satterthwaite[, *dfopts*]	generalized Satterthwaite approximation; REML estimation only
kroger[, *dfopts*]	Kenward–Roger; REML estimation only

restype	Description
independent	i.i.d. Gaussian within-group errors with one common variance; the default
exchangeable	within-group errors with equal variances and one common covariance
ar [#]	within-group errors with autoregressive (AR) structure of order #, AR(#); ar 1 is implied by ar
ma [#]	within-group errors with moving-average (MA) structure of order #, MA(#); ma 1 is implied by ma
unstructured	within-group errors with distinct variances and covariances
banded [#]	within-group errors with distinct variances and covariances within first # off-diagonals; banded implies all matrix bands (unstructured)
toeplitz [#]	within-group errors have Toeplitz structure of order #; toeplitz implies that all matrix off-diagonals be estimated
exponential	within-group errors with an exponential function for the pairwise correlations and one overall error variance
+spexponential	within-group errors with a spatial exponential function for the pairwise correlations with a spatial scale parameter and one overall error variance
+sppower	within-group errors with a spatial power function for the pairwise correlations with a positive correlation parameter and one overall error variance
+spgaussian	within-group errors with a spatial Gaussian function for the pairwise correlations with a spatial scale parameter and one overall error variance
+spspherical	within-group errors with a spatial spherical function for the pairwise correlations with a range parameter and one overall error variance

+These features are part of StataNow.

scale_method	Description
size	scale first-level (observation-level) weights to sum to the sample size of their corresponding second-level cluster
effective	scale first-level weights to sum to the effective sample size of their corresponding second-level cluster
gk	set second-level weights to the cluster averages of the products of the weights at both levels and first-level weights to 1

dftable	Description
default	test statistics, p-values, and confidence intervals; the default
ci	DFs and confidence intervals
pvalue	DFs, test statistics, and p-values

indepvars and *varlist* may contain factor variables; see **[U] 11.4.3 Factor variables**.

depvar, *indepvars*, and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes`, `bootstrap`, `by`, `collect`, `jackknife`, `mi estimate`, `rolling`, and `statsby` are allowed; see **[U] 11.1.10 Prefix commands**. For more details, see [BAYES] **bayes: mixed**.

`mi estimate` is not allowed if `dfmethod()` is specified.

Weights are not allowed with the `bootstrap` prefix; see [R] **bootstrap**.

`pweights` and `fweights` are allowed; see **[U] 11.1.6 weight**. However, no weights are allowed if either option `reml` or option `dfmethod()` is specified.

`small` and `coeflegend` do not appear in the dialog box.

See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any of or all the random-effects equations.

`covariance(`*vartype*`)` specifies the structure of the covariance matrix for the random effects and may be specified for each random-effects equation. *vartype* is one of the following: `independent`, `exchangeable`, `identity`, or `unstructured`.

`independent` allows for a distinct variance for each random effect within a random-effects equation and assumes that all covariances are 0.

`exchangeable` specifies one common variance for all random effects and one common pairwise covariance.

`identity` is short for "multiple of the identity"; that is, all variances are equal and all covariances are 0.

`unstructured` allows for all variances and covariances to be distinct. If an equation consists of p random-effects terms, the unstructured covariance matrix will have $p(p+1)/2$ unique parameters.

`covariance(independent)` is the default, except when the `R.` notation is used, in which case `covariance(identity)` is the default and only `covariance(identity)` and `covariance(exchangeable)` are allowed.

`fweight(`*exp*`)` specifies frequency weights at higher levels in a multilevel model, whereas frequency weights at the first level (the observation level) are specified in the usual manner, for example, `[fw=`*fwtvar1*`]`. *exp* can be any valid Stata variable, and you can specify `fweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mixed fixed_portion [fw = wt1] || school: ..., fweight(wt2) ...
```

the variable `wt1` would hold the first-level (the observation-level) frequency weights, and `wt2` would hold the second-level (the school-level) frequency weights.

`pweight(`*exp*`)` specifies sampling weights at higher levels in a multilevel model, whereas sampling weights at the first level (the observation level) are specified in the usual manner, for example, `[pw=`*pwtvar1*`]`. *exp* can be any valid Stata variable, and you can specify `pweight()` at levels two and higher of a multilevel model. For example, in the two-level model

```
. mixed fixed_portion [pw = wt1] || school: ..., pweight(wt2) ...
```

variable `wt1` would hold the first-level (the observation-level) sampling weights, and `wt2` would hold the second-level (the school-level) sampling weights.

See Survey data in *Remarks and examples* below for more information regarding the use of sampling weights in multilevel models.

`mle` and `reml` specify the statistical method for fitting the model.

`mle`, the default, specifies that the model be fit using ML. The `dfmethod(satterthwaite)` and `dfmethod(kroger)` options are not supported under ML estimation.

`reml` specifies that the model be fit using REML, also known as residual maximum likelihood.

`dfmethod(`*df_method*`)` requests that reported hypothesis tests for the fixed effects (coefficients) use a small-sample adjustment. By default, inference is based on a large-sample approximation of the sampling distributions of the test statistics by normal and χ^2 distributions. Caution should be exercised when choosing a DF method; see Small-sample inference for fixed effects in *Remarks and examples* for details.

When `dfmethod(`*df_method*`)` is specified, the sampling distributions of the test statistics are approximated by a t distribution, according to the requested method for computing the DF. *df_method* is one of the following: `residual`, `repeated`, `anova`, `satterthwaite`, or `kroger`.

`residual` uses the residual degrees of freedom, $n - \text{rank}(X)$, as the DF for all tests of fixed effects. For a linear model without random effects with independent and identically distributed (i.i.d.) errors, the distributions of the test statistics for fixed effects are t distributions with the residual DF. For other mixed-effects models, this method typically leads to poor approximations of the actual sampling distributions of the test statistics.

`repeated` uses the repeated-measures ANOVA method for computing the DF. It is used with balanced repeated-measures designs with spherical correlation error structures. It partitions the residual degrees of freedom into the between-subject degrees of freedom and the within-subject degrees of freedom. `repeated` is supported only with two-level models. For more complex mixed-effects models or with unbalanced data, this method typically leads to poor approximations of the actual sampling distributions of the test statistics.

`anova` uses the traditional ANOVA method for computing the DF. According to this method, the DF for a test of a fixed effect of a given variable depends on whether that variable is also included in any of the random-effects equations. For traditional ANOVA models with balanced designs, this method

provides exact sampling distributions of the test statistics. For more complex mixed-effects models or with unbalanced data, this method typically leads to poor approximations of the actual sampling distributions of the test statistics.

`satterthwaite`[, *dfopts*] implements a generalization of the Satterthwaite (1946) approximation of the unknown sampling distributions of test statistics for complex linear mixed-effect models. This method is supported only with REML estimation.

`kroger`[, *dfopts*] implements the Kenward and Roger (1997) method, which is designed to approximate unknown sampling distributions of test statistics for complex linear mixed-effects models. This method is supported only with REML estimation.

dfopts is either `eim` or `oim`.

`eim` specifies that the expected information matrix be used to compute Satterthwaite or Kenward–Roger degrees of freedom. This is the default.

`oim` specifies that the observed information matrix be used to compute Satterthwaite or Kenward–Roger degrees of freedom.

Residual, repeated, and ANOVA methods are suitable only when the sampling distributions of the test statistics are known to be t or F. This is usually only known for certain classes of linear mixed-effects models with simple covariance structures and when data are balanced. These methods are available with both ML and REML estimation.

For unbalanced data or balanced data with complicated covariance structures, the sampling distributions of the test statistics are unknown and can only be approximated. The Satterthwaite and Kenward–Roger methods provide approximations to the distributions in these cases. According to Schaalje, McBride, and Fellingham (2002), the Kenward–Roger method should, in general, be preferred to the Satterthwaite method. However, there are situations in which the two methods are expected to perform similarly, such as with compound symmetry covariance structures. The Kenward–Roger method is more computationally demanding than the Satterthwaite method. Both methods are available only with REML estimation. See *Small-sample inference for fixed effects* in *Remarks and examples* for examples and more detailed descriptions of the DF methods.

`dfmethod()` may not be combined with weighted estimation, the `mi estimate` prefix, or `vce()`, unless it is the default `vce(oim)`.

`residuals(`*restype*[, *resopts*]`)` specifies the structure of the within-group errors or, loosely, the residual errors. Within-group errors are errors within the lowest-level groups (the second level of a multilevel model with the observations comprising the first level) of the linear mixed model. For example, if you are modeling random effects for classes nested within schools, then `residuals()` refers to the error variance–covariance structure of the observations within classes, the lowest-level groups. *restype* is one of the following: `independent`, `exchangeable`, `ar` [#], `ma` [#], `unstructured`, `banded` [#], `toeplitz` [#], `exponential`, `spexponential`, `sppower`, `spgaussian`, or `spspherical`.

`independent`, the default, specifies that all within-group errors be i.i.d. Gaussian with one common variance. When combined with `by(`*varname*`)`, independence is still assumed, but you estimate a distinct variance for each level of *varname*. Unlike with the structures described below, *varname* does not need to be constant within groups.

`exchangeable` estimates two parameters, one common within-group variance and one common pairwise covariance. When combined with `by(`*varname*`)`, these two parameters are distinctly estimated for each level of *varname*. Because you are modeling a within-group covariance, *varname* must be constant within lowest-level groups.

`ar` [#] assumes that within-group errors have an autoregressive (AR) structure of order #; `ar 1` is the default. The `t(`*varname*`)` option is required, where *varname* is an integer-valued time variable used to order the observations within groups and to determine the lags between successive observations. Any nonconsecutive time values will be treated as gaps. For this structure, # + 1 parameters are estimated (# AR coefficients and one overall error variance). *restype* `ar` may be combined with `by(`*varname*`)`, but *varname* must be constant within groups.

`ma` [#] assumes that within-group errors have a moving-average (MA) structure of order #; `ma 1` is the default. The `t(`*varname*`)` option is required, where *varname* is an integer-valued time variable used to order the observations within groups and to determine the lags between successive observations. Any nonconsecutive time values will be treated as gaps. For this structure, # + 1 parameters are estimated (# MA coefficients and one overall error variance). *restype* `ma` may be combined with `by(`*varname*`)`, but *varname* must be constant within groups.

`unstructured` is the most general structure; it estimates distinct variances for each within-group error and distinct covariances for each within-group error pair. The `t(`*varname*`)` option is required, where *varname* is a nonnegative-integer–valued variable that identifies the observations within each group. The groups may be unbalanced in that not all levels of `t()` need to be observed within every group, but you may not have repeated `t()` values within any particular group. When you have p levels of `t()`, then $p(p+1)/2$ parameters are estimated. *restype* `unstructured` may be combined with `by(`*varname*`)`, but *varname* must be constant within groups.

`banded` [#] is a special case of `unstructured` that restricts estimation to the covariances within the first # off-diagonals and sets the covariances outside this band to 0. The `t(`*varname*`)` option is required, where *varname* is a nonnegative-integer–valued variable that identifies the observations within each group. # is an integer between 0 and $p-1$, where p is the number of levels of `t()`. By default, # is $p-1$; that is, all elements of the covariance matrix are estimated. When # is 0, only the diagonal elements of the covariance matrix are estimated. *restype* `banded` may be combined with `by(`*varname*`)`, but *varname* must be constant within groups.

`toeplitz` [#] assumes that within-group errors have Toeplitz structure of order #, for which correlations are constant with respect to time lags less than or equal to # and are 0 for lags greater than #. The `t(`*varname*`)` option is required, where *varname* is an integer-valued time variable used to order the observations within groups and to determine the lags between successive observations. # is an integer between 1 and the maximum observed lag (the default). Any nonconsecutive time values will be treated as gaps. For this structure, #+1 parameters are estimated (# correlations and one overall error variance). *restype* `toeplitz` may be combined with `by(`*varname*`)`, but *varname* must be constant within groups.

`exponential` is a generalization of the AR covariance model that allows for unequally spaced and noninteger time values. The `t(`*varname*`)` option is required, where *varname* is real-valued. For the exponential covariance model, the correlation between two errors is the parameter ρ, raised to a power equal to the absolute value of the difference between the `t()` values for those errors. For this structure, two parameters are estimated (the correlation parameter ρ and one overall error variance). *restype* `exponential` may be combined with `by(`*varname*`)`, but *varname* must be constant within groups.

`spexponential` is part of StataNow. It specifies a spatial exponential covariance structure. It assumes that the correlation between two within-group errors decays exponentially with the distance between them, as defined by the coordinate variables specified in the `coordinates()` suboption.

Two parameters are estimated: a spatial scale parameter and one overall error variance. The scale parameter controls the range of spatial dependence, with correlation decreasing rapidly at short distances and decaying slowly at longer distances but never dropping to exactly zero.

`sppower` is part of StataNow. It specifies a spatial power covariance structure. It assumes that the correlation between two within-group errors decays according to a power law with the distance between them, as defined by the coordinate variables specified in the `coordinates()` suboption. Two parameters are estimated: a correlation parameter ρ and one overall error variance. This structure is a generalization of the `exponential` covariance structure. It is useful for modeling long-range dependence or "heavy-tailed" spatial correlations that persist across large distances.

`spgaussian` is part of StataNow. It specifies a spatial Gaussian covariance structure. It assumes that the correlation between two within-group errors decays as a Gaussian function of the distance between them, as defined by the coordinate variables specified in the `coordinates()` suboption. Two parameters are estimated: a spatial scale parameter and one overall error variance. The Gaussian structure produces very smooth spatial fields, making it appropriate for processes that vary continuously without abrupt changes.

`spspherical` is part of StataNow. It specifies a spatial spherical covariance structure. It assumes that the correlation between two within-group errors decreases as a cubic function of the distance between them, as defined by the coordinate variables specified in the `coordinates()` suboption, reaching exactly zero at the value of the range parameter. Two parameters are estimated: a range parameter and one overall error variance. This structure is appropriate for processes with a finite spatial influence.

resopts are `by(`*varname*`)`, `t(`*varname*`)`, `coordinates(`*varlist*`)`, `nugget`, `metric(`*distance*`)`, and *spinitopts*.

`by(`*varname*`)` is for use within the `residuals()` option and specifies that a set of distinct within-group error parameters be estimated for each level of *varname*. In other words, you use `by()` to model heteroskedasticity.

`t(`*varname*`)` is for use within the `residuals()` option to specify a time variable for the `ar`, `ma`, `toeplitz`, and `exponential` structures, or to identify the observations when *restype* is `unstructured` or `banded`.

`coordinates(`*varlist*`)` is part of StataNow. It is used within the `residuals()` option to specify the coordinate variables used to compute spatial distances for the spatial covariance structures `spexponential`, `sppower`, `spgaussian`, and `spspherical`.

`nugget` is part of StataNow. It is used within the `residuals()` option when *restype* is one of the spatial covariance structures `spexponential`, `sppower`, `spgaussian`, or `spspherical`. `nugget` estimates a nugget proportion η, representing the fraction of the within-group error variance attributable to measurement error or microscale spatial variation. With a nonzero nugget proportion, the correlation at zero distance is $(1 - \eta)$ instead of 1; see *Spatial residual-error structures* for details.

`metric(`*distance*`)` is part of StataNow. It is used within the `residuals()` option when *restype* is one of the spatial covariance structures `spexponential`, `sppower`, `spgaussian`, or `spspherical`. `metric()` specifies the type of distance used to compute spatial correlations. *distance* is one of `euclidean` (the default), `manhattan`, or `maximum`.

spinitopts are part of StataNow. They are initial-value options available when *restype* is one of the spatial covariance structures `spexponential`, `sppower`, `spgaussian`, or `spspherical`. These options allow you to specify starting values for the spatial correlation parameters and

for the nugget proportion. *spinitopts* are `rho0()`, `range0()`, `phi0()`, `distptile0()`, and `nugget0()`. When the `by()` option is specified, *spinitopts* can be given a numlist. The length of the numlist must be either 1 or equal to the number of levels of the variable name in `by()`. If the numlist has length 1, the same starting value is used for all levels; otherwise, values are assigned in the order they appear in the numlist. In the descriptions below, D^* is the 25th percentile of all within-group pairwise distances; see *Initial values for spatial error parameters* for details.

`rho0(`#|*numlist*`)` specifies initial values for the correlation parameter ρ of the `sppower` structure. Values must be between 0 and 1. The default value is $\exp(-3/D^*)$. *numlist* may be specified only when the `by()` option is used.

`range0(`#|*numlist*`)` specifies initial values for the range parameter of the `spspherical` structure. Values must be positive. The default value is D^*. *numlist* may be specified only when the `by()` option is used.

`phi0(`#|*numlist*`)` specifies initial values for the decay parameter ϕ of the `spexponential` and `spgaussian` structures. Values must be positive. The default value is $D^*/3$ for the `spexponential` structure and $D^*/\sqrt{3}$ for the `spgaussian` structure. *numlist* may be specified only when the `by()` option is used.

`distptile0(`#|*numlist*`)` specifies that the initial value for the spatial correlation parameter (`rho`, `range`, or `phi`) be determined from the #th percentile of all within-group pairwise distances, where # must be between 0 and 100. The default value is D^*, `distptile0(25)`. `distptile0()` may not be combined with `rho0()`, `range0()`, or `phi0()`. *numlist* may be specified only when the `by()` option is used.

`nugget0(`#|*numlist*`)` specifies initial values for the nugget proportion when the nugget effect is estimated. Values must be between 0 and 1. The default value is `nugget0(0.1)`. *numlist* may be specified only when the `by()` option is used. `nugget0()` implies the `nugget` suboption.

`pwscale(`*scale_method*`)` controls how sampling weights (if specified) are scaled in two-level models. *scale_method* is one of the following: `size`, `effective`, or `gk`.

`size` specifies that first-level (observation-level) weights be scaled so that they sum to the sample size of their corresponding second-level cluster. Second-level sampling weights are left unchanged.

`effective` specifies that first-level weights be scaled so that they sum to the effective sample size of their corresponding second-level cluster. Second-level sampling weights are left unchanged.

`gk` specifies the Graubard and Korn (1996) method. Under this method, second-level weights are set to the cluster averages of the products of the weights at both levels, and first-level weights are then set equal to 1.

`pwscale()` is supported only with two-level models. See *Survey data* in *Remarks and examples* below for more details on using `pwscale()`. `pwscale()` may not be combined with the `dfmethod()` option.

`constraints(`*constraints*`)`; see [R] **Estimation options**.

SE/Robust

vce(*vcetype*) specifies the type of standard error reported, which includes types that are derived from asymptotic theory (oim), that are robust to some kinds of misspecification (robust), and that allow for intragroup correlation (cluster *clustvar*); see [R] ***vce_option***. If vce(robust) is specified, robust variances are clustered at the highest level in the multilevel model.

vce(robust) and vce(cluster *clustvar*) are not supported with REML estimation. Only vce(oim) is allowed in combination with dfmethod().

Reporting

level(#); see [R] **Estimation options**.

variance, the default, displays the random-effects and within-group error parameter estimates as variances and covariances.

stddeviations displays the random-effects and within-group error parameter estimates as standard deviations and correlations.

dftable(*dftable*) specifies the contents of the fixed-effects table for small-sample inference when dfmethod() is used during estimation. *dftable* is one of the following: default, ci, or pvalue.

default displays the default standard fixed-effects table that contains test statistics, p-values, and confidence intervals.

ci displays the fixed-effects table in which the columns containing statistics and p-values are replaced with a column containing coefficient-specific DFs. Confidence intervals are also displayed.

pvalue displays the fixed-effects table that includes a column containing DFs with the standard columns containing test statistics and p-values. Confidence intervals are not displayed.

noretable suppresses the random-effects table from the output.

nofetable suppresses the fixed-effects table from the output.

estmetric displays all parameter estimates in one table using the metric in which they are stored in e(b). The results are stored in the same metric regardless of the parameterization of the variance components, matsqrt or matlog, used at estimation time. Random-effects parameter estimates are stored as log standard-deviations and hyperbolic arctangents of correlations, with equation names that organize them by model level. Residual-variance parameter estimates are stored as log standard-deviations and, when applicable, as hyperbolic arctangents of correlations. Note that fixed-effects estimates are always stored and displayed in the same metric.

noheader suppresses the output header, either at estimation or upon replay.

nogroup suppresses the display of group summary information (number of groups, average group size, minimum, and maximum) from the output header.

nostderr prevents mixed from calculating standard errors for the estimated random-effects parameters, although standard errors are still provided for the fixed-effects parameters. Specifying this option will speed up computation times. nostderr is available only when within-group errors are modeled as independent with constant variance.

nocnsreport; see [R] **Estimation options**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

EM options

These options control the expectation-maximization (EM) iterations that take place before estimation switches to a gradient-based method. When within-group errors are modeled as independent with constant variance, EM will either converge to the solution or bring parameter estimates close to the solution. For other within-group error covariance structures or for weighted estimation, EM is used to obtain starting values.

`emiterate(#)` specifies the number of EM iterations to perform. The default is `emiterate(20)`.

`emtolerance(#)` specifies the convergence tolerance for the EM algorithm. The default is `emtolerance(1e-10)`. EM iterations will be halted once the log (restricted) likelihood changes by a relative amount less than #. At that point, optimization switches to a gradient-based method, unless `emonly` is specified, in which case maximization stops.

`emonly` specifies that the likelihood be maximized exclusively using EM. The advantage of specifying `emonly` is that EM iterations are typically much faster than those for gradient-based methods. The disadvantages are that EM iterations can be slow to converge (if at all) and that EM provides no facility for estimating standard errors for the random-effects parameters. `emonly` is available only with unweighted estimation and when within-group errors are modeled as independent with constant variance.

`emlog` specifies that the EM iteration log be shown. The EM iteration log is, by default, not displayed unless the `emonly` option is specified.

`emdots` specifies that the EM iterations be shown as dots. This option can be convenient because the EM algorithm may require many iterations to converge.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, `[no]log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, and `nonrtolerance`; see [R] **Maximize**. Those that require special mention for `mixed` are listed below.

For the `technique()` option, the default is `technique(nr)`. The `bhhh` algorithm may not be specified.

`matsqrt` (the default), during optimization, parameterizes variance components by using the matrix square roots of the variance–covariance matrices formed by these components at each model level.

`matlog`, during optimization, parameterizes variance components by using the matrix logarithms of the variance–covariance matrices formed by these components at each model level.

The `matsqrt` parameterization ensures that variance–covariance matrices are positive semidefinite, while `matlog` ensures matrices that are positive definite. For most problems, the matrix square root is more stable near the boundary of the parameter space. However, if convergence is problematic, one option may be to try the alternate `matlog` parameterization. When convergence is not an issue, both parameterizations yield equivalent results.

The following options are available with `mixed` but are not shown in the dialog box:

`small` replays previously obtained small-sample results. This option is available only upon replay and requires that the `dfmethod()` option be used during estimation. `small` is equivalent to `dftable(default)` upon replay.

collinear specifies that mixed not omit collinear variables from the random-effects equation. Usually, there is no reason to leave collinear variables in place; in fact, doing so usually causes the estimation to fail because of the matrix singularity caused by the collinearity. However, with certain models (for example, a random-effects model with a full set of contrasts), the variables may be collinear, yet the model is fully identified because of restrictions on the random-effects covariance structure. In such cases, using the collinear option allows the estimation to take place with the random-effects equation intact.

coeflegend; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

Introduction
Two-level models
Covariance structures
Likelihood versus restricted likelihood
Three-level models
Blocked-diagonal covariance structures
Heteroskedastic random effects
Heteroskedastic residual errors
Other residual-error structures
Serial residual-error structures
Spatial residual-error structures (StataNow)
Initial values for spatial error parameters (StataNow)
Crossed-effects models
Diagnosing convergence problems
Survey data
Small-sample inference for fixed effects

Introduction

Linear mixed models are models containing both fixed effects and random effects. They are a generalization of linear regression allowing for the inclusion of random deviations (effects) other than those associated with the overall error term. In matrix notation,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon} \tag{1}$$

where $\mathbf{y}$ is the $n \times 1$ vector of responses, $\mathbf{X}$ is an $n \times p$ design/covariate matrix for the fixed effects $\boldsymbol{\beta}$, and $\mathbf{Z}$ is the $n \times q$ design/covariate matrix for the random effects $\mathbf{u}$. The $n \times 1$ vector of errors $\boldsymbol{\epsilon}$ is assumed to be multivariate normal with mean 0 and variance matrix $\sigma^2_\epsilon\mathbf{R}$.

The fixed portion of (1), $\mathbf{X}\boldsymbol{\beta}$, is analogous to the linear predictor from a standard OLS regression model with $\boldsymbol{\beta}$ being the regression coefficients to be estimated. For the random portion of (1), $\mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}$, we assume that $\mathbf{u}$ has variance–covariance matrix $\mathbf{G}$ and that $\mathbf{u}$ is orthogonal to $\boldsymbol{\epsilon}$ so that

$$\text{Var}\begin{bmatrix}\mathbf{u}\\ \boldsymbol{\epsilon}\end{bmatrix} = \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ \mathbf{0} & \sigma^2_\epsilon\mathbf{R}\end{bmatrix}$$

The random effects $\mathbf{u}$ are not directly estimated (although they may be predicted), but instead are characterized by the elements of $\mathbf{G}$, known as variance components, that are estimated along with the overall error variance σ^2_ϵ and other within-group error parameters that are contained within $\mathbf{R}$.

The general forms of the design matrices $\mathbf{X}$ and $\mathbf{Z}$ allow estimation for a broad class of linear models: blocked designs, split-plot designs, growth curves, multilevel or hierarchical designs, etc. They also allow a flexible method of modeling within-cluster correlation. Subjects within the same cluster can be correlated as a result of a shared random intercept, or through a shared random slope on (say) age, or both. The general specification of $\mathbf{G}$ also provides additional flexibility—the random intercept and random slope could themselves be modeled as independent, or correlated, or independent with equal variances, and so forth. The general structure of $\mathbf{R}$ also allows for within-group errors to be heteroskedastic and correlated, and allows flexibility in exactly how these characteristics can be modeled.

Comprehensive treatments of mixed models are provided by, among others, Searle, Casella, and McCulloch (1992); McCulloch, Searle, and Neuhaus (2008); Verbeke and Molenberghs (2000); Raudenbush and Bryk (2002); and Pinheiro and Bates (2000). In particular, chapter 2 of Searle, Casella, and McCulloch (1992) provides an excellent history.

The key to fitting mixed models lies in estimating the variance components, and for that there exist many methods. Most of the early literature in mixed models dealt with estimating variance components in ANOVA models. For simple models with balanced data, estimating variance components amounts to solving a system of equations obtained by setting expected mean-squares expressions equal to their observed counterparts. Much of the work in extending the ANOVA method to unbalanced data for general ANOVA designs is due to Henderson (1953).

The ANOVA method, however, has its shortcomings. Among these is a lack of uniqueness in that alternative, unbiased estimates of variance components could be derived using other quadratic forms of the data in place of observed mean squares (Searle, Casella, and McCulloch 1992, 38–39). As a result, ANOVA methods gave way to more modern methods, such as minimum norm quadratic unbiased estimation (MINQUE) and minimum variance quadratic unbiased estimation (MIVQUE); see Rao (1973) for MINQUE and LaMotte (1973) for MIVQUE. Both methods involve finding optimal quadratic forms of the data that are unbiased for the variance components.

The most popular methods, however, are ML and REML, and these are the two methods that are supported by `mixed`. The ML estimates are based on the usual application of likelihood theory, given the distributional assumptions of the model. The basic idea behind REML (Thompson 1962) is that you can form a set of linear contrasts of the response that do not depend on the fixed effects $\boldsymbol{\beta}$, but instead depend only on the variance components to be estimated. You then apply ML methods by using the distribution of the linear contrasts to form the likelihood.

Returning to (1): in clustered-data situations, it is convenient not to consider all n observations at once but instead to organize the mixed model as a series of M independent groups or clusters

$$\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j + \boldsymbol{\epsilon}_j \tag{2}$$

for $j = 1, \ldots, M$, with cluster j consisting of n_j observations. The response $\mathbf{y}_j$ comprises the rows of $\mathbf{y}$ corresponding with the jth cluster, with $\mathbf{X}_j$ and $\boldsymbol{\epsilon}_j$ defined analogously. The random effects $\mathbf{u}_j$ can now be thought of as M realizations of a $q \times 1$ vector that is normally distributed with mean $\mathbf{0}$ and $q \times q$ variance matrix $\boldsymbol{\Sigma}$. The matrix $\mathbf{Z}_i$ is the $n_j \times q$ design matrix for the jth cluster random effects. Relating this to (1), note that

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Z}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}_M \end{bmatrix}; \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_M \end{bmatrix}; \quad \mathbf{G} = \mathbf{I}_M \otimes \boldsymbol{\Sigma}; \quad \mathbf{R} = \mathbf{I}_M \otimes \boldsymbol{\Lambda} \tag{3}$$

The mixed-model formulation (2) is from Laird and Ware (1982) and offers two key advantages. First, it makes specifications of random-effects terms easier. If the clusters are schools, you can simply specify a random effect at the school level, as opposed to thinking of what a school-level random effect would mean when all the data are considered as a whole (if it helps, think Kronecker products). Second, representing a mixed-model with (2) generalizes easily to more than one set of random effects. For example, if classes are nested within schools, then (2) can be generalized to allow random effects at both the school and the class-within-school levels. This we demonstrate later.

In the sections that follow, we assume that within-group errors are independent with constant variance; that is, in (3) we treat $\mathbf{\Lambda}$ equal to the identity matrix and limit ourselves to estimating one overall error variance, σ^2_ϵ. Beginning in *Heteroskedastic residual errors*, we relax this assumption.

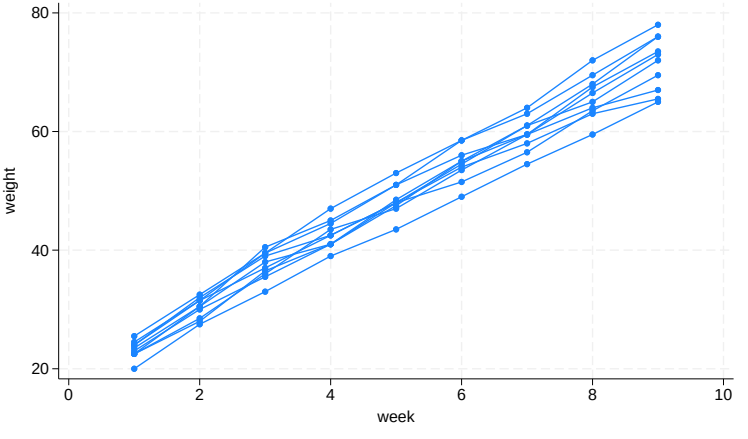
Two-level models

We begin with a simple application of (2) as a two-level model, because a one-level linear model, by our terminology, is just standard OLS regression.

▷ Example 1: Two-level random intercept model

Consider a longitudinal dataset, used by both Ruppert, Wand, and Carroll (2003) and Diggle et al. (2002), consisting of `weight` measurements of 48 pigs on 9 successive `weeks`. Pigs are identified by the variable `id`. Below is a plot of the growth curves for the first 10 pigs.

```
. use https://www.stata-press.com/data/r19/pig
(Longitudinal analysis of pig weights)
. twoway connected weight week if id<=10, connect(L)
```



It seems clear that each pig experiences a linear trend in growth and that overall weight measurements vary from pig to pig. Because we are not really interested in these particular 48 pigs per se, we instead treat them as a random sample from a larger population and model the between-pig variability as a random effect, or in the terminology of (2), as a random-intercept term at the pig level. We thus wish to fit the model

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_j + \epsilon_{ij} \tag{4}$$

for $i = 1, \ldots, 9$ weeks and $j = 1, \ldots, 48$ pigs. The fixed portion of the model, $\beta_0 + \beta_1 \texttt{week}_{ij}$, simply states that we want one overall regression line representing the population average. The random effect u_j serves to shift this regression line up or down according to each pig. Because the random effects occur at the pig level (`id`), we fit the model by typing

```
. mixed weight week || id:
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -1014.9268
Iteration 1:  Log likelihood = -1014.9268
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =   25337.49
Log likelihood = -1014.9268                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0390124	159.18	0.000	6.133433	6.286359
_cons	19.35561	.5974059	32.40	0.000	18.18472	20.52651

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(_cons)	14.81751	3.124225	9.801716	22.40002
var(Residual)	4.383264	.3163348	3.805112	5.04926

```
LR test vs. linear model: chibar2(01) = 472.65        Prob >= chibar2 = 0.0000
```

Notes:

1. By typing `weight week`, we specified the response, `weight`, and the fixed portion of the model in the same way that we would if we were using `regress` or any other estimation command. Our fixed effects are a coefficient on `week` and a constant term.
2. When we added `|| id:`, we specified random effects at the level identified by the group variable `id`, that is, the pig level (level two). Because we wanted only a random intercept, that is all we had to type.
3. The estimation log consists of three parts:
   a. A set of EM iterations used to refine starting values. By default, the iterations themselves are not displayed, but you can display them with the `emlog` option.
   b. A set of gradient-based iterations. By default, these are Newton–Raphson iterations, but other methods are available by specifying the appropriate *maximize_options*; see [R] **Maximize**.
   c. The message "Computing standard errors". This is just to inform you that `mixed` has finished its iterative maximization and is now reparameterizing from a matrix-based parameterization (see *Methods and formulas*) to the natural metric of variance components and their estimated standard errors.

4. The output title, "Mixed-effects ML regression", informs us that our model was fit using ML, the default. For REML estimates, use the `reml` option.

   Because this model is a simple random-intercept model fit by ML, it would be equivalent to using `xtreg` with its `mle` option.

5. The first estimation table reports the fixed effects. We estimate $\beta_0 = 19.36$ and $\beta_1 = 6.21$.

6. The second estimation table shows the estimated variance components. The first section of the table is labeled `id: Identity`, meaning that these are random effects at the `id` (pig) level and that their variance–covariance matrix is a multiple of the identity matrix; that is, $\mathbf{\Sigma} = \sigma_u^2\mathbf{I}$. Because we have only one random effect at this level, `mixed` knew that `Identity` is the only possible covariance structure. In any case, the variance of the level-two errors, σ_u^2, is estimated as 14.82 with standard error 3.12.

7. The row labeled `var(Residual)` displays the estimated variance of the overall error term; that is, $\hat{\sigma}_\epsilon^2 = 4.38$. This is the estimated variance of the level-one errors, which are loosely labeled as `Residual` in the output.

8. Finally, a likelihood-ratio test comparing the model with one-level ordinary linear regression, model (4) without u_j, is provided and is highly significant for these data.

   We now store our estimates for later use:

```
. estimates store randint
```

◁

▷ Example 2: Two-level random slope model

Extending (4) to allow for a random slope on `week` yields the model

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_{0j} + u_{1j}\texttt{week}_{ij} + \epsilon_{ij} \tag{5}$$

and we fit this with `mixed`:

```
. mixed weight week || id: week
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -869.03825
Iteration 1:  Log likelihood = -869.03825
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =    4689.51
Log likelihood = -869.03825                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0906819	68.48	0.000	6.032163	6.387629
_cons	19.35561	.3979159	48.64	0.000	18.57571	20.13551

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Independent				
var(week)	.3680668	.0801181	.2402389	.5639103
var(_cons)	6.756364	1.543503	4.317721	10.57235
var(Residual)	1.598811	.1233988	1.374359	1.85992

```
LR test vs. linear model: chi2(2) = 764.42                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
. estimates store randslope
```

Because we did not specify a covariance structure for the random effects $(u_{0j}, u_{1j})'$, `mixed` used the default `Independent` structure; that is,

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} = \begin{bmatrix} \sigma^2_{u0} & 0 \\ 0 & \sigma^2_{u1} \end{bmatrix} \tag{6}$$

with $\hat{\sigma}^2_{u0} = 6.76$ and $\hat{\sigma}^2_{u1} = 0.37$. Our point estimates of the fixed effects are essentially identical to those from model (4), but note that this does not hold generally. Given the 95% confidence interval for $\hat{\sigma}^2_{u1}$, it would seem that the random slope is significant, and we can use `lrtest` and our two stored estimation results to verify this fact:

```
. lrtest randslope randint
Likelihood-ratio test
Assumption: randint nested within randslope
 LR chi2(1) = 291.78
Prob > chi2 = 0.0000
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

The near-zero significance level favors the model that allows for a random pig-specific regression line over the model that allows only for a pig-specific shift.

◁

Covariance structures

In example 2, we fit a model with the default `Independent` covariance given in (6). Within any random-effects level specification, we can override this default by specifying an alternative covariance structure via the `covariance()` option.

▷ Example 3: Two-level model with correlated random effects

We generalize (6) to allow u_{0j} and u_{1j} to be correlated; that is,

$$\boldsymbol{\Sigma} = \text{Var}\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} = \begin{bmatrix} \sigma^2_{u0} & \sigma_{01} \\ \sigma_{01} & \sigma^2_{u1} \end{bmatrix}$$

```
. mixed weight week || id: week, covariance(unstructured)
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -868.96185
Iteration 1:  Log likelihood = -868.96185
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =    4649.17
Log likelihood = -868.96185                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0910745	68.18	0.000	6.031393	6.388399
_cons	19.35561	.3996387	48.43	0.000	18.57234	20.13889

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Unstructured				
var(week)	.3715251	.0812958	.2419532	.570486
var(_cons)	6.823363	1.566194	4.351297	10.69986
cov(week,_cons)	-.0984378	.2545767	-.5973991	.4005234
var(Residual)	1.596829	.123198	1.372735	1.857505

```
LR test vs. linear model: chi2(3) = 764.58                Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

But we do not find the correlation to be at all significant.

```
. lrtest . randslope

Likelihood-ratio test
Assumption: randslope nested within .

 LR chi2(1) =   0.15
Prob > chi2 = 0.6959
```

◁

Instead, we could have also specified `covariance(identity)`, restricting u_{0j} and u_{1j} to not only be independent but also to have common variance, or we could have specified `covariance(exchangeable)`, which imposes a common variance but allows for a nonzero correlation.

Likelihood versus restricted likelihood

Thus far, all our examples have used ML to estimate variance components. We could have just as easily asked for REML estimates. Refitting the model in example 2 by REML, we get

```
. mixed weight week || id: week, reml

Performing EM optimization ...

Performing gradient-based optimization:
Iteration 0:  Log restricted-likelihood = -870.51473
Iteration 1:  Log restricted-likelihood = -870.51473

Computing standard errors ...

Mixed-effects REML regression                   Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =    4592.10
Log restricted-likelihood = -870.51473          Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0916387	67.77	0.000	6.030287	6.389504
_cons	19.35561	.4021144	48.13	0.000	18.56748	20.14374

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Independent				
var(week)	.3764405	.0827027	.2447317	.5790317
var(_cons)	6.917604	1.593247	4.404624	10.86432
var(Residual)	1.598784	.1234011	1.374328	1.859898

LR test vs. linear model: chi2(2) = 765.92 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

Although ML estimators are based on the usual likelihood theory, the idea behind REML is to transform the response into a set of linear contrasts whose distribution is free of the fixed effects $\boldsymbol{\beta}$. The restricted likelihood is then formed by considering the distribution of the linear contrasts. This not only frees the maximization problem from $\boldsymbol{\beta}$ but also incorporates the degrees of freedom used to estimate $\boldsymbol{\beta}$ into the estimation of the variance components. This follows because, by necessity, the rank of the linear contrasts must be less than the number of observations.

As a simple example, consider a constant-only regression where $y_i \sim N(\mu, \sigma^2)$ for $i = 1, \ldots, n$. The ML estimate of σ^2 can be derived theoretically as the n-divided sample variance. The REML estimate can be derived by considering the first $n-1$ error contrasts, $y_i - \overline{y}$, whose joint distribution is free of μ. Applying maximum likelihood to this distribution results in an estimate of σ^2, that is, the $(n-1)$-divided sample variance, which is unbiased for σ^2.

The unbiasedness property of REML extends to all mixed models when the data are balanced, and thus REML would seem the clear choice in balanced-data problems, although in large samples the difference between ML and REML is negligible. One disadvantage of REML is that likelihood-ratio (LR) tests based on REML are inappropriate for comparing models with different fixed-effects specifications. ML is appropriate for such LR tests and has the advantage of being easy to explain and being the method of choice for other estimators.

Another factor to consider is that ML estimation under `mixed` is more feature-rich, allowing for weighted estimation and robust variance–covariance matrices, features not supported under REML. In the end, which method to use should be based both on your needs and on personal taste.

Examining the REML output, we find that the estimates of the variance components are slightly larger than the ML estimates. This is typical, because ML estimates, which do not incorporate the degrees of freedom used to estimate the fixed effects, tend to be biased downward.

Three-level models

The clustered-data representation of the mixed model given in (2) can be extended to two nested levels of clustering, creating a three-level model once the observations are considered. Formally,

$$\mathbf{y}_{jk} = \mathbf{X}_{jk}\boldsymbol{\beta} + \mathbf{Z}^{(3)}_{jk}\mathbf{u}^{(3)}_{k} + \mathbf{Z}^{(2)}_{jk}\mathbf{u}^{(2)}_{jk} + \boldsymbol{\epsilon}_{jk} \tag{7}$$

for $i = 1, \ldots, n_{jk}$ first-level observations nested within $j = 1, \ldots, M_k$ second-level groups, which are nested within $k = 1, \ldots, M$ third-level groups. Group j, k consists of n_{jk} observations, so $\mathbf{y}_{jk}$, $\mathbf{X}_{jk}$, and $\boldsymbol{\epsilon}_{jk}$ each have row dimension n_{jk}. $\mathbf{Z}^{(3)}_{jk}$ is the $n_{jk} \times q_3$ design matrix for the third-level random effects $\mathbf{u}^{(3)}_{k}$, and $\mathbf{Z}^{(2)}_{jk}$ is the $n_{jk} \times q_2$ design matrix for the second-level random effects $\mathbf{u}^{(2)}_{jk}$. Furthermore, assume that

$$\mathbf{u}_k^{(3)} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_3); \quad \mathbf{u}_{jk}^{(2)} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_2); \quad \boldsymbol{\epsilon}_{jk} \sim N(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I})$$

and that $\mathbf{u}_k^{(3)}$, $\mathbf{u}_{jk}^{(2)}$, and $\boldsymbol{\epsilon}_{jk}$ are independent.

Fitting a three-level model requires you to specify two random-effects equations: one for level three and then one for level two. The variable list for the first equation represents $\mathbf{Z}_{jk}^{(3)}$ and for the second equation represents $\mathbf{Z}_{jk}^{(2)}$; that is, you specify the levels top to bottom in `mixed`.

▷ Example 4: Three-level model with random intercepts

Baltagi, Song, and Jung (2001) estimate a Cobb–Douglas production function examining the productivity of public capital in each state's private output. Originally provided by Munnell (1990), the data were recorded over 1970–1986 for 48 states grouped into nine regions.

```
. use https://www.stata-press.com/data/r19/productivity
(Public capital productivity)
. describe
Contains data from https://www.stata-press.com/data/r19/productivity.dta
 Observations:           816                  Public capital productivity
    Variables:            11                  29 Mar 2024 10:57
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
state	byte	%9.0g		States 1-48
region	byte	%9.0g		Regions 1-9
year	int	%9.0g		Years 1970-1986
public	float	%9.0g		Public capital stock
hwy	float	%9.0g		log(highway component of public)
water	float	%9.0g		log(water component of public)
other	float	%9.0g		log(bldg/other component of public)
private	float	%9.0g		log(private capital stock)
gsp	float	%9.0g		log(gross state product)
emp	float	%9.0g		log(nonagriculture payrolls)
unemp	float	%9.0g		State unemployment rate

```
Sorted by:
```

Because the states are nested within regions, we fit a three-level mixed model with random intercepts at both the region and the state-within-region levels. That is, we use (7) with both $\mathbf{Z}_{jk}^{(3)}$ and $\mathbf{Z}_{jk}^{(2)}$ set to the $n_{jk} \times 1$ column of ones, and $\boldsymbol{\Sigma}_3 = \sigma_3^2$ and $\boldsymbol{\Sigma}_2 = \sigma_2^2$ are both scalars.

```
. mixed gsp private emp hwy water other unemp || region: || state:
  (output omitted)
Mixed-effects ML regression                     Number of obs =      816
        Grouping information
 ------------------------------------------------------------------------
  Group variable |     No. of       Observations per group
                 |     groups    Minimum    Average    Maximum
 ----------------+-------------------------------------------------------
          region |          9         51       90.7        136
           state |         48         17       17.0         17
 ------------------------------------------------------------------------

                                                Wald chi2(6)  = 18829.06
Log likelihood =  1430.5017                     Prob > chi2   =   0.0000
------------------------------------------------------------------------------
         gsp | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
     private |   .2671484   .0212591    12.57   0.000     .2254814    .3088154
         emp |    .754072   .0261868    28.80   0.000     .7027468    .8053973
         hwy |   .0709767    .023041     3.08   0.002     .0258172    .1161363
       water |   .0761187   .0139248     5.47   0.000     .0488266    .1034109
       other |  -.0999955   .0169366    -5.90   0.000    -.1331906   -.0668004
       unemp |  -.0058983   .0009031    -6.53   0.000    -.0076684   -.0041282
       _cons |   2.128823   .1543854    13.79   0.000     1.826233    2.431413
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects parameters |   Estimate   Std. err.      [95% conf. interval]
----------------------------+-------------------------------------------------
region: Identity            |
                  var(_cons) |  .0014506   .0012995      .0002506    .0083957
----------------------------+-------------------------------------------------
state: Identity             |
                  var(_cons) |  .0062757   .0014871      .0039442    .0099855
----------------------------+-------------------------------------------------
               var(Residual) |  .0013461   .0000689      .0012176    .0014882
------------------------------------------------------------------------------
LR test vs. linear model: chi2(2) = 1154.73                   Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Notes:

1. Our model now has two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `region` level (level three), and the second is a random intercept at the `state` level (level two). The order in which these are specified (from left to right) is significant—`mixed` assumes that `state` is nested within `region`.
2. The information on groups is now displayed as a table, with one row for each grouping. You can suppress this table with the `nogroup` or the `noheader` option, which will suppress the rest of the header, as well.
3. The variance-component estimates are now organized and labeled according to level.

After adjusting for the nested-level error structure, we find that the highway and water components of public capital had significant positive effects on private output, whereas the other public buildings component had a negative effect.

◁

❑ Technical note

In the previous example, the states are coded 1–48 and are nested within nine regions. `mixed` treated the states as nested within regions, regardless of whether the codes for each state were unique between regions. That is, even if codes for states were duplicated between regions, `mixed` would have enforced the nesting and produced the same results.

The group information at the top of the `mixed` output and that produced by the `estat group` postestimation command (see [ME] **estat group**) take the nesting into account. The statistics are thus not necessarily what you would get if you instead `tabulated` each group variable individually.

❑

Model (7) extends in a straightforward manner to more than three levels, as does the specification of such models in `mixed`.

Blocked-diagonal covariance structures

Covariance matrices of random effects within an equation can be modeled either as a multiple of the identity matrix, as diagonal (that is, `Independent`), as exchangeable, or as general symmetric (`Unstructured`). These may also be combined to produce more complex block-diagonal covariance structures, effectively placing constraints on the variance components.

▷ Example 5: Using repeated levels to induce blocked-diagonal covariance structures

Returning to our productivity data, we now add random coefficients on `hwy` and `unemp` at the `region` level. This only slightly changes the estimates of the fixed effects, so we focus our attention on the variance components:

```
. mixed gsp private emp hwy water other unemp || region: hwy unemp || state:,
> nolog nogroup nofetable
Mixed-effects ML regression                     Number of obs =      816
                                                Wald chi2(6)  = 17137.94
Log likelihood =  1447.6787                     Prob > chi2   =   0.0000
```

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
region: Independent				
var(hwy)	.0000209	.0001103	6.71e-10	.6507106
var(unemp)	.0000238	.0000135	7.84e-06	.0000722
var(_cons)	.0030349	.0086684	.0000112	.8191376
state: Identity				
var(_cons)	.0063658	.0015611	.0039365	.0102943
var(Residual)	.0012469	.0000643	.001127	.0013795

```
LR test vs. linear model: chi2(4) = 1189.08                   Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

. estimates store prodrc
```

This model is the same as that fit in example 4 except that $\mathbf{Z}_{jk}^{(3)}$ is now the $n_{jk} \times 3$ matrix with columns determined by the values of hwy, unemp, and an intercept term (one), in that order, and (because we used the default Independent structure) $\mathbf{\Sigma}_3$ is

$$\mathbf{\Sigma}_3 = \begin{array}{c} \begin{array}{ccc} \texttt{hwy} & \texttt{unemp} & \texttt{_cons} \end{array} \\ \begin{pmatrix} \sigma_a^2 & 0 & 0 \\ 0 & \sigma_b^2 & 0 \\ 0 & 0 & \sigma_c^2 \end{pmatrix} \end{array}$$

The random-effects specification at the state level remains unchanged; that is, $\mathbf{\Sigma}_2$ is still treated as the scalar variance of the random intercepts at the state level.

An LR test comparing this model with that from example 4 favors the inclusion of the two random coefficients, a fact we leave to the interested reader to verify.

The estimated variance components, upon examination, reveal that the variances of the random coefficients on hwy and unemp could be treated as equal. That is,

$$\mathbf{\Sigma}_3 = \begin{array}{c} \begin{array}{ccc} \texttt{hwy} & \texttt{unemp} & \texttt{_cons} \end{array} \\ \begin{pmatrix} \sigma_a^2 & 0 & 0 \\ 0 & \sigma_a^2 & 0 \\ 0 & 0 & \sigma_c^2 \end{pmatrix} \end{array}$$

looks plausible. We can impose this equality constraint by treating $\mathbf{\Sigma}_3$ as block diagonal: the first block is a 2×2 multiple of the identity matrix, that is, $\sigma_a^2 \mathbf{I}_2$; the second is a scalar, equivalently, a 1×1 multiple of the identity.

We construct block-diagonal covariances by repeating level specifications:

```
. mixed gsp private emp hwy water other unemp || region: hwy unemp,
> cov(identity) || region: || state:, nolog nogroup nofetable
Mixed-effects ML regression                     Number of obs =      816
                                                Wald chi2(6)  = 17136.65
Log likelihood =  1447.6784                     Prob > chi2   =   0.0000
```

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
region: Identity				
var(hwy unemp)	.0000238	.0000134	7.89e-06	.0000719
region: Identity				
var(_cons)	.0028191	.0030429	.0003399	.023383
state: Identity				
var(_cons)	.006358	.0015309	.0039661	.0101925
var(Residual)	.0012469	.0000643	.001127	.0013795

```
LR test vs. linear model: chi2(3) = 1189.08                   Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We specified two equations for the region level: the first for the random coefficients on hwy and unemp with covariance set to Identity and the second for the random intercept _cons, whose covariance defaults to Identity because it is of dimension 1. mixed labeled the estimate of σ_a^2 as var(hwy unemp) to designate that it is common to the random coefficients on both hwy and unemp.

An LR test shows that the constrained model fits equally well.

```
. lrtest . prodrc
Likelihood-ratio test
Assumption: . nested within prodrc
 LR chi2(1) =   0.00
Prob > chi2 = 0.9784
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

◁

Because the null hypothesis for this test is one of equality ($H_0 : \sigma_a^2 = \sigma_b^2$), it is not on the boundary of the parameter space. As such, we can take the reported significance as precise rather than a conservative estimate.

You can repeat level specifications as often as you like, defining successive blocks of a block-diagonal covariance matrix. However, repeated-level equations must be listed consecutively; otherwise, `mixed` will give an error.

❑ Technical note

In the previous estimation output, there was no constant term included in the first `region` equation, even though we did not use the `noconstant` option. When you specify repeated-level equations, `mixed` knows not to put constant terms in each equation because such a model would be unidentified. By default, it places the constant in the last repeated-level equation, but you can use `noconstant` creatively to override this.

❑

Linear mixed-effects models can also be fit using `meglm` with the default `gaussian` family. `meglm` provides two more covariance structures through which you can impose constraints on variance components; see [ME] **meglm** for details.

Heteroskedastic random effects

Blocked-diagonal covariance structures and repeated-level specifications of random effects can also be used to model heteroskedasticity among random effects at a given level.

▷ Example 6: Using repeated levels to model heteroskedasticity

Following Rabe-Hesketh and Skrondal (2022, sec. 7.2), we analyze data from Asian children in a British community who were weighed up to four times, roughly between the ages of 6 weeks and 27 months. The dataset is a random sample of data previously analyzed by Goldstein (1986) and Prosser, Rasbash, and Goldstein (1991).

```
. use https://www.stata-press.com/data/r19/childweight
(Weight data on Asian children)

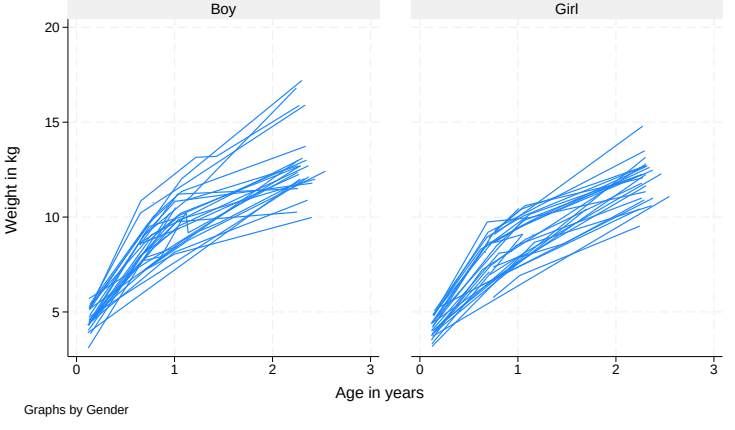
. describe

Contains data from https://www.stata-press.com/data/r19/childweight.dta
 Observations:           198                  Weight data on Asian children
    Variables:             5                  23 May 2024 15:12
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
id	int	%8.0g		Child identifier
age	float	%8.0g		Age in years
weight	float	%8.0g		Weight in Kg
brthwt	int	%8.0g		Birthweight in g
girl	byte	%9.0g	bg	Gender

```
Sorted by: id  age

. graph twoway (line weight age, connect(ascending)), by(girl)
> xtitle(Age in years) ytitle(Weight in kg)
```



Ignoring gender effects for the moment, we begin with the following model for the ith measurement on the jth child:

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{age}_{ij} + \beta_2 \texttt{age}_{ij}^2 + u_{j0} + u_{j1} \texttt{age}_{ij} + \epsilon_{ij}$$

This models overall mean growth as quadratic in age and allows for two child-specific random effects: a random intercept u_{j0}, which represents each child's vertical shift from the overall mean (β_0), and a random age slope u_{j1}, which represents each child's deviation in linear growth rate from the overall mean linear growth rate (β_1). For simplicity, we do not consider child-specific changes in the quadratic component of growth.

```
. mixed weight age c.age#c.age || id: age, nolog
Mixed-effects ML regression                     Number of obs     =       198
Group variable: id                              Number of groups  =        68
                                                Obs per group:
                                                              min =         1
                                                              avg =       2.9
                                                              max =         5
                                                Wald chi2(2)      =   1863.46
Log likelihood = -258.51915                     Prob > chi2       =    0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
age	7.693701	.2381076	32.31	0.000	7.227019	8.160384
c.age#c.age	-1.654542	.0874987	-18.91	0.000	-1.826037	-1.483048
_cons	3.497628	.1416914	24.68	0.000	3.219918	3.775338

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Independent				
var(age)	.2987207	.0827569	.1735603	.5141388
var(_cons)	.5023857	.141263	.2895294	.8717297
var(Residual)	.3092897	.0474887	.2289133	.417888

```
LR test vs. linear model: chi2(2) = 114.70                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

◁

Because there is no reason to believe that the random effects are uncorrelated, it is always a good idea to first fit a model with the `covariance(unstructured)` option. We do not include the output for such a model because for these data the correlation between random effects is not significant; however, we did check this before reverting to `mixed`'s default `Independent` structure.

Next we introduce gender effects into the fixed portion of the model by including a main gender effect and a gender–age interaction for overall mean growth. We specify `ibn.girl` and the `noconstant` option to omit the constant and estimate separate intercepts for boys and girls. The `nofvlabel` option requests that the values of the `girl` variable instead of value labels be shown in the results.

```
. mixed weight ibn.girl i.girl#c.age c.age#c.age, noconstant nofvlabel
> || id: age, nolog
Mixed-effects ML regression                     Number of obs     =        198
Group variable: id                              Number of groups  =         68
                                                Obs per group:
                                                              min =          1
                                                              avg =        2.9
                                                              max =          5
                                                Wald chi2(5)      =    6583.73
Log likelihood =   -253.182                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
girl						
0	3.754275	.1726404	21.75	0.000	3.415906	4.092644
1	3.243808	.174255	18.62	0.000	2.902274	3.585341
girl#c.age						
0	7.806765	.2524583	30.92	0.000	7.311956	8.301574
1	7.577296	.2531318	29.93	0.000	7.081166	8.073425
c.age#c.age	-1.654323	.0871752	-18.98	0.000	-1.825183	-1.483463

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Independent				
var(age)	.2772846	.0769233	.1609861	.4775987
var(_cons)	.4076892	.12386	.2247635	.7394906
var(Residual)	.3131704	.047684	.2323672	.422072

```
LR test vs. linear model: chi2(2) = 104.39                Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

. estimates store homoskedastic
```

The main gender effect is significant at the 5% level, but the gender–age interaction is not:

```
. test 0.girl#c.age = 1.girl#c.age
 ( 1)  [weight]0bn.girl#c.age - [weight]1.girl#c.age = 0

           chi2(  1) =    1.66
         Prob > chi2 =    0.1978
```

On average, boys are heavier than girls, but their average linear growth rates are not significantly different.

In the above model, we introduced a gender effect on average growth, but we still assumed that the variability in child-specific deviations from this average was the same for boys and girls. To check this assumption, we introduce gender into the random component of the model.

```
. mixed weight ibn.girl i.girl#c.age c.age#c.age, noconstant nofvlabel
> || id: ibn.girl i.girl#c.age, noconstant nolog nofetable
Mixed-effects ML regression                     Number of obs     =       198
Group variable: id                              Number of groups  =        68
                                                Obs per group:
                                                              min =         1
                                                              avg =       2.9
                                                              max =         5
                                                Wald chi2(5)      =   7319.20
Log likelihood = -248.94752                     Prob > chi2       =    0.0000
```

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Independent				
var(0.girl)	.3161091	.1557911	.1203181	.8305061
var(1.girl)	.5798676	.1959725	.2989896	1.124609
var(0.girl#age)	.4734482	.1574626	.2467028	.9085962
var(1.girl#age)	.0664634	.0553274	.0130017	.3397538
var(Residual)	.3078826	.046484	.2290188	.4139037

```
LR test vs. linear model: chi2(4) = 112.86                Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

. estimates store heteroskedastic
```

In the above, we suppress displaying the fixed portion of the model (the `nofetable` option) because it does not differ much from that of the previous model.

Our previous model had the random-effects specification

```
|| id: age
```

which we have replaced with

```
|| id: ibn.girl i.girl#c.age, noconstant
```

The former models a random intercept and random slope on age, and does so treating all children as a random sample from one population. The latter also specifies a random intercept and random slope on age, but allows for the variability of the random intercepts and slopes to differ between boys and girls. In other words, it allows for heteroskedasticity in random effects due to gender. We use the `noconstant` option so that we can separate the overall random intercept (automatically provided by the former syntax) into one specific to boys and one specific to girls.

There seems to be a large gender effect in the variability of linear growth rates. We can compare both models with an LR test, recalling that we stored the previous estimation results under the name `homoskedastic`:

```
. lrtest homoskedastic heteroskedastic

Likelihood-ratio test
Assumption: homoskedastic nested within heteroskedas~c

 LR chi2(2) =   8.47
Prob > chi2 = 0.0145

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

Because the null hypothesis here is one of equality of variances and not that variances are 0, the above does not test on the boundary; thus we can treat the significance level as precise and not conservative. Either way, the results favor the new model with heteroskedastic random effects.

Heteroskedastic residual errors

Up to this point, we have assumed that the level-one errors—the ϵ's in the stated models—have been i.i.d. Gaussian with variance σ^2_ϵ. (We use the terms "level-one errors", "within-group errors", and, less proper, "residual errors" interchangeably.) This is demonstrated in `mixed` output in the random-effects table, where up until now we have estimated one overall error variance, labeled as `var(Residual)`.

To relax the assumptions of homoskedasticity or independence of residual errors, use the `residuals()` option.

▷ Example 7: Independent residual variance structure

West, Welch, and Gałecki (2022, chap. 7) analyze data studying the effect of ceramic dental veneer placement on gingival (gum) health. Data on 55 teeth located in the maxillary arches of 12 patients were considered.

```
. use https://www.stata-press.com/data/r19/veneer, clear
(Dental veneer data)
. describe
Contains data from https://www.stata-press.com/data/r19/veneer.dta
 Observations:           110                  Dental veneer data
    Variables:             7                  24 May 2024 12:11
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
patient         byte    %8.0g                 Patient ID
tooth           byte    %8.0g                 Tooth number with patient
gcf             byte    %8.0g                 Gingival crevicular fluid (GCF)
age             byte    %8.0g                 Patient age
base_gcf        byte    %8.0g                 Baseline GCF
cda             float   %9.0g                 Average contour difference after
                                                veneer placement
followup        byte    %9.0g      t          Follow-up time: 3 or 6 months
------------------------------------------------------------------------------
Sorted by:
```

Veneers were placed to match the original contour of the tooth as closely as possible, and researchers were interested in how contour differences (variable `cda`) impacted gingival health. Gingival health was measured as the amount of gingival crevicular fluid (GCF) at each tooth, measured at baseline (variable `base_gcf`) and at two posttreatment follow-ups at 3 and 6 months. The variable `gcf` records GCF at follow-up, and the variable `followup` records the follow-up time.

Because two measurements were taken for each tooth and there exist multiple teeth per patient, we fit a three-level model with the following random effects: a random intercept and random slope on follow-up time at the patient level, and a random intercept at the tooth level. For the ith measurement of the jth tooth from the kth patient, we have

$$\begin{aligned}\texttt{gcf}_{ijk} = \beta_0 + \beta_1\texttt{followup}_{ijk} + \beta_2\texttt{base_gcf}_{ijk} + \beta_3\texttt{cda}_{ijk} + \beta_4\texttt{age}_{ijk} + \\ u_{0k} + u_{1k}\texttt{followup}_{ijk} + v_{0jk} + \epsilon_{ijk}\end{aligned}$$

which we can fit using mixed:

```
. mixed gcf followup base_gcf cda age || patient: followup, cov(un) || tooth:,
> reml nolog
Mixed-effects REML regression                   Number of obs =    110
        Grouping information
------------------------------------------------------------------------------
                |     No. of       Observations per group
 Group variable |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
        patient |         12          2        9.2         12
          tooth |         55          2        2.0          2
------------------------------------------------------------------------------
                                                Wald chi2(4)  =   7.48
Log restricted-likelihood = -420.92761          Prob > chi2   = 0.1128
------------------------------------------------------------------------------
         gcf | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
    followup |   .3009815   1.936863     0.16   0.877     -3.4952    4.097163
    base_gcf |  -.0183127   .1433094    -0.13   0.898     -.299194    .2625685
         cda |   -.329303   .5292525    -0.62   0.534    -1.366619    .7080128
         age |  -.5773932   .2139656    -2.70   0.007    -.9967582   -.1580283
       _cons |   45.73862   12.55497     3.64   0.000     21.13133    70.34591
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects parameters |   Estimate   Std. err.      [95% conf. interval]
----------------------------+-------------------------------------------------
patient: Unstructured       |
                 var(followup) |  41.88772   18.79997       17.38009    100.9535
                    var(_cons) |  524.9851   253.0205       204.1287    1350.175
            cov(followup,_cons) | -140.4229   66.57623      -270.9099   -9.935904
----------------------------+-------------------------------------------------
tooth: Identity             |
                    var(_cons) |  47.45738   16.63034        23.8792     94.3165
----------------------------+-------------------------------------------------
                 var(Residual) |  48.86704   10.50523       32.06479    74.47382
------------------------------------------------------------------------------
LR test vs. linear model: chi2(4) = 91.12                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We used REML estimation for no other reason than variety.

Among the other features of the model fit, we note that the error variance σ_ϵ^2 was estimated as 48.87 and that our model assumed that the errors were independent with constant variance (homoskedastic). Because it may be the case that the precision of gcf measurements could change over time, we modify the above to estimate two distinct error variances: one for the 3-month follow-up and one for the 6-month follow-up.

To fit this model, we add the `residuals(independent, by(followup))` option, which maintains independence of errors but allows for heteroskedasticity with respect to follow-up time.

```
. mixed gcf followup base_gcf cda age || patient: followup, cov(un) || tooth:,
> residuals(independent, by(followup)) reml nolog
Mixed-effects REML regression                   Number of obs =    110
        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
        patient |         12          2        9.2         12
          tooth |         55          2        2.0          2
------------------------------------------------------------------------------
                                                Wald chi2(4)  =   7.51
Log restricted-likelihood =  -420.4576          Prob > chi2   = 0.1113
------------------------------------------------------------------------------
         gcf | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
    followup |    .2703944   1.933096     0.14   0.889    -3.518405    4.059193
    base_gcf |    .0062144   .1419121     0.04   0.965    -.2719283     .284357
         cda |   -.2947235   .5245126    -0.56   0.574    -1.322749    .7333023
         age |   -.5743755   .2142249    -2.68   0.007    -.9942487   -.1545024
       _cons |    45.15089   12.51452     3.61   0.000     20.62288    69.6789
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects parameters |   Estimate   Std. err.      [95% conf. interval]
----------------------------+-------------------------------------------------
patient: Unstructured       |
               var(followup) |   41.75169   18.72989       17.33099    100.583
                  var(_cons) |   515.2018   251.9661       197.5542    1343.595
          cov(followup,_cons) |  -139.0496   66.27806      -268.9522   -9.146944
----------------------------+-------------------------------------------------
tooth: Identity             |
                  var(_cons) |   47.35914   16.48931       23.93514    93.70693
----------------------------+-------------------------------------------------
Residual: Independent,      |
    by followup             |
            3 months: var(e) |   61.36785   18.38913       34.10946    110.4096
            6 months: var(e) |   36.42861   14.97501       16.27542    81.53666
------------------------------------------------------------------------------
LR test vs. linear model: chi2(5) = 92.06                 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

Comparison of both models via an LR test reveals the two estimates of the error variance to be not substantively different, something we leave to you to verify as an exercise.

◁

The default residual-variance structure is `independent`, and when specified without `by()` is equivalent to the default behavior of `mixed`: estimating one overall error variance for the entire model.

Other residual-error structures

Besides the default `independent` residual-error structure, `mixed` supports other within-group error covariance structures that allow for correlation between errors within the lowest-level (smallest or level two) groups. These structures can be broadly divided into two classes: serial and spatial. Serial error covariance structures model correlation as a function of the ordering of observations within groups, typically with respect to the positional index or another indexing variable supplied in the `t()` suboption.

These structures are `exchangeable`, `ar`, `ma`, `unstructured`, `banded`, `toeplitz`, and `exponential`. Spatial error covariance structures model correlation as a function of the physical or geographic distance between observations, computed from coordinate variables specified in the `coordinates()` suboption. These structures are `spexponential`, `spgaussian`, `spspherical`, and `sppower`. Spatial structures are useful when the strength of dependence between errors depends on proximity in space instead of the observation order.

For purposes of notation, in what follows we assume a two-level model, with the obvious extension to higher-level models.

Serial residual-error structures

The `exchangeable` structure assumes one overall variance and one common pairwise covariance; that is,

$$\mathrm{Var}(\boldsymbol{\epsilon}_j) = \mathrm{Var}\begin{bmatrix}\epsilon_{j1}\\ \epsilon_{j2}\\ \vdots\\ \epsilon_{jn_j}\end{bmatrix} = \begin{bmatrix}\sigma_\epsilon^2 & \sigma_1 & \cdots & \sigma_1\\ \sigma_1 & \sigma_\epsilon^2 & \cdots & \sigma_1\\ \vdots & \vdots & \ddots & \vdots\\ \sigma_1 & \sigma_1 & \sigma_1 & \sigma_\epsilon^2\end{bmatrix}$$

By default, `mixed` will report estimates of the two parameters as estimates of the common variance σ_ϵ^2 and of the covariance σ_1. When the `by(`*varname*`)` option is also specified, these two parameters are estimated for each level *varname*.

The `ar` p structure assumes that the errors have an AR structure of order p. That is,

$$\epsilon_{ij} = \phi_1\epsilon_{i-1,j} + \cdots + \phi_p\epsilon_{i-p,j} + u_{ij}$$

where u_{ij}'s are i.i.d. Gaussian with mean 0 and variance σ_u^2. `mixed` reports estimates of $\phi_1, \ldots, \phi_p$ and the overall error variance σ_ϵ^2, which can be derived from the above expression. The `t(`*varname*`)` option is required, where *varname* is a time variable used to order the observations within lowest-level groups and to determine any gaps between observations. When the `by(`*varname*`)` option is also specified, the set of $p+1$ parameters is estimated for each level of *varname*. If $p = 1$, then the estimate of ϕ_1 is reported as `rho`, because in this case it represents the correlation between successive error terms.

The `ma` q structure assumes that the errors are an MA process of order q. That is,

$$\epsilon_{ij} = u_{ij} + \theta_1 u_{i-1,j} + \cdots + \theta_q u_{i-q,j}$$

where u_{ij}'s are i.i.d. Gaussian with mean 0 and variance σ_u^2. `mixed` reports estimates of $\theta_1, \ldots, \theta_q$ and the overall error variance σ_ϵ^2, which can be derived from the above expression. The `t(`*varname*`)` option is required, where *varname* is a time variable used to order the observations within lowest-level groups and to determine any gaps between observations. When the `by(`*varname*`)` option is also specified, the set of $q+1$ parameters is estimated for each level of *varname*.

The `unstructured` structure is the most general and estimates unique variances and unique pairwise covariances for all errors within the lowest-level grouping. Because the data may be unbalanced and the ordering of the observations is arbitrary, the `t(`*varname*`)` option is required, where *varname* is an identification variable that matches error terms in different groups. If *varname* has n distinct levels, then $n(n+1)/2$ parameters are estimated. Not all n levels need to be observed within each group, but duplicated levels of *varname* within a given group are not allowed because they would cause a singularity in the estimated error-variance matrix for that group. When the `by(`*varname*`)` option is also specified, the set of $n(n+1)/2$ parameters is estimated for each level of *varname*.

The banded q structure is a special case of unstructured that confines estimation to within the first q off-diagonal elements of the error variance–covariance matrix and sets the covariances outside this band to 0. As is the case with unstructured, the t(*varname*) option is required, where *varname* is an identification variable that matches error terms in different groups. However, with banded variance structures, the ordering of the values in *varname* is important because it determines which covariances are to be estimated and which are to be set to 0. For example, if *varname* has $n = 5$ distinct values $t = 1, 2, 3, 4, 5$, then a banded variance–covariance structure of order $q = 2$ would estimate the following:

$$\text{Var}(\boldsymbol{\epsilon}_j) = \text{Var}\begin{bmatrix}\epsilon_{1j}\\ \epsilon_{2j}\\ \epsilon_{3j}\\ \epsilon_{4j}\\ \epsilon_{5j}\end{bmatrix} = \begin{bmatrix}\sigma_1^2 & \sigma_{12} & \sigma_{13} & 0 & 0\\ \sigma_{12} & \sigma_2^2 & \sigma_{23} & \sigma_{24} & 0\\ \sigma_{13} & \sigma_{23} & \sigma_3^2 & \sigma_{34} & \sigma_{35}\\ 0 & \sigma_{24} & \sigma_{34} & \sigma_4^2 & \sigma_{45}\\ 0 & 0 & \sigma_{35} & \sigma_{45} & \sigma_5^2\end{bmatrix}$$

In other words, you would have an unstructured variance matrix that constrains $\sigma_{14} = \sigma_{15} = \sigma_{25} = 0$. If *varname* has n distinct levels, then $(q+1)(2n-q)/2$ parameters are estimated. Not all n levels need to be observed within each group, but duplicated levels of *varname* within a given group are not allowed because they would cause a singularity in the estimated error-variance matrix for that group. When the by(*varname*) option is also specified, the set of parameters is estimated for each level of *varname*. If q is left unspecified, then banded is equivalent to unstructured; that is, all variances and covariances are estimated. When $q = 0$, $\text{Var}(\boldsymbol{\epsilon}_j)$ is treated as diagonal and can thus be used to model uncorrelated yet heteroskedastic errors.

The toeplitz q structure assumes that the errors are homoskedastic and that the correlation between two errors is determined by the time lag between the two. That is, $\text{Var}(\epsilon_{ij}) = \sigma_\epsilon^2$ and

$$\text{Corr}(\epsilon_{ij}, \epsilon_{i+k,j}) = \rho_k$$

If the lag k is less than or equal to q, then the pairwise correlation ρ_k is estimated; if the lag is greater than q, then ρ_k is assumed to be 0. If q is left unspecified, then ρ_k is estimated for each observed lag k. The t(*varname*) option is required, where *varname* is a time variable t used to determine the lags between pairs of error terms. As such, t() must be integer-valued. $q + 1$ parameters are estimated: one overall variance σ_ϵ^2 and q correlations. When the by(*varname*) option is also specified, the set of $q + 1$ parameters is estimated for each level of *varname*.

The exponential structure is a generalization of the AR structure that allows for noninteger and irregularly spaced time lags. That is, $\text{Var}(\epsilon_{ij}) = \sigma_\epsilon^2$ and

$$\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) = \rho^{|i-k|}$$

for $0 \le \rho \le 1$, with i and k not required to be integers. The t(*varname*) option is required, where *varname* is a time variable used to determine i and k for each error pair. t() is real-valued. mixed reports estimates of σ_ϵ^2 and ρ. When the by(*varname*) option is also specified, these two parameters are estimated for each level of *varname*.

Spatial residual-error structures (StataNow)

In addition to the residual-error structures described above, mixed supports four spatial structures that model correlation between within-group errors as a function of their geographic or physical distance. These structures are useful for data collected at specific spatial locations, such as agricultural field trials, environmental monitoring studies, or geological surveys. They can also be useful for unequally spaced longitudinal data by treating time as a one-dimensional spatial domain. For more details, see Cressie (1993).

Let ϵ_{ij} denote the error for observation i within group j, and let d_{ijk} be the distance between observations i and k within the same group. All spatial structures assume a common error variance,

$$\text{Var}(\epsilon_{ij}) = \sigma^2_\epsilon$$

referred to as the "sill". The spatial correlation is modeled as

$$\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) = \eta\,\delta_{ik} + (1-\eta)\,h(d_{ijk};\phi)$$

where δ_{ik} is 1 when $i = k$ and 0 otherwise, ϕ is a (spatial) scale parameter, and $0 \le \eta \le 1$ is the "nugget proportion", representing the fraction of the sill attributable to measurement error or microscale spatial variation. When the `nugget` suboption is not specified, $\eta = 0$. $h(d;\phi)$ is a spatial correlation kernel with $h(0;\phi) = 1$.

The `coordinates(`*varlist*`)` suboption is required with spatial structures, where *varlist* specifies variables used to compute distances between observations within groups. Suppose `coordinates()` contains r variables, and let

$$\mathbf{s}_{ij} = (s_{ij1}, \ldots, s_{ijr})'$$

denote the coordinate vector for observation i within group j. By default, distances are computed using the Euclidean metric,

$$d_{ijk} = \left\{ \sum_{\ell=1}^{r} (s_{ij\ell} - s_{kj\ell})^2 \right\}^{1/2}$$

which may be changed using the `metric()` suboption. The Manhattan metric computes distances as

$$d_{ijk} = \sum_{\ell=1}^{r} |s_{ij\ell} - s_{kj\ell}|$$

and the maximum metric computes distances as

$$d_{ijk} = \max_{1\le\ell\le r} |s_{ij\ell} - s_{kj\ell}|$$

The `spexponential` structure assumes that the correlation between two errors decreases exponentially with the distance between them. The `spexponential` structure uses an exponential correlation kernel,

$$h_{\text{EXP}}(d;\phi) = \exp\left(-\frac{d}{\phi}\right)$$

so that

$$\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) = \eta\,\delta_{ik} + (1-\eta)\,\exp\left(-\frac{d_{ijk}}{\phi}\right)$$

The correlation decays smoothly toward zero as the distance increases but never exactly reaches zero. The parameter ϕ controls the rate of decay; larger values imply spatial correlation extends over greater distances. Observations are approximately uncorrelated (correlation ≤ 0.05) when separated by distances greater than the practical range, which is approximately 3ϕ. Parameter ϕ is labeled as `phi` in the output.

The `spgaussian` structure is similar to `spexponential` but assumes a Gaussian (squared exponential) decay in correlation. The `spgaussian` structure uses the following correlation kernel:

$$h_{\text{GAU}}(d;\phi) = \exp\left\{ -\left(\frac{d}{\phi}\right)^2 \right\}$$

Therefore,

$$\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) = \eta\,\delta_{ik} + (1-\eta)\,\exp\left(-\frac{d_{ijk}^2}{\phi^2}\right)$$

This structure implies very high correlation for observations in close proximity (the correlation function is flat near the origin) and more rapid decay beyond the scale parameter ϕ compared with the exponential structure. Observations are approximately uncorrelated when separated by distances of roughly $\sqrt{3}\phi$ or more. Parameter ϕ is labeled as `phi` in the output. The Gaussian structure is appropriate when the spatial process is believed to be highly smooth.

The `spspherical` structure assumes spherical correlation, which has a finite range beyond which the correlation is exactly zero. The `spspherical` structure uses the following correlation kernel:

$$h_{\text{SPH}}(d;\phi) = \begin{cases} 1 - \frac{3}{2}\frac{d}{\phi} + \frac{1}{2}\left(\frac{d}{\phi}\right)^3, & d \le \phi \\ 0, & d > \phi \end{cases}$$

Thus,

$$\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) = \eta\,\delta_{ik} + (1-\eta)\,h_{\text{SPH}}(d_{ijk};\phi)$$

The spherical structure has linear behavior near the origin and reaches exactly zero correlation at the scale parameter ϕ. This structure is commonly used in geostatistics and is appropriate when domain knowledge suggests spatial influence has a specific cutoff distance. Parameter ϕ is referred to as the "geostatistical range" in this case. It is labeled as `Range` in the output.

The `sppower` structure is a power-law correlation structure that generalizes the `exponential` structure. In a one-dimensional setting (when only one *varname* is specified in the `coordinates()` suboption), the two correlation structures yield identical results. The `sppower` structure uses the following correlation kernel,

$$h_{\text{POW}}(d;\rho) = \rho^d, \qquad 0 \le \rho \le 1$$

so that

$$\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) = \eta\,\delta_{ik} + (1-\eta)\,\rho^{d_{ijk}}$$

where $0 \le \rho \le 1$. Parameter ρ is labeled as `rho` in the output. The practical range (correlation ≈ 0.05) for this correlation structure is $\ln(0.05)/\ln(\rho)$.

Initial values for spatial error parameters (StataNow)

The estimation of spatial error correlation parameters is known to be sensitive to the choice of starting values. `mixed` provides several options for specifying initial values for the spatial correlation parameters via the `range0()`, `phi0()`, `rho0()`, `nugget0()`, and `distptile0()` suboptions of the `residuals()` option. Supplying reasonable initial values can substantively improve numerical stability, reduce computation time, and help avoid convergence to suboptimal local maximums.

You may specify initial values for the range parameter of the `spspherical` correlation structure directly in the `range0()` suboption. You may also specify initial values for the decay parameter ϕ of the `spexponential` and `spgaussian` structures directly in the `phi0()` suboption. You can specify initial values for the correlation parameter ρ of the `sppower` structure in the `rho0()` suboption. Initial values for the nugget proportion η can be specified in the `nugget0()` suboption when a nugget effect is estimated.

Alternatively, you can use the `distptile0()` suboption to request that `mixed` construct initial values for range, ϕ, and ρ automatically from a representative distance D^*. Let d_{ijk} denote the distance between two observations i and k within group j, computed under the chosen metric (Euclidean by default). When you specify `distptile0(#)`, the value # is interpreted as a percentile of the empirical distribution of all within-group pairwise distances $\{d_{ijk}\}$, and D^* is set equal to that percentile. For example, `distptile0(25)` sets D^* equal to the 25th percentile of all within-group distances. When `by()` is specified, *numlist* in `distptile0()` may be used to supply distinct percentiles for each level of the `by()` variable; these are then used to compute a separate D^* and thus separate initial values for each group.

Once D^* is determined, `mixed` targets a small correlation level $c = 0.05$ at the distance D^* and solves for the corresponding spatial correlation parameter. This ensures that the initial correlation function is neither too flat nor too steep at distances that are typical in the data, which often improves convergence during optimization.

Conceptually, for spatial correlation structures with asymptotic decay (all but the `spspherical` structure), D^* represents an initial guess at the distance beyond which spatial correlation becomes negligible (0.05). For the `spspherical` structure, which has a finite range, D^* is the range parameter; for other structures with asymptotic decay, D^* is mapped to the corresponding model parameter via the kernel-specific transformations described below.

- `spexponential`: the decay parameter is initialized as

$$\phi_0 = -\frac{D^*}{\ln(0.05)} \approx \frac{D^*}{3}$$

  so that $\text{Corr}(\epsilon_{ij}, \epsilon_{kj}) \approx 0.05$ when $d_{ijk} = D^*$.

- `spgaussian`: the scale parameter is initialized as

$$\phi_0 = \frac{D^*}{\sqrt{-\ln(0.05)}} \approx \frac{D^*}{\sqrt{3}}$$

- `spspherical`: because the spherical structure has a finite geostatistical range, the range parameter is initialized directly as

$$\phi_0 = D^*$$

- `sppower`: the correlation parameter is initialized as

$$\rho_0 = \exp\left\{\frac{\ln(0.05)}{D^*}\right\} \approx \exp\left(-\frac{3}{D^*}\right)$$

▷ Example 8: Autoregressive residual variance structure

Pinheiro and Bates (2000, chap. 5) analyze data from a study of the estrus cycles of mares. Originally analyzed in Pierson and Ginther (1987), the data record the number of ovarian follicles larger than 10mm, daily over a period ranging from three days before ovulation to three days after the subsequent ovulation.

```
. use https://www.stata-press.com/data/r19/ovary
(Ovarian follicles in mares)
. describe
Contains data from https://www.stata-press.com/data/r19/ovary.dta
 Observations:           308                  Ovarian follicles in mares
    Variables:             6                  20 May 2024 13:49
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
mare	byte	%9.0g		Mare ID
stime	float	%9.0g		Scaled time
follicles	byte	%9.0g		Number of ovarian follicles > 10 mm in diameter
sin1	float	%9.0g		sine(2*pi*stime)
cos1	float	%9.0g		cosine(2*pi*stime)
time	byte	%9.0g		Time order within mare

```
Sorted by: mare  stime
```

The `stime` variable is time that has been scaled so that ovulation occurs at scaled times 0 and 1, and the `time` variable records the time ordering within mares. Because graphical evidence suggests a periodic behavior, the analysis includes the `sin1` and `cos1` variables, which are sine and cosine transformations of scaled time, respectively.

We consider the following model for the ith measurement on the jth mare:

$$\texttt{follicles}_{ij} = \beta_0 + \beta_1 \texttt{sin1}_{ij} + \beta_2 \texttt{cos1}_{ij} + u_j + \epsilon_{ij}$$

The above model incorporates the cyclical nature of the data as affecting the overall average number of follicles and includes mare-specific random effects u_j. Because we believe successive measurements within each mare are probably correlated (even after controlling for the periodicity in the average), we also model the within-mare errors as being AR of order 2.

```
. mixed follicles sin1 cos1 || mare:, residuals(ar 2, t(time)) reml nolog
Mixed-effects REML regression                   Number of obs     =        308
Group variable: mare                            Number of groups  =         11
                                                Obs per group:
                                                              min =         25
                                                              avg =       28.0
                                                              max =         31
                                                Wald chi2(2)      =      34.72
Log restricted-likelihood = -772.59855          Prob > chi2       =     0.0000
```

follicles	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
sin1	-2.899228	.5110784	-5.67	0.000	-3.900923	-1.897532
cos1	-.8652936	.5432923	-1.59	0.111	-1.930127	.1995397
_cons	12.14455	.9473731	12.82	0.000	10.28773	14.00137

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
mare: Identity				
var(_cons)	7.09265	4.402051	2.101409	23.93903
Residual: AR(2)				
phi1	.5386103	.0624897	.4161328	.6610878
phi2	.1446711	.0632039	.0207938	.2685484
var(e)	14.25103	2.435226	10.19512	19.9205

LR test vs. linear model: chi2(3) = 251.67 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

We picked an order of 2 as a guess, but we could have used LR tests of competing AR models to determine the optimal order, because models of smaller order are nested within those of larger order.

◁

▷ Example 9: Unstructured residual variance structure

Fitzmaurice, Laird, and Ware (2011, chap. 7) analyzed data on 37 subjects who participated in an exercise therapy trial.

```
. use https://www.stata-press.com/data/r19/exercise
(Exercise Therapy Trial)
. describe
Contains data from https://www.stata-press.com/data/r19/exercise.dta
 Observations:           259                  Exercise Therapy Trial
    Variables:             4                  24 Jun 2024 18:35
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
id	byte	%9.0g		Person ID
day	byte	%9.0g		Day of measurement
program	byte	%9.0g		1 = reps increase; 2 = weights increase
strength	byte	%9.0g		Strength measurement

Sorted by: id day

Subjects (variable id) were placed on either an increased-repetition regimen (program==1) or a program that kept the repetitions constant but increased weight (program==2). Muscle-strength measurements (variable strength) were taken at baseline (day==0) and then every two days over the next twelve days.

Following Fitzmaurice, Laird, and Ware (2011, chap. 7), and to demonstrate fitting residual-error structures to data collected at uneven time points, we confine our analysis to those data collected at baseline and at days 4, 6, 8, and 12. We fit a full two-way factorial model of strength on program and day, with an unstructured error covariance matrix over those repeated measurements taken on the same subject:

```
. keep if inlist(day, 0, 4, 6, 8, 12)
(74 observations deleted)

. mixed strength i.program##i.day || id:,
> noconstant residuals(unstructured, t(day)) nolog
```

```
Mixed-effects ML regression                     Number of obs     =       173
Group variable: id                              Number of groups  =        37
                                                Obs per group:
                                                              min =         3
                                                              avg =       4.7
                                                              max =         5
                                                Wald chi2(9)      =     45.85
Log likelihood = -296.58215                     Prob > chi2       =    0.0000
```

strength	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
2.program	1.360119	1.003549	1.36	0.175	-.606801	3.327039
day						
4	1.125	.3322582	3.39	0.001	.4737858	1.776214
6	1.360127	.3766893	3.61	0.000	.6218299	2.098425
8	1.583563	.4905876	3.23	0.001	.6220286	2.545097
12	1.623576	.5372947	3.02	0.003	.5704977	2.676654
program#day						
2 4	-.169034	.4423472	-0.38	0.702	-1.036019	.6979505
2 6	.2113011	.4982384	0.42	0.671	-.7652283	1.187831
2 8	-.1299762	.6524813	-0.20	0.842	-1.408816	1.148864
2 12	.3212829	.7306782	0.44	0.660	-1.11082	1.753386
_cons	79.6875	.7560446	105.40	0.000	78.20568	81.16932

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: (empty)				
Residual: Unstructured				
var(e0)	9.145655	2.12623	5.798597	14.42469
var(e4)	11.87113	2.761183	7.524988	18.72745
var(e6)	10.0657	2.348831	6.371126	15.90274
var(e8)	13.22463	3.113881	8.336026	20.98012
var(e12)	13.16908	3.167313	8.219245	21.09984
cov(e0,e4)	9.62523	2.331941	5.054709	14.19575
cov(e0,e6)	8.489038	2.106348	4.360672	12.6174
cov(e0,e8)	9.280408	2.369519	4.636236	13.92458
cov(e0,e12)	8.898001	2.348209	4.295596	13.50041
cov(e4,e6)	10.49184	2.492494	5.606641	15.37704
cov(e4,e8)	11.89786	2.84871	6.314494	17.48123
cov(e4,e12)	11.28344	2.804988	5.785761	16.78111
cov(e6,e8)	11.05069	2.646951	5.862765	16.23862
cov(e6,e12)	10.5006	2.590242	5.423814	15.57738
cov(e8,e12)	12.4091	3.010757	6.508123	18.31007

```
LR test vs. linear model: chi2(14) = 314.67                   Prob > chi2 = 0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space.  If this is not true, then the
      reported test is conservative.
```

Because we are using the variable id only to group the repeated measurements and not to introduce random effects at the subject level, we use the noconstant option to omit any subject-level effects. The unstructured covariance matrix is the most general and contains many parameters. In this example, we estimate a distinct error variance for each day and a distinct covariance for each pair of days.

That there is positive covariance between all pairs of measurements is evident, but what is not as evident is whether the covariances may be more parsimoniously represented. One option would be to explore whether the correlation diminishes as the time gap between strength measurements increases and whether it diminishes systematically. Given the irregularity of the time intervals, an exponential structure would be more appropriate than, say, an AR or MA structure.

```
. estimates store unstructured

. mixed strength i.program##i.day || id:, noconstant
> residuals(exponential, t(day)) nolog nofetable

Mixed-effects ML regression                     Number of obs     =       173
Group variable: id                              Number of groups  =        37
                                                Obs per group:
                                                              min =         3
                                                              avg =       4.7
                                                              max =         5
                                                Wald chi2(9)      =     36.77
Log likelihood = -307.83324                     Prob > chi2       =    0.0000
```

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: (empty)				
Residual: Exponential				
rho	.9786462	.0051238	.9659207	.9866854
var(e)	11.22349	2.338372	7.460764	16.88389

```
LR test vs. linear model: chi2(1) = 292.17                Prob > chi2 = 0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

In the above example, we suppressed displaying the main regression parameters because they did not differ much from those of the previous model. While the unstructured model estimated 15 variance–covariance parameters, the exponential model claims to get the job done with just 2, a fact that is not disputed by an LR test comparing the two nested models (at least not at the 0.01 level).

```
. lrtest unstructured .

Likelihood-ratio test
Assumption: . nested within unstructured

LR chi2(13) =  22.50
Prob > chi2 = 0.0481

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

◁

▷ Example 10: Spatial residual covariance structure with nugget effect (StataNow)

We analyze data from a simulated agricultural field trial measuring wheat yield across multiple farms. These fictional data come from a study design in which 12 farms were sampled, with 60 plots measured per farm for a total of 720 observations. At each plot, wheat yield (`yield`) was recorded along with nitrogen fertilizer application (`nitrogen`) and rainfall (`rainfall`). The spatial coordinates (x and y in meters) of each plot within its farm are stored in variables `x` and `y`, respectively.

```
. use https://www.stata-press.com/data/r19/wheatyield, clear
(Fictional dataset of wheat yield from multifarm agricultural trial)
. describe
Contains data from https://www.stata-press.com/data/r19/wheatyield.dta
 Observations:           720                  Fictional dataset of wheat
                                                yield from multifarm
                                                agricultural trial
    Variables:             9                  24 Jun 2026 13:25
─────────────────────────────────────────────────────────────────────────
Variable      Storage   Display    Value
    name         type    format    label      Variable label
─────────────────────────────────────────────────────────────────────────
plotid          int     %12.0g                Plot ID
farmid          byte    %12.0g                Farm ID
yield           double  %10.0g                Wheat yield (tons/ha)
nitrogen        double  %10.0g                Nitrogen fertilizer application
                                                (kg/ha)
rainfall        double  %10.0g                Rainfall during growing season
                                                (mm)
nitrogen_c      double  %10.0g                Nitrogen centered at 100 kg/ha
rainfall_c      double  %10.0g                Rainfall centered at 500 mm
x               double  %10.0g                X coordinate within farm (meters)
y               double  %10.0g                Y coordinate within farm (meters)
─────────────────────────────────────────────────────────────────────────
Sorted by:
```

The study wants to quantify the effects of nitrogen and rainfall on wheat yield, while accounting for two sources of spatial dependence: farm-to-farm variation (captured by random intercepts) and within-farm spatial correlation (modeled through spatial error covariance structures). Additionally, because yield measurements exhibit inherent measurement error and because microscale variation exists below the sampling resolution, we expect a nugget effect (a discontinuity in the spatial correlation at zero distance).

In the mixed model below, the predictors `nitrogen_c` and `rainfall_c` are centered at (roughly) their sample means (100 kg/ha and 500 mm, respectively). The spatial coordinates `x` and `y` define plot locations within each farm on a grid 300 meters by 300 meters.

We fit a linear mixed model for the yield $\texttt{yield}_{ij}$ of plot i within farm j,

$$\texttt{yield}_{ij} = \beta_0 + \beta_1\,\texttt{nitrogen_c}_{ij} + \beta_2\,\texttt{rainfall_c}_{ij} + u_j + \epsilon_{ij}$$

where β_0 represents the expected yield at average nitrogen and rainfall, β_1 and β_2 are the fixed effects of nitrogen and rainfall, u_j's $\sim N(0, \sigma_u^2)$ are the farm-specific random intercepts, and ϵ_{ij}'s are the errors. We model the errors by assuming a spherical spatial correlation structure that includes a nugget effect. We specify the `spspherical` structure with the `nugget` suboption in the `residuals()` option to fit this model.

```
. mixed yield nitrogen_c rainfall_c || farmid:,
> residuals(spspherical, coordinates(x y) nugget) nolog reml
Mixed-effects REML regression                   Number of obs     =        720
Group variable: farmid                          Number of groups  =         12
                                                Obs per group:
                                                              min =         60
                                                              avg =       60.0
                                                              max =         60
                                                Wald chi2(2)      =     892.70
Log restricted-likelihood = -452.51046          Prob > chi2       =     0.0000
```

yield	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
nitrogen_c	.0145382	.0005215	27.88	0.000	.0135161	.0155602
rainfall_c	.0082663	.0007632	10.83	0.000	.0067704	.0097622
_cons	4.384632	.1695019	25.87	0.000	4.052414	4.716849

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
farmid: Identity				
var(_cons)	.3347489	.1472153	.1413771	.7926093
Residual: Spatial spherical				
Range	109.2307	10.80181	89.98486	132.5928
Nugget	.3861334	.0704936	.2599008	.5297884
var(e)	.2187123	.0138238	.1932292	.2475562

```
LR test vs. linear model: chi2(3) = 673.00                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
. estimates store spher_nug
```

The estimated fixed effects indicate, for instance, that each additional kilogram per hectare of nitrogen fertilizer above the sample mean increases yield by approximately 0.015 tons per hectare, with a 95% CI of $[0.0135, 0.0156]$. The intercept of 4.38 tons per hectare represents the expected yield under average conditions (nitrogen $=$ 100 kg/ha, rainfall $=$ 500 mm) for a farm with random effect $u_j = 0$.

The estimated range parameter is about 109 meters, indicating that plots within approximately 109 meters of each other exhibit spatial correlation. Beyond this distance, the spherical structure assigns 0 correlation.

The nugget effect accounts for 39% of the total error variance (nugget ratio ≈ 0.3861). This suggests that a substantial portion of within-farm error variation is attributable to measurement error in yield assessment and to microscale spatial heterogeneity operating at distances smaller than the plot spacing. The presence of a sizable nugget effect justifies its inclusion in the model.

After a spatial mixed model is fit, it is important to conduct a sensitivity analysis by varying the initial value of the spatial correlation parameter. In this example, the spatial correlation is controlled by the range, and we can use the `range0()` suboption to specify different values for it. Doing so helps assess the robustness of the results and reduces the risk of convergence to a local maximum of the likelihood function; see *Initial values for spatial error parameters* for details.

We store this model as `spher_nug` for further analysis:

```
. estimates store spher_nug
```

For illustration, we also fit two additional spatial mixed-effects models: one with the same spatial error structure as above but without a nugget effect and one with a Gaussian spatial error structure and no nugget effect. We store the results as `spher`. The goal is to compare the three candidate models using information-criterion measures.

```
. mixed yield nitrogen_c rainfall_c || farmid:,
> residuals(spspherical, coordinates(x y)) nolog reml
Mixed-effects REML regression                   Number of obs     =        720
Group variable: farmid                          Number of groups  =         12
                                                Obs per group:
                                                              min =         60
                                                              avg =       60.0
                                                              max =         60
                                                Wald chi2(2)      =     887.84
Log restricted-likelihood = -465.41807          Prob > chi2       =     0.0000
```

yield	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
nitrogen_c	.0139925	.0005064	27.63	0.000	.0129999	.0149851
rainfall_c	.0085881	.0007584	11.32	0.000	.0071016	.0100745
_cons	4.380038	.1700379	25.76	0.000	4.04677	4.713306

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
farmid: Identity				
var(_cons)	.3388631	.1480653	.1439111	.797911
Residual: Spatial spherical				
Range	72.49896	3.894097	65.25468	80.54747
var(e)	.2301023	.01405	.2041487	.2593554

```
LR test vs. linear model: chi2(2) = 647.18                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
. estimates store spher
```

Below, we fit the Gaussian spatial error structure model and store it as gauss.

```
. mixed yield nitrogen_c rainfall_c || farmid:,
> residuals(spgaussian, coordinates(x y)) nolog reml
Mixed-effects REML regression                   Number of obs     =        720
Group variable: farmid                          Number of groups  =         12
                                                Obs per group:
                                                              min =         60
                                                              avg =       60.0
                                                              max =         60
                                                Wald chi2(2)      =     798.80
Log restricted-likelihood = -483.63586          Prob > chi2       =     0.0000
```

yield	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
nitrogen_c	.0134571	.0005136	26.20	0.000	.0124505	.0144636
rainfall_c	.0090491	.0007753	11.67	0.000	.0075295	.0105687
_cons	4.374288	.1672699	26.15	0.000	4.046445	4.702131

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
farmid: Identity				
var(_cons)	.3296814	.1431761	.1407438	.7722529
Residual: Spatial Gaussian				
phi	26.45716	1.511448	23.6546	29.59176
var(e)	.2232835	.012767	.1996118	.2497623

```
LR test vs. linear model: chi2(2) = 610.75                Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

. estimates store gauss
```

Compared with a spherical spatial model, a Gaussian spatial model estimates a parameter phi instead of parameter range. This parameter controls the rate at which spatial correlation decays as the distance between two within-farm observations increases. Observations are approximately uncorrelated (correlation $\leq$ 0.05) when separated by distances greater than the practical range, which is approximately $\sqrt{3}\phi = \sqrt{3} \times 26.46 = 45.83$ meters in this example; see *Spatial residual-error structures* for details.

We use estimates stats to compute the information criteria (AIC, BIC) and compare the three spatial models:

```
. estimates stats spher_nug spher gauss
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
spher_nug	720	.	-452.5105	7	919.0209	951.0757
spher	720	.	-465.4181	6	942.8361	970.3117
gauss	720	.	-483.6359	6	979.2717	1006.747

```
Note: BIC uses N = number of observations. See [R] IC note.
```

The spherical spatial model with a nugget effect provides the best fit for this dataset, as indicated by the lowest AIC and BIC values.

◁

Example 11: Specifying initial values for spatial correlation parameters (StataNow)

In this example, we illustrate the use of the `phi0()` and `distptile0()` suboptions for initializing spatial correlation parameters. The data come from a subset of the Meuse environmental dataset (Rikken and Van Rijn 1993). For each sampling location, zinc concentration in soil (mg/Kg) was measured, along with distance to the river (`dist`), percentage of organic matter (`om`), and spatial coordinates (`x`, `y`). We analyze `log_zinc` as the response.

```
. use https://www.stata-press.com/data/r19/zinc, clear
(Zinc concentration in floodplain soil of the river Meuse)
. describe
Contains data from https://www.stata-press.com/data/r19/zinc.dta
 Observations:           134                  Zinc concentration in
                                                floodplain soil of the river
                                                Meuse
    Variables:             6                  25 Jun 2026 13:06
                                              (_dta has notes)
------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
x               long    %12.0g                Easting (m) in the Dutch
                                                topographical map coordinates
y               long    %12.0g                Northing (m) in the Dutch
                                                topographical map coordinates
dist            float   %9.0g                 Distance to the Meuse river
                                                (normalized to [0, 1])
om              float   %9.0g                 Organic matter content (as a
                                                percentage)
soil            byte    %8.0g                 Soil type
log_zinc        float   %9.0g                 Topsoil log concentration of zinc
                                                (mg/Kg)
------------------------------------------------------------------------------
Sorted by:
```

We fit a random-intercept model for the zinc log-concentration $\texttt{log_zinc}_{ij}$ at location i within soil type j,

$$\texttt{log_zinc}_{ij} = \beta_0 + \beta_1 \texttt{dist}_{ij} + \beta_2 \texttt{om}_{ij} + u_j + \epsilon_{ij},$$

where β_0, β_1, and β_2 are the fixed-effects coefficients. u_j's $\sim N(0, \sigma_u^2)$ are the soil-specific random intercepts, and ϵ_{ij}'s are the errors. We model the errors by assuming an exponential spatial correlation structure (`spexponential`).

```
. mixed log_zinc dist om || soil:, residual(spexponential, coordinates(x y))
> reml

Obtaining starting values by EM ...

Performing gradient-based optimization:
Iteration 0:  Log restricted-likelihood = -74.566606
Iteration 1:  Log restricted-likelihood = -74.227293
Iteration 2:  Log restricted-likelihood = -74.215435
Iteration 3:  Log restricted-likelihood = -74.215434

Computing standard errors ...

Mixed-effects REML regression                   Number of obs     =        134
Group variable: soil                            Number of groups  =          3
                                                Obs per group:
                                                              min =         12
                                                              avg =       44.7
                                                              max =         78
                                                Wald chi2(2)      =     122.01
Log restricted-likelihood = -74.215434          Prob > chi2       =     0.0000
```

log_zinc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
dist	-1.870884	.2900412	-6.45	0.000	-2.439354	-1.302414
om	.0889941	.0151355	5.88	0.000	.0593292	.1186591
_cons	5.680536	.1653298	34.36	0.000	5.356495	6.004576

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
soil: Identity				
var(_cons)	3.60e-16	1.97e-12	0	.
Residual: Spatial exponential				
phi	133.537	30.52712	85.31266	209.021
var(e)	.2047883	.0307949	.1525129	.2749818

```
LR test vs. linear model: chi2(2) = 25.09                 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.
```

The random-intercept variance is estimated as essentially zero. This situation is common in spatial mixed models, especially when the number of groups is small. Between-group random effects and within-group spatial correlation often compete to explain the variability in the data. In this example, only three soil types are observed, and the exponential spatial error structure can capture most of the within-soil dependence. As a result, a negligible amount of additional variability remains to be explained by the soil-specific random intercepts.

Next we remove the random intercept from the model and estimate a generalized least-squares model (a model with no random effects but with a within-group error covariance structure).

```
. mixed log_zinc dist om || soil:,  noconstant
> residual(spexponential, coordinates(x y) ) reml nolog
Mixed-effects REML regression                   Number of obs     =       134
Group variable: soil                            Number of groups  =         3
                                                Obs per group:
                                                              min =        12
                                                              avg =      44.7
                                                              max =        78
                                                Wald chi2(2)      =    122.01
Log restricted-likelihood = -74.215434          Prob > chi2       =    0.0000
```

log_zinc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
dist	-1.870884	.2900412	-6.45	0.000	-2.439354	-1.302414
om	.0889941	.0151355	5.88	0.000	.0593292	.1186591
_cons	5.680536	.1653298	34.36	0.000	5.356495	6.004576

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
soil: (empty)				
Residual: Spatial exponential				
phi	133.537	30.52712	85.31267	209.021
var(e)	.2047883	.0307949	.1525129	.2749818

```
LR test vs. linear model: chi2(1) = 25.09                 Prob > chi2 = 0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space.  If this is not true, then the
      reported test is conservative.
```

For the spatial exponential model, observations are approximately uncorrelated (correlation $\leq$ 0.05) when separated by distances greater than the practical range, which is approximately $3\phi = 3 \times 133.54 = 400.62$ meters in this example; see *Spatial residual-error structures* for details.

Next we investigate the robustness of our results to the choice of initial values by specifying a list of initial values for the spatial scale parameter `phi`. Another scenario in which you may need to specify alternative initial values is when the model fails to converge or issues an error such as

```
could not calculate numerical derivatives -- discontinuous region with missing values
encountered
```

The error shown above indicates that the optimizer has encountered flat or discontinuous regions of the likelihood, and the model failed to converge.

Below, we provide four initial values (50, 100, 200, and 400) for the spatial scale parameter phi by using the phi0() suboption of residuals(). We store the corresponding models as exp_50, exp_100, exp_200, and exp_400, respectively.

```
. local init_vals 50 100 200 400
. foreach i of local init_vals {
  2.         quietly mixed log_zinc dist om || soil:, noconstant
>                 residual(spexponential, coordinates(x y) phi0(‘i’)) reml
  3.         estimates store exp_‘i’
  4. }
```

We will use the etable command to compare the results from the four models; see [R] **etable** for details. We also report the log restricted-likelihood value by using the mstat() option.

```
. etable, estimates(exp_50 exp_100 exp_200 exp_400) column(estimates) mstat(ll)
```

	exp_50	exp_100	exp_200	exp_400
Distance to the Meuse river (normalized ...)	-1.871	-1.871	-1.871	-1.871
	(0.290)	(0.290)	(0.290)	(0.290)
Organic matter content (as a percentage)	0.089	0.089	0.089	0.089
	(0.015)	(0.015)	(0.015)	(0.015)
Intercept	5.681	5.681	5.681	5.681
	(0.165)	(0.165)	(0.165)	(0.165)
phi	133.537	133.537	133.537	133.537
	(30.527)	(30.527)	(30.527)	(30.527)
var(e)	0.205	0.205	0.205	0.205
	(0.031)	(0.031)	(0.031)	(0.031)
Log restricted-likelihood	-74.22	-74.22	-74.22	-74.22

The results of the four models are identical, suggesting that our results are robust to the specification of the initial value for the spatial scale parameter phi. In general, when the likelihood surface for spatial models contains multiple local maxima, we would typically choose the model corresponding to the highest log restricted-likelihood value.

Alternatively, we could have used the distptile0() option to (indirectly) specify an initial value for phi. This is done by specifying a distance D^*, which is used in computing the initial value $\phi^{(0)}$; see *Initial values for spatial error parameters* for details. Below, we specify that the initial value of phi should be based on the 15th percentile of all within-soil pairwise distances (D^*).

```
. mixed log_zinc dist om || soil:, noconstant
> residual(spexponential, coordinates(x y) distptile0(15)) reml

Performing gradient-based optimization:
Iteration 0:  Log restricted-likelihood = -74.444289
Iteration 1:  Log restricted-likelihood = -74.215438
Iteration 2:  Log restricted-likelihood = -74.215434

Computing standard errors ...

Mixed-effects REML regression                   Number of obs     =        134
Group variable: soil                            Number of groups  =          3
                                                Obs per group:
                                                              min =         12
                                                              avg =       44.7
                                                              max =         78
                                                Wald chi2(2)      =     122.01
Log restricted-likelihood = -74.215434          Prob > chi2       =     0.0000
```

log_zinc	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
dist	-1.870884	.2900411	-6.45	0.000	-2.439354	-1.302414
om	.0889941	.0151355	5.88	0.000	.0593292	.1186591
_cons	5.680536	.1653298	34.36	0.000	5.356495	6.004576

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
soil: (empty)				
Residual: Spatial exponential				
phi	133.537	30.52712	85.31266	209.021
var(e)	.2047883	.0307949	.1525129	.2749817

```
LR test vs. linear model: chi2(1) = 25.09                 Prob > chi2 = 0.0000

Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space.  If this is not true, then the
      reported test is conservative.
```

The model yields results identical to those from the earlier specifications.

◁

Crossed-effects models

Not all mixed models contain nested levels of random effects.

▷ Example 12: Crossed-effects model

Returning to our longitudinal analysis of pig weights, suppose that instead of (5) we wish to fit

$$\texttt{weight}_{ij} = \beta_0 + \beta_1\texttt{week}_{ij} + u_i + v_j + \epsilon_{ij} \tag{8}$$

for the $i = 1, \ldots, 9$ weeks and $j = 1, \ldots, 48$ pigs and

$$u_i \sim N(0, \sigma_u^2); \quad v_j \sim N(0, \sigma_v^2); \quad \epsilon_{ij} \sim N(0, \sigma_\epsilon^2)$$

all independently. Both (5) and (8) assume an overall population-average growth curve $\beta_0 + \beta_1\texttt{week}$ and a random pig-specific shift.

The models differ in how `week` enters into the random part of the model. In (5), we assume that the effect due to `week` is linear and pig specific (a random slope); in (8), we assume that the effect due to `week`, u_i, is systematic to that week and common to all pigs. The rationale behind (8) could be that, assuming that the pigs were measured contemporaneously, we might be concerned that week-specific random factors such as weather and feeding patterns had significant systematic effects on all pigs.

Model (8) is an example of a two-way crossed-effects model, with the pig effects v_j being crossed with the week effects u_i. One way to fit such models is to consider all the data as one big cluster, and treat the u_i and v_j as a series of $9 + 48 = 57$ random coefficients on indicator variables for `week` and `pig`. In the notation of (2),

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_9 \\ v_1 \\ \vdots \\ v_{48} \end{bmatrix} \sim N(\mathbf{0}, \mathbf{G}); \quad \mathbf{G} = \begin{bmatrix} \sigma_u^2\mathbf{I}_9 & \mathbf{0} \\ \mathbf{0} & \sigma_v^2\mathbf{I}_{48} \end{bmatrix}$$

Because $\mathbf{G}$ is block diagonal, it can be represented in `mixed` as repeated-level equations. All we need is an identification variable to identify all the observations as one big group and a way to tell `mixed` to treat `week` and `pig` as crossed-effects factor variables (or equivalently, as two sets of overparameterized indicator variables identifying weeks and pigs, respectively). `mixed` supports the special group designation `_all` for the former and the `R.`*varname* notation for the latter.

```
. use https://www.stata-press.com/data/r19/pig, clear
(Longitudinal analysis of pig weights)
. mixed weight week || _all: R.week || _all: R.id
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood =  -1013.824
Iteration 1:  Log likelihood =  -1013.824
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: _all                            Number of groups  =          1
                                                Obs per group:
                                                              min =        432
                                                              avg =      432.0
                                                              max =        432
                                                Wald chi2(1)      =   13258.28
Log likelihood =  -1013.824                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0539313	115.14	0.000	6.104192	6.315599
_cons	19.35561	.6333982	30.56	0.000	18.11418	20.59705

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
_all: Identity				
var(R.week)	.0849874	.0868856	.0114588	.6303302
_all: Identity				
var(R.id)	14.83623	3.126142	9.816733	22.42231
var(Residual)	4.297328	.3134404	3.724888	4.957741

```
LR test vs. linear model: chi2(2) = 474.85                    Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
. estimates store crossed
```

Thus we estimate $\hat{\sigma}_u^2 = 0.08$ and $\hat{\sigma}_v^2 = 14.84$. Both (5) and (8) estimate a total of five parameters: two fixed effects and three variance components. The models, however, are not nested within each other, which precludes the use of an LR test to compare both models. Refitting model (5) and looking at the Akaike information criteria values by using `estimates stats`,

```
. quietly mixed weight week || id:week
. estimates stats crossed .
Akaike's information criterion and Bayesian information criterion
```

Model	N	ll(null)	ll(model)	df	AIC	BIC
crossed	432	.	-1013.824	5	2037.648	2057.99
.	432	.	-869.0383	5	1748.077	1768.419

```
Note: BIC uses N = number of observations. See [R] IC note.
```

definitely favors model (5). This finding is not surprising given that our rationale behind (8) was somewhat fictitious. In our `estimates stats` output, the values of `ll(null)` are missing. `mixed` does not fit a constant-only model as part of its usual estimation of the full model, but you can use `mixed` to fit a constant-only model directly, if you wish.

◁

The `R.`*varname* notation is equivalent to giving a list of overparameterized (none dropped) indicator variables for use in a random-effects specification. When you specify `R.`*varname*, `mixed` handles the calculations internally rather than creating the indicators in the data. Because the set of indicators is overparameterized, `R.`*varname* implies `noconstant`.

❑ Technical note

Although we were able to fit the crossed-effects model (8), it came at the expense of increasing the column dimension of our random-effects design from 2 in model (5) to 57 in model (8). Computation time and memory requirements grow (roughly) quadratically with the dimension of the random effects. As a result, fitting such crossed-effects models is feasible only when the total column dimension is small to moderate.

Reexamining model (8), we note that if we drop u_i, we end up with a model equivalent to (4), meaning that we could have fit (4) by typing

```
. mixed weight week || _all: R.id
```

instead of

```
. mixed weight week || id:
```

as we did when we originally fit the model. The results of both estimations are identical, but the latter specification, organized at the cluster (pig) level with random-effects dimension 1 (a random intercept) is much more computationally efficient. Whereas with the first form we are limited in how many pigs we can analyze, there is no such limitation with the second form.

Furthermore, we fit model (8) by using

```
. mixed weight week || _all: R.week || _all: R.id
```

as a direct way to demonstrate the `R.` notation. However, we can technically treat pigs as nested within the `_all` group, yielding the equivalent and more efficient (total column dimension 10) way to fit (8):

```
. mixed weight week || _all: R.week || id:
```

We leave it to you to verify that both produce identical results. See Rabe-Hesketh and Skrondal (2022) for additional techniques to make calculations more efficient in more complex models.

❑

▷ Example 13: Three-level model expressed in terms of a two-level model

As another example of how the same model may be fit in different ways by using `mixed` (and as a way to demonstrate `covariance(exchangeable)`), consider the three-level model used in example 4:

$$\mathbf{y}_{jk} = \mathbf{X}_{jk}\boldsymbol{\beta} + u_k^{(3)} + u_{jk}^{(2)} + \boldsymbol{\epsilon}_{jk}$$

where $\mathbf{y}_{jk}$ represents the logarithms of gross state products for the $n_{jk} = 17$ observations from state j in region k, $\mathbf{X}_{jk}$ is a set of regressors, $u_k^{(3)}$ is a random intercept at the region level, and $u_{jk}^{(2)}$ is a random intercept at the state (nested within region) level. We assume that $u_k^{(3)} \sim N(0, \sigma_3^2)$ and $u_{jk}^{(2)} \sim N(0, \sigma_2^2)$ independently. Define

$$\mathbf{v}_k = \begin{bmatrix} u_k^{(3)} + u_{1k}^{(2)} \\ u_k^{(3)} + u_{2k}^{(2)} \\ \vdots \\ u_k^{(3)} + u_{M_k,k}^{(2)} \end{bmatrix}$$

where M_k is the number of states in region k. Making this substitution, we can stack the observations for all the states within region k to get

$$\mathbf{y}_k = \mathbf{X}_k\boldsymbol{\beta} + \mathbf{Z}_k\mathbf{v}_k + \boldsymbol{\epsilon}_k$$

where $\mathbf{Z}_k$ is a set of indicators identifying the states within each region; that is,

$$\mathbf{Z}_k = \mathbf{I}_{M_k} \otimes \mathbf{J}_{17}$$

for a k-column vector of 1s $\mathbf{J}_k$, and

$$\boldsymbol{\Sigma} = \mathrm{Var}(\mathbf{v}_k) = \begin{bmatrix} \sigma_3^2 + \sigma_2^2 & \sigma_3^2 & \cdots & \sigma_3^2 \\ \sigma_3^2 & \sigma_3^2 + \sigma_2^2 & \cdots & \sigma_3^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_3^2 & \sigma_3^2 & \sigma_3^2 & \sigma_3^2 + \sigma_2^2 \end{bmatrix}_{M_k \times M_k}$$

Because $\boldsymbol{\Sigma}$ is an exchangeable matrix, we can fit this alternative form of the model by specifying the `exchangeable` covariance structure.

```
. use https://www.stata-press.com/data/r19/productivity
(Public capital productivity)
. mixed gsp private emp hwy water other unemp || region: R.state,
> cov(exchangeable)
```

(*output omitted*)

```
Mixed-effects ML regression                     Number of obs     =        816
Group variable: region                          Number of groups  =          9
                                                Obs per group:
                                                              min =         51
                                                              avg =       90.7
                                                              max =        136
                                                Wald chi2(6)      =   18829.06
Log likelihood =  1430.5017                     Prob > chi2       =     0.0000
```

gsp	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
private	.2671484	.0212591	12.57	0.000	.2254813	.3088154
emp	.7540721	.0261868	28.80	0.000	.7027468	.8053973
hwy	.0709767	.023041	3.08	0.002	.0258172	.1161363
water	.0761187	.0139248	5.47	0.000	.0488266	.1034109
other	-.0999955	.0169366	-5.90	0.000	-.1331907	-.0668004
unemp	-.0058983	.0009031	-6.53	0.000	-.0076684	-.0041282
_cons	2.128823	.1543855	13.79	0.000	1.826233	2.431413

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
region: Exchangeable				
var(R.state)	.0077263	.0017926	.0049032	.0121749
cov(R.state)	.0014506	.0012995	-.0010963	.0039975
var(Residual)	.0013461	.0000689	.0012176	.0014882

LR test vs. linear model: chi2(2) = 1154.73 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

The estimates of the fixed effects and their standard errors are equivalent to those from example 4, and remapping the variance components from $(\sigma_3^2 + \sigma_2^2, \sigma_3^2, \sigma_\epsilon^2)$, as displayed here, to $(\sigma_3^2, \sigma_2^2, \sigma_\epsilon^2)$, as displayed in example 4, will show that they are equivalent as well.

Of course, given the discussion in the previous technical note, it is more efficient to fit this model as we did originally, as a three-level model.

◁

Diagnosing convergence problems

Given the flexibility of mixed-effects models, you will find that some models fail to converge when used with your data; see *Diagnosing convergence problems* in [ME] **me** for advice applicable to mixed-effects models in general.

In unweighted linear mixed-effects models with independent and homoskedastic errors, one useful way to diagnose problems of nonconvergence is to rely on the EM algorithm (Dempster, Laird, and Rubin 1977), normally used by `mixed` only as a means of refining starting values. The advantages of EM are that it does not require a Hessian calculation, each successive EM iteration will result in a larger likelihood, iterations can be calculated quickly, and iterations will quickly bring parameter estimates into a neighborhood of the solution. The disadvantages of EM are that, once in a neighborhood of the solution, it can be slow to converge, if at all, and EM provides no facility for estimating standard errors of the estimated variance components. One useful property of EM is that it is always willing to provide a solution if you allow it to iterate enough times, if you are satisfied with being in a neighborhood of the optimum rather than right on the optimum, and if standard errors of variance components are not crucial to your analysis.

If you encounter a nonconvergent model, try using the `emonly` option to bypass gradient-based optimization. Use `emiterate(#)` to specify the maximum number of EM iterations, which you will usually want to set much higher than the default of 20. If your EM solution shows an estimated variance component that is near 0, a ridge is formed by an interval of values near 0, which produces the same likelihood and looks equally good to the optimizer. In this case, the solution is to drop the offending variance component from the model.

Survey data

Multilevel modeling of survey data is a little different from standard modeling in that weighted sampling can take place at multiple levels in the model, resulting in multiple sampling weights. Most survey datasets, regardless of the design, contain one overall inclusion weight for each observation in the data. This weight reflects the inverse of the probability of ultimate selection, and by "ultimate" we mean that it factors in all levels of clustered sampling, corrections for noninclusion and oversampling, poststratification, etc.

For simplicity, in what follows assume a simple two-stage sampling design where groups are randomly sampled and then individuals within groups are sampled. Also assume that no additional weight corrections are performed; that is, sampling weights are simply the inverse of the probability of selection. The sampling weight for observation i in cluster j in our two-level sample is then $w_{ij} = 1/\pi_{ij}$, where π_{ij} is the probability that observation i, j is selected. If you were performing a standard analysis such as OLS regression with `regress`, you would simply use a variable holding w_{ij} as your `pweight` variable, and the fact that it came from two levels of sampling would not concern you. Perhaps you would type `vce(cluster` *groupvar*`)` where *groupvar* identifies the top-level groups to get standard errors that control for correlation within these groups, but you would still use only a single weight variable.

Now take these same data and fit a two-level model with `mixed`. As seen in (14) in *Methods and formulas* later in this entry, it is not sufficient to use the single sampling weight w_{ij}, because weights enter into the log likelihood at both the group level and the individual level. Instead, what is required for a two-level model under this sampling design is w_j, the inverse of the probability that group j is selected in the first stage, and $w_{i|j}$, the inverse of the probability that individual i from group j is selected at the second stage *conditional on group j already being selected*. It simply will not do to just use w_{ij} without making any assumptions about w_j.

Given the rules of conditional probability, $w_{ij} = w_j w_{i|j}$. If your dataset has only w_{ij}, then you will need to either assume equal probability sampling at the first stage ($w_j = 1$ for all j) or find some way to recover w_j from other variables in your data; see Rabe-Hesketh and Skrondal (2006) and the references therein for some suggestions on how to do this, but realize that there is little yet known about how well these approximations perform in practice.

What you really need to fit your two-level model are data that contain w_j in addition to either w_{ij} or $w_{i|j}$. If you have w_{ij}—that is, the unconditional inclusion weight for observation i, j—then you need to either divide w_{ij} by w_j to obtain $w_{i|j}$ or rescale w_{ij} so that its dependence on w_j disappears. If you already have $w_{i|j}$, then rescaling becomes optional (but still an important decision to make).

Weight rescaling is not an exact science, because the scale of the level-one weights is at issue regardless of whether they represent w_{ij} or $w_{i|j}$: because w_{ij} is unique to group j, the group-to-group magnitudes of these weights need to be normalized so that they are "consistent" from group to group. This is in stark contrast to a standard analysis, where the scale of sampling weights does not factor into estimation, instead only affecting the estimate of the total population size.

`mixed` offers three methods for standardizing weights in a two-level model, and you can specify which method you want via the `pwscale()` option. If you specify `pwscale(size)`, then the $w_{i|j}$ (or w_{ij}, it does not matter) are scaled to sum to the cluster size n_j. Method `pwscale(effective)` adds in a dependence on the sum of the squared weights so that level-one weights sum to the "effective" sample size. Just like `pwscale(size)`, `pwscale(effective)` also behaves the same whether you have $w_{i|j}$ or w_{ij}, and so it can be used with either.

Although both `pwscale(size)` and `pwscale(effective)` leave w_j untouched, the `pwscale(gk)` method is a little different in that 1) it changes the weights at both levels and 2) it does assume you have $w_{i|j}$ for level-one weights and not w_{ij} (if you have the latter, then first divide by w_j). Using the method of Graubard and Korn (1996), it sets the weights at the group level (level two) to the cluster averages of the products of both level weights (this product being w_{ij}). It then sets the individual weights to 1 everywhere; see *Methods and formulas* for the computational details of all three methods.

Determining which method is "best" is a tough call and depends on cluster size (the smaller the clusters, the greater the sensitivity to scale), whether the sampling is informative (that is, the sampling weights are correlated with the residuals), whether you are interested primarily in regression coefficients or in variance components, whether you have a simple random-intercept model or a more complex random-

coefficients model, and other factors; see Rabe-Hesketh and Skrondal (2006), Carle (2009), and Pfeffermann et al. (1998) for some detailed advice. At the very least, you want to compare estimates across all three scaling methods (four, if you add no scaling) and perform a sensitivity analysis.

If you choose to rescale level-one weights, it does not matter whether you have $w_{i|j}$ or w_{ij}. For the `pwscale(size)` and `pwscale(effective)` methods, you get identical results, and even though `pwscale(gk)` assumes $w_{i|j}$, you can obtain this as $w_{i|j} = w_{ij}/w_j$ before proceeding.

If you do not specify `pwscale()`, then no scaling takes place, and thus at a minimum, you need to make sure you have $w_{i|j}$ in your data and not w_{ij}.

▷ Example 14: Mixed-effect models with survey data

Rabe-Hesketh and Skrondal (2006) analyzed data from the 2000 Programme for International Student Assessment (PISA) study on reading proficiency among 15-year-old American students, as performed by the Organisation for Economic Co-operation and Development (OECD). The original study was a three-stage cluster sample, where geographic areas were sampled at the first stage, schools at the second, and students at the third. Our version of the data does not contain the geographic-areas variable, so we treat this as a two-stage sample where schools are sampled at the first stage and students at the second.

```
. use https://www.stata-press.com/data/r19/pisa2000
(Programme for International Student Assessment (PISA) 2000 data)
. describe
Contains data from https://www.stata-press.com/data/r19/pisa2000.dta
 Observations:         2,069                  Programme for International
                                                Student Assessment (PISA) 2000
                                                data
    Variables:            11                  12 Jun 2024 10:08
                                              (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
female	byte	%8.0g		1 if female
isei	byte	%8.0g		International socioeconomic index
w_fstuwt	float	%9.0g		Student-level weight
wnrschbw	float	%9.0g		School-level weight
high_school	byte	%8.0g		1 if highest level by either parent is high school
college	byte	%8.0g		1 if highest level by either parent is college
one_for	byte	%8.0g		1 if one parent foreign born
both_for	byte	%8.0g		1 if both parents are foreign born
test_lang	byte	%8.0g		1 if English (the test language) is spoken at home
pass_read	byte	%8.0g		1 if passed reading proficiency threshold
id_school	int	%8.0g		School ID

```
Sorted by:
```

For student i in school j, where the variable `id_school` identifies the schools, the variable `w_fstuwt` is a student-level overall inclusion weight (w_{ij}, not $w_{i|j}$) adjusted for noninclusion and nonparticipation of students, and the variable `wnrschbw` is the school-level weight w_j adjusted for oversampling of schools with more minority students. The weight adjustments do not interfere with the methods prescribed above, and thus we can treat the weight variables simply as w_{ij} and w_j, respectively.

Rabe-Hesketh and Skrondal (2006) fit a two-level logistic model for passing a reading proficiency threshold. We fit a two-level linear random-intercept model for socioeconomic index. Because we have w_{ij} and not $w_{i|j}$, we rescale using `pwscale(size)` and thus obtain results as if we had $w_{i|j}$.

```
. mixed isei female high_school college one_for both_for test_lang
> [pw=w_fstuwt] || id_school:, pweight(wnrschbw) pwscale(size)
```

(*output omitted*)

```
Mixed-effects regression                        Number of obs     =  2,069
Group variable: id_school                       Number of groups  =    148
                                                Obs per group:
                                                              min =      1
                                                              avg =   14.0
                                                              max =     28
                                                Wald chi2(6)      = 187.23
Log pseudolikelihood = -1443093.9               Prob > chi2       = 0.0000

                          (Std. err. adjusted for 148 clusters in id_school)
```

isei	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
female	.59379	.8732886	0.68	0.497	-1.117824	2.305404
high_school	6.410618	1.500337	4.27	0.000	3.470011	9.351224
college	19.39494	2.121145	9.14	0.000	15.23757	23.55231
one_for	-.9584613	1.789947	-0.54	0.592	-4.466692	2.54977
both_for	-.2021101	2.32633	-0.09	0.931	-4.761633	4.357413
test_lang	2.519539	2.393165	1.05	0.292	-2.170978	7.210056
_cons	28.10788	2.435712	11.54	0.000	23.33397	32.88179

Random-effects parameters	Estimate	Robust std. err.	[95% conf.	interval]
id_school: Identity				
var(_cons)	34.69374	8.574865	21.37318	56.31617
var(Residual)	218.7382	11.22111	197.8147	241.8748

Notes:

1. We specified the level-one weights using standard Stata weight syntax, that is, `[pw=w_fstuwt]`.
2. We specified the level-two weights via the `pweight(wnrschbw)` option as part of the random-effects specification for the `id_school` level. As such, it is treated as a school-level weight. Accordingly, `wnrschbw` needs to be constant within schools, and `mixed` did check for that before estimating.
3. Because our level-one weights are unconditional, we specified `pwscale(size)` to rescale them.
4. As is the case with other estimation commands in Stata, standard errors in the presence of sampling weights are robust.
5. Robust standard errors are clustered at the top level of the model, and this will always be true unless you specify `vce(cluster` *clustvar*`)`, where *clustvar* identifies an even higher level of grouping.

As a form of sensitivity analysis, we compare the above with scaling via `pwscale(gk)`. Because `pwscale(gk)` assumes $w_{i|j}$, you want to first divide w_{ij} by w_j. But you can handle that within the weight specification itself.

```
. mixed isei female high_school college one_for both_for test_lang
> [pw=w_fstuwt/wnrschbw] || id_school:, pweight(wnrschbw) pwscale(gk)
```

(*output omitted*)

```
Mixed-effects regression                        Number of obs     =  2,069
Group variable: id_school                       Number of groups  =    148
                                                Obs per group:
                                                              min =      1
                                                              avg =   14.0
                                                              max =     28
                                                Wald chi2(6)      = 291.37
Log pseudolikelihood = -7270505.6               Prob > chi2       = 0.0000
```

(Std. err. adjusted for 148 clusters in id_school)

isei	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
female	-.3519458	.7436334	-0.47	0.636	-1.80944	1.105549
high_school	7.074911	1.139777	6.21	0.000	4.84099	9.308833
college	19.27285	1.286029	14.99	0.000	16.75228	21.79342
one_for	-.9142879	1.783091	-0.51	0.608	-4.409082	2.580506
both_for	1.214151	1.611885	0.75	0.451	-1.945085	4.373388
test_lang	2.661866	1.556491	1.71	0.087	-.3887996	5.712532
_cons	31.20145	1.907413	16.36	0.000	27.46299	34.93991

Random-effects parameters	Estimate	Robust std. err.	[95% conf.	interval]
id_school: Identity				
var(_cons)	31.67522	6.792239	20.80622	48.22209
var(Residual)	226.2429	8.150714	210.8188	242.7955

The results are somewhat similar to before, which is good news from a sensitivity standpoint. Note that we specified `[pw=w_fstwtw/wnrschbw]` and thus did the conversion from w_{ij} to $w_{i|j}$ within our call to `mixed`.

◁

We close this section with a bit of bad news. Although weight rescaling and the issues that arise have been well studied for two-level models, as pointed out by Carle (2009), "... a best practice for scaling weights across multiple levels has yet to be advanced." As such, `pwscale()` is currently supported only for two-level models. If you are fitting a higher-level model with survey data, you need to make sure your sampling weights are conditional on selection at the previous stage and not overall inclusion weights, because there is currently no rescaling option to fall back on if you do not.

Small-sample inference for fixed effects

Researchers are often interested in making inferences about fixed effects in a linear mixed-effects model. In the special case where the data are balanced and the mixed-effects model has a simple covariance structure, the sampling distributions of the statistics for testing hypotheses about fixed effects are known to follow an F distribution with specific denominator degrees of freedom (DDF) under the null hypothesis. For example, the test statistics for testing hypotheses about fixed effects in balanced split-plot designs and balanced repeated-measures designs have exact t or F distributions. In general, however, the null sampling distributions of test statistics for fixed effects are not known and can only be approximated in more complicated mixed-effects models.

For a large sample, the null sampling distributions of the test statistics can be approximated by a normal distribution for a one-hypothesis test and a χ^2 distribution for a multiple-hypotheses test. This is the default behavior of `mixed`. However, these large-sample approximations may not be appropriate in small samples, and t and F distributions may provide better approximations.

You can specify the `dfmethod()` option to request small-sample inference for fixed effects. `mixed` with the `dfmethod()` option uses a t distribution for one-hypothesis tests and an F distribution for multiple-hypotheses tests for inference about fixed effects. We use DF to refer to degrees of freedom of a t distribution, and we use DDF to refer to denominator degrees of freedom of an F distribution.

Researchers have proposed various approximations that use t and F distributions but differ in how respective DF and DDF are computed (for example, Khuri, Mathew, and Sinha [1998]; Brown and Prescott [2015]; Schluchter and Elashoff [1990]; Elston [1998]; Kackar and Harville [1984]; Giesbrecht and Burns [1985]; Fai and Cornelius [1996]; and Kenward and Roger [1997, 2009]). `mixed` provides five methods with the `dfmethod()` option for calculating the DF of a t distribution: `residual`, `repeated`, `anova`, `satterthwaite`, and `kroger`.

Residual DDF (DF). This method uses the residual degrees of freedom, $n - p$, as the DDF for all tests of fixed effects. For a linear model without random effects and with i.i.d errors, the distributions of the test statistics for testing the fixed effects are exact t or F distributions with the residual DF.

Repeated DDF (DF). This method partitions the residual degrees of freedom into the between-subject degrees of freedom and the within-subject degrees of freedom. This partitioning of the degrees of freedom arises from balanced repeated-measures ANOVA analysis. If levels of a fixed effect change within a subject, then the within-subject degrees of freedom is assigned to the fixed effect of interest; otherwise, the between-subject degrees of freedom is assigned to that fixed effect. Winer, Brown, and Michels (1991) showed that this method is appropriate only when the data are balanced and the correlation structure is assumed to be spherical. The repeated DDF method is supported only with two-level models. For DDF methods accounting for unbalanced repeated measures, see, for example, Schluchter and Elashoff (1990).

ANOVA DDF (DF). This method mimics the traditional ANOVA method. It determines the DDF for a fixed effect depending on whether the corresponding covariate is contained in any of the random-effects equations. If the covariate is contained in a random-effects equation, the DDF for the fixed effect is computed as the number of levels of the level variable from that equation minus one. If the covariate is specified in more than one random-effects equation, the DDF for the fixed effect is computed as the smallest number of levels of the level variables from those equations minus one and is a conservative estimate of the true DDF. If the covariate is specified only in the fixed-effects equation, the DDF is computed as $\nu_{\text{ddf}} = n - \text{rank}(\mathbf{X}, \mathbf{Z})$. This method leads to an exact sampling distribution of the test statistics only when random effects are balanced and the errors are i.i.d; see, for example, chapter 1.6 in Brown and Prescott (2015) for details.

Satterthwaite DDF (DF). This method performs a generalization of the Satterthwaite approximation based on Kackar and Harville (1984), Giesbrecht and Burns (1985), and Fai and Cornelius (1996). Giesbrecht and Burns (1985) developed a method of computing the DDF for a single-hypothesis test that is analogous to Satterthwaite's approximation of the degrees of freedom of a linear combination of ANOVA mean squares. For a multiple-hypotheses test, Fai and Cornelius (1996) proposed an extension of the Giesbrecht–Burns single-degree-of-freedom method. This method involves the spectral decomposition of the contrast matrix of the hypothesis test and repeated application of the single-degree-of-freedom t test. See *Denominator degrees of freedom* in *Methods and formulas* for more computational details.

Kenward–Roger DDF (DF). This method, developed by Kenward and Roger (1997), was designed to provide an approximation that improves the performance of hypothesis tests about fixed effects in small samples for complicated mixed-effects models and reproduces the exact inference available for simpler mixed-effects models. It provides adjusted test statistics, more appropriate DDFs for the approximate F distributions when exact inference is not available, and yields the exact t and F distributions when exact inference is available. This method first accounts for the small-sample bias and the variability of the estimated random effects to obtain an adjusted estimator of the fixed-effects covariance matrix. Then, it proposes an approximate F test based on a scaled Wald test statistic that uses the adjusted variance–covariance estimator. See *Denominator degrees of freedom* in *Methods and formulas* for more computational details.

Residual, repeated, and ANOVA are known as "exact" methods in the literature. These methods are suitable only when the sampling distributions of the test statistics are known to be t or F. This is usually only known for certain classes of linear mixed-effects models with simple covariance structures and when data are balanced. These methods are available with both ML and REML estimation.

Satterthwaite and Kenward–Roger are known as "approximation" methods in the literature. These methods are for unbalanced data and complicated covariance structures where the sampling distributions of test statistics are unknown and can only be approximated. Both methods are available only with REML estimation. For single-hypothesis tests, DDFs calculated with the Kenward–Roger method are the same as those calculated with the Satterthwaite method, but they differ for multiple-hypotheses tests. Although DDFs of the two methods are the same for single-hypothesis tests, the inference is not the same because the Kenward–Roger method uses bias-adjusted standard errors.

Except for the special cases for which the sampling distributions are known, there is no definitive recommendation for which approximation performs best. Schaalje, McBride, and Fellingham (2002) compared the Satterthwaite method with the Kenward–Roger method via simulation using different covariance structures and various sample sizes. They concluded that the Kenward–Roger method outperforms the Satterthwaite method in most situations. They recommend using the Satterthwaite method only when the covariance structure of the data is compound symmetry and the sample size is moderately large. The Kenward–Roger method, however, is not guaranteed to work well in all situations. For example, for more complicated covariance structures and very small-sample sizes, the Kenward–Roger method may produce inflated type I error rates. In conclusion, you should choose your DDF method carefully. See, for example, Schaalje, McBride, and Fellingham (2002), Chen and Wei (2003), Vallejo et al. (2004), and West, Welch, and Gałecki (2022) for a comparison of different approximations.

Both types of methods, exact and approximation, are available for single-hypothesis tests. For multiple-hypotheses tests, exact methods are available only if DDFs associated with fixed effects are the same for all tested covariates. See *Denominator degrees of freedom* in *Methods and formulas* for details.

▷ Example 15: Small-sample inference with a balanced repeated-measures design

Consider an example from Winer, Brown, and Michels (1991, table 4.3), also analyzed in example 15 of [R] **anova**, which reports the reaction time for five subjects who were tested with four drugs. The reaction time was recorded in the variable `score`. Assume that `person` is random (that is, we wish to infer to the larger population of possible subjects) and `drug` is fixed (that is, only four drugs are of interest). This is an example of a mixed-effects model with a simple covariance structure—a balanced repeated-measures design. The dataset contains only 20 observations, so we would like to account for the small sample in our analysis. Because this is a balanced repeated-measures design, we can use the repeated

method to obtain small-sample inference for fixed effects. We specify the `dfmethod(repeated)` option with `mixed`. We also request REML estimates by specifying the `reml` option to account for the small number of groups.

```
. use https://www.stata-press.com/data/r19/t43
(T4.3 -- Winer, Brown, Michels)

. mixed score i.drug || person:, reml dfmethod(repeated)

Performing EM optimization ...

Performing gradient-based optimization:
Iteration 0:  Log restricted-likelihood = -49.640099
Iteration 1:  Log restricted-likelihood = -49.640099

Computing standard errors ...

Computing degrees of freedom ...

Mixed-effects REML regression                   Number of obs     =         20
Group variable: person                          Number of groups  =          5
                                                Obs per group:
                                                              min =          4
                                                              avg =        4.0
                                                              max =          4
DF method: Repeated                             DF:           min =       4.00
                                                              avg =      10.00
                                                              max =      12.00
                                                F(3, 12.00)       =      24.76
Log restricted-likelihood = -49.640099          Prob > F          =     0.0000
```

score	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
drug						
2	-.8	1.939072	-0.41	0.687	-5.024874	3.424874
3	-10.8	1.939072	-5.57	0.000	-15.02487	-6.575126
4	5.6	1.939072	2.89	0.014	1.375126	9.824874
_cons	26.4	3.149604	8.38	0.001	17.6553	35.1447

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
person: Identity				
var(_cons)	40.20004	30.10272	9.264606	174.4319
var(Residual)	9.399997	3.837532	4.22305	20.92325

```
LR test vs. linear model: chibar2(01) = 15.03         Prob >= chibar2 = 0.0001
```

In the table for fixed effects, t statistics are reported instead of the default z statistics. We can compare our small-sample inference with the corresponding large-sample inference for fixed effects. We do not need to rerun the estimation command, because we can obtain large-sample results upon replay by default.

```
. mixed
Mixed-effects REML regression                   Number of obs     =         20
Group variable: person                          Number of groups  =          5
                                                Obs per group:
                                                              min =          4
                                                              avg =        4.0
                                                              max =          4
                                                Wald chi2(3)      =      74.28
Log restricted-likelihood = -49.640099          Prob > chi2       =     0.0000
```

score	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
drug						
2	-.8	1.939072	-0.41	0.680	-4.600511	3.000511
3	-10.8	1.939072	-5.57	0.000	-14.60051	-6.999489
4	5.6	1.939072	2.89	0.004	1.799489	9.400511
_cons	26.4	3.149604	8.38	0.000	20.22689	32.57311

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
person: Identity				
var(_cons)	40.20004	30.10272	9.264606	174.4319
var(Residual)	9.399997	3.837532	4.22305	20.92325

```
LR test vs. linear model: chibar2(01) = 15.03          Prob >= chibar2 = 0.0001
```

Comparing the above large-sample inference for fixed effects of `drug` with the small-sample inference, we see that the p-value for the level 4 of `drug` changes from 0.014 to 0.004.

If we wanted to replay our small-sample estimation results, we would type

```
. mixed, small
```
(*output omitted*)

The specified DF method and summaries of the coefficient-specific DFs are reported in the output header. We can use the `dftable()` option to display a fixed-effects table containing coefficient-specific DFs. `dftable(pvalue)` reports the fixed-effects table containing DFs, t statistics, and p-values, and `dftable(ci)` reports the fixed-effects table containing DFs and confidence intervals.

```
. mixed, dftable(pvalue) noretable
Mixed-effects REML regression                   Number of obs     =         20
Group variable: person                          Number of groups  =          5
                                                Obs per group:
                                                              min =          4
                                                              avg =        4.0
                                                              max =          4
DF method: Repeated                             DF:           min =       4.00
                                                              avg =      10.00
                                                              max =      12.00
                                                F(3, 12.00)       =      24.76
Log restricted-likelihood = -49.640099          Prob > F          =     0.0000
```

score	Coefficient	Std. err.	df	t	P>|t|
drug					
2	-.8	1.939072	12.0	-0.41	0.687
3	-10.8	1.939072	12.0	-5.57	0.000
4	5.6	1.939072	12.0	2.89	0.014
_cons	26.4	3.149604	4.0	8.38	0.001

Because levels of `drug` vary within `person`, the within-subject degrees of freedom, 12, are assigned to the coefficients for the levels of `drug`. The DF for the constant term is always the between-subject degrees of freedom, 4 in this example, because it is constant within random-effects levels.

The model F test is reported in the output header instead of the default χ^2 test. The F statistic for testing `drug` $= 0$ is 24.76 with DDF $= 12$, which agrees with the results of `anova, repeated()`:

```
. anova score person drug, repeated(drug)
                           Number of obs =         20    R-squared     =  0.9244
                           Root MSE      =    3.06594    Adj R-squared =  0.8803
```

Source	Partial SS	df	MS	F	Prob>F
Model	1379	7	197	20.96	0.0000
person	680.8	4	170.2	18.11	0.0001
drug	698.2	3	232.73333	24.76	0.0000
Residual	112.8	12	9.4		
Total	1491.8	19	78.515789		

```
Between-subjects error term:  person
                     Levels:  5         (4 df)
     Lowest b.s.e. variable:  person
Repeated variable: drug
                                   Huynh-Feldt epsilon        =  1.0789
                                   *Huynh-Feldt epsilon reset to 1.0000
                                   Greenhouse-Geisser epsilon =  0.6049
                                   Box's conservative epsilon =  0.3333
```

			Prob > F			
Source	df	F	Regular	H-F	G-G	Box
drug	3	24.76	0.0000	0.0000	0.0006	0.0076
Residual	12					

▷ Example 16: Small-sample inference with an unbalanced repeated-measures design

Consider West, Welch, and Gałecki's (2022) dental veneer dataset from example 7, containing two measurements on each tooth from multiple teeth per patient. Because of small-sample size, we would like to obtain small-sample inference for fixed effects.

Some patients are missing observations for some teeth:

```
. use https://www.stata-press.com/data/r19/veneer, clear
(Dental veneer data)
. table patient tooth
```

	Tooth number with patient						
	6	7	8	9	10	11	Total
Patient ID							
1	2	2	2	2	2	2	12
3	2	2	2	2	2	2	12
4	2	2	2	2	2	2	12
5		2	2	2	2		8
6	2	2	2	2	2	2	12
7	2	2	2	2	2	2	12
8	2	2	2	2	2	2	12
9	2	2					4
10	2	2	2	2	2	2	12
12		2	2	2	2		8
13					2		2
14			2		2		4
Total	16	20	20	18	22	14	110

The dataset is unbalanced; therefore, exact F tests for fixed effects are unavailable. As such, we will use the Satterthwaite and the Kenward–Roger approximation methods for calculating DF. Let's fit the model using the Kenward–Roger method first by specifying `dfmethod(kroger)`.

```
. mixed gcf followup base_gcf cda age || patient: followup, cov(un)
> || tooth:, reml nolog dfmethod(kroger)
Mixed-effects REML regression                   Number of obs =    110
```

Grouping information

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
patient	12	2	9.2	12
tooth	55	2	2.0	2

```
DF method: Kenward-Roger                        DF:       min =  10.41
                                                          avg =  28.96
                                                          max =  50.71
                                                F(4, 27.96)   =   1.47
Log restricted-likelihood = -420.92761          Prob > F      = 0.2370
```

gcf	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
followup	.3009815	1.938641	0.16	0.879	-3.96767	4.569633
base_gcf	-.0183127	.1466261	-0.12	0.901	-.3132419	.2766164
cda	-.329303	.5533506	-0.60	0.554	-1.440355	.7817493
age	-.5773932	.2350491	-2.46	0.033	-1.098324	-.056462
_cons	45.73862	13.21824	3.46	0.002	18.53866	72.93858

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
patient: Unstructured				
var(followup)	41.88772	18.79997	17.38009	100.9535
var(_cons)	524.9851	253.0205	204.1287	1350.175
cov(followup,_cons)	-140.4229	66.57623	-270.9099	-9.935904
tooth: Identity				
var(_cons)	47.45738	16.63034	23.8792	94.3165
var(Residual)	48.86704	10.50523	32.06479	74.47382

LR test vs. linear model: chi2(4) = 91.12 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

Compared with the p-values of the large-sample results from example 7, the p-values for age and _cons change substantially from 0.007 and 0.000 to 0.033 and 0.002, respectively. Note that for the Kenward–Roger method, not only the p-values and confidence intervals differ from those of the large-sample results but also the standard errors for the fixed effects differ. The standard errors differ because this method uses a bias-adjusted estimator of the variance–covariance matrix of fixed effects.

Now, let's fit the model using the Satterthwaite approximation:

```
. mixed gcf followup base_gcf cda age || patient: followup, cov(un)
> || tooth:, reml nolog dfmethod(satterthwaite)
Mixed-effects REML regression                   Number of obs =    110
```

Grouping information

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
patient	12	2	9.2	12
tooth	55	2	2.0	2

DF method: Satterthwaite DF: min = 10.41
avg = 28.96
max = 50.71
F(4, 16.49) = 1.87
Log restricted-likelihood = -420.92761 Prob > F = 0.1638

gcf	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
followup	.3009815	1.936863	0.16	0.879	-3.963754	4.565717
base_gcf	-.0183127	.1433094	-0.13	0.899	-.3065704	.269945
cda	-.329303	.5292525	-0.62	0.537	-1.39197	.7333636
age	-.5773932	.2139656	-2.70	0.022	-1.051598	-.1031885
_cons	45.73862	12.55497	3.64	0.001	19.90352	71.57372

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
patient: Unstructured				
var(followup)	41.88772	18.79997	17.38009	100.9535
var(_cons)	524.9851	253.0205	204.1287	1350.175
cov(followup,_cons)	-140.4229	66.57623	-270.9099	-9.935904
tooth: Identity				
var(_cons)	47.45738	16.63034	23.8792	94.3165
var(Residual)	48.86704	10.50523	32.06479	74.47382

LR test vs. linear model: chi2(4) = 91.12 Prob > chi2 = 0.0000

Note: LR test is conservative and provided only for reference.

Using the Satterthwaite method, we see that the p-value for age is 0.022 and for _cons is 0.001 and that these are again substantially different from their large-sample counterparts. On the other hand, unlike the standard errors for the Kenward–Roger method, those for the Satterthwaite method are the same as the standard errors from the large-sample results.

Looking at the DF summaries in the output header of the two methods, we notice that they are exactly the same. This is because DFs for fixed effects obtained using the Kenward–Roger and Satterthwaite methods are the same for single-hypothesis tests. (You can verify this by specifying, for example, dftable(pvalue) with the above commands or by using estat df; see [ME] **estat df**.) The DDFs differ, however, for multiple-hypotheses tests. For example, DDF computed for the overall model test using dfmethod(satterthwaite) (16.49) is smaller than that computed using dfmethod(kroger) (27.96).

There are no general guidelines to which method should be preferred, but according to Schaalje, McBride, and Fellingham (2002), the Kenward–Roger method outperforms the Satterthwaite method when the variance–covariance structure of the random effects is unstructured, which is the case in our example.

◁

Determining which DDF method is best is a difficult task and may often need simulation. The choice of the method depends on the specified covariance structure, sample size, and imbalance of the data. No method applies to all situations; thus you should use caution when choosing among methods.

Stored results

mixed stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_f)	number of fixed-effects parameters
e(k_r)	number of random-effects parameters
e(k_rs)	number of variances
e(k_rc)	number of covariances
e(k_res)	number of residual-error parameters
e(N_clust)	number of clusters
e(nrgroups)	number of residual-error by() groups
e(ar_p)	AR order of residual errors, if specified
e(ma_q)	MA order of residual errors, if specified
e(res_order)	order of residual-error structure, if appropriate
e(df_m)	model degrees of freedom

e(small)	1 if dfmethod() option specified, 0 otherwise
e(spnugget)	1 if nugget suboption is specified within residuals(), 0 otherwise
e(F)	overall F test statistic when dfmethod() is specified
e(ddf_m)	model DDF
e(df_max)	maximum DF
e(df_avg)	average DF
e(df_min)	minimum DF
e(ll)	log (restricted) likelihood
e(chi2)	χ^2
e(p)	p-value for model test
e(ll_c)	log likelihood, comparison model
e(chi2_c)	χ^2, comparison test
e(df_c)	degrees of freedom, comparison test
e(p_c)	p-value for comparison test
e(rank)	rank of e(V)
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	mixed
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(wtype)	weight type (first-level weights)
e(wexp)	weight expression (first-level weights)
e(fweight$k$)	fweight variable for kth highest level, if specified
e(pweight$k$)	pweight variable for kth highest level, if specified
e(ivars)	grouping variables
e(title)	title in estimation output
e(redim)	random-effects dimensions
e(vartypes)	variance-structure types
e(revars)	random-effects covariates
e(resopt)	residuals() specification, as typed
e(rstructure)	residual-error structure
e(rstructlab)	residual-error structure output label
e(rbyvar)	residual-error by() variable, if specified
e(rglabels)	residual-error by() groups labels
e(pwscale)	sampling-weight scaling method
e(timevar)	residual-error t() variable, if specified
e(spcoordvars)	residual-error coordinates() variables, if specified
e(spmetric)	residual-error metric() type, if specified
e(dfmethod)	DF method specified in dfmethod()
e(dftitle)	title for DF method
e(chi2type)	Wald; type of model χ^2 test
e(clustvar)	name of cluster variable
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(method)	ML or REML
e(opt)	type of optimization
e(optmetric)	matsqrt or matlog; random-effects matrix parameterization
e(emonly)	emonly, if specified
e(ml_method)	type of ml method
e(technique)	maximization technique
e(datasignature)	the checksum
e(datasignaturevars)	variables used in calculation of checksum
e(properties)	b V
e(estat_cmd)	program used to implement estat
e(predict)	program used to implement predict
e(marginswtype)	weight type for margins

e(marginswexp)	weight expression for margins
e(asbalanced)	factor variables fvset as asbalanced
e(asobserved)	factor variables fvset as asobserved
Matrices	
e(b)	coefficient vector
e(Cns)	constraints matrix
e(N_g)	group counts
e(g_min)	group-size minimums
e(g_avg)	group-size averages
e(g_max)	group-size maximums
e(tmap)	ID mapping for unstructured residual errors
e(V)	variance–covariance matrix of the estimators
e(V_modelbased)	model-based variance
e(df)	parameter-specific DF for fixed effects
e(V_df)	variance–covariance matrix of the estimators when dfmethod(kroger) is specified
Functions	
e(sample)	marks estimation sample

In addition to the above, the following is stored in r():

Matrices	
r(table)	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in r() are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

Methods and formulas are presented under the following headings:

Estimation using ML and REML
Denominator degrees of freedom
Residual DDF
Repeated DDF
ANOVA DDF
Satterthwaite DDF
Kenward–Roger DDF
Fixed-effects constraints

Estimation using ML and REML

As given by (1), in the absence of weights we have the linear mixed model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}$$

where $\mathbf{y}$ is the $n \times 1$ vector of responses, $\mathbf{X}$ is an $n \times p$ design/covariate matrix for the fixed effects $\boldsymbol{\beta}$, and $\mathbf{Z}$ is the $n \times q$ design/covariate matrix for the random effects $\mathbf{u}$. The $n \times 1$ vector of errors $\boldsymbol{\epsilon}$ is for now assumed to be multivariate normal with mean 0 and variance matrix $\sigma^2_\epsilon \mathbf{I}_n$. We also assume that $\mathbf{u}$ has variance–covariance matrix $\mathbf{G}$ and that $\mathbf{u}$ is orthogonal to $\boldsymbol{\epsilon}$ so that

$$\mathrm{Var}\begin{bmatrix}\mathbf{u}\\ \boldsymbol{\epsilon}\end{bmatrix} = \begin{bmatrix}\mathbf{G} & \mathbf{0}\\ \mathbf{0} & \sigma^2_\epsilon \mathbf{I}_n\end{bmatrix}$$

Considering the combined error term $\mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}$, we see that $\mathbf{y}$ is multivariate normal with mean $\mathbf{X}\boldsymbol{\beta}$ and $n \times n$ variance–covariance matrix

$$\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \sigma^2_\epsilon \mathbf{I}_n$$

Defining $\boldsymbol{\theta}$ as the vector of unique elements of $\mathbf{G}$ results in the log likelihood

$$L(\boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2_\epsilon) = -\frac{1}{2}\left\{n\log(2\pi) + \log|\mathbf{V}| + (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right\} \tag{9}$$

which is maximized as a function of $\boldsymbol{\beta}$, $\boldsymbol{\theta}$, and σ^2_ϵ. As explained in chapter 6 of Searle, Casella, and McCulloch (1992), considering instead the likelihood of a set of linear contrasts $\mathbf{Ky}$ that do not depend on $\boldsymbol{\beta}$ results in the restricted log likelihood

$$L_R(\boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2_\epsilon) = L(\boldsymbol{\beta}, \boldsymbol{\theta}, \sigma^2_\epsilon) - \frac{1}{2}\log\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| \tag{10}$$

Given the high dimension of $\mathbf{V}$, however, the log-likelihood and restricted log-likelihood criteria are not usually computed by brute-force application of the above expressions. Instead, you can simplify the problem by subdividing the data into independent clusters (and subclusters if possible) and using matrix decomposition methods on the smaller matrices that result from treating each cluster one at a time.

Consider the two-level model described previously in (2),

$$\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j + \boldsymbol{\epsilon}_j$$

for $j = 1, \ldots, M$ clusters with cluster j containing n_j observations, with $\text{Var}(\mathbf{u}_j) = \boldsymbol{\Sigma}$, a $q \times q$ matrix.

Efficient methods for computing (9) and (10) are given in chapter 2 of Pinheiro and Bates (2000). Namely, for the two-level model, define $\boldsymbol{\Delta}$ to be the Cholesky factor of $\sigma^2_\epsilon\boldsymbol{\Sigma}^{-1}$, such that $\sigma^2_\epsilon\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Delta}'\boldsymbol{\Delta}$. For $j = 1, \ldots, M$, decompose

$$\begin{bmatrix}\mathbf{Z}_j \\ \boldsymbol{\Delta}\end{bmatrix} = \mathbf{Q}_j\begin{bmatrix}\mathbf{R}_{11j} \\ \mathbf{0}\end{bmatrix}$$

by using an orthogonal-triangular (QR) decomposition, with $\mathbf{Q}_j$ a $(n_j + q)$-square matrix and $\mathbf{R}_{11j}$ a q-square matrix. We then apply $\mathbf{Q}_j$ as follows:

$$\begin{bmatrix}\mathbf{R}_{10j} \\ \mathbf{R}_{00j}\end{bmatrix} = \mathbf{Q}_j'\begin{bmatrix}\mathbf{X}_j \\ \mathbf{0}\end{bmatrix}; \qquad \begin{bmatrix}\mathbf{c}_{1j} \\ \mathbf{c}_{0j}\end{bmatrix} = \mathbf{Q}_j'\begin{bmatrix}\mathbf{y}_j \\ \mathbf{0}\end{bmatrix}$$

Stack the $\mathbf{R}_{00j}$ and $\mathbf{c}_{0j}$ matrices, and perform the additional QR decomposition

$$\begin{bmatrix}\mathbf{R}_{001} & \mathbf{c}_{01} \\ \vdots & \vdots \\ \mathbf{R}_{00M} & \mathbf{c}_{0M}\end{bmatrix} = \mathbf{Q}_0\begin{bmatrix}\mathbf{R}_{00} & \mathbf{c}_0 \\ \mathbf{0} & \mathbf{c}_1\end{bmatrix}$$

Pinheiro and Bates (2000) show that ML estimates of $\boldsymbol{\beta}$, σ^2_ϵ, and $\boldsymbol{\Delta}$ (the unique elements of $\boldsymbol{\Delta}$, that is) are obtained by maximizing the profile log likelihood (profiled in $\boldsymbol{\Delta}$)

$$L(\boldsymbol{\Delta}) = \frac{n}{2}\left\{\log n - \log(2\pi) - 1\right\} - n\log||\mathbf{c}_1|| + \sum_{j=1}^{M}\log\left|\frac{\det(\boldsymbol{\Delta})}{\det(\mathbf{R}_{11j})}\right| \tag{11}$$

where $||\cdot||$ denotes the 2-norm. Following this maximization with

$$\widehat{\boldsymbol{\beta}} = \mathbf{R}_{00}^{-1}\mathbf{c}_0; \quad \hat{\sigma}^2_\epsilon = n^{-1}||\mathbf{c}_1||^2 \tag{12}$$

REML estimates are obtained by maximizing

$$L_R(\boldsymbol{\Delta}) = \frac{n-p}{2}\{\log(n-p) - \log(2\pi) - 1\} - (n-p)\log||\mathbf{c}_1|| - \log|\det(\mathbf{R}_{00})| + \sum_{j=1}^{M}\log\left|\frac{\det(\boldsymbol{\Delta})}{\det(\mathbf{R}_{11j})}\right| \tag{13}$$

followed by

$$\widehat{\boldsymbol{\beta}} = \mathbf{R}_{00}^{-1}\mathbf{c}_0; \quad \hat{\sigma}_\epsilon^2 = (n-p)^{-1}||\mathbf{c}_1||^2$$

For numerical stability, maximization of (11) and (13) is not performed with respect to the unique elements of $\boldsymbol{\Delta}$ but instead with respect to the unique elements of the matrix square root (or matrix logarithm if the `matlog` option is specified) of $\boldsymbol{\Sigma}/\sigma_\epsilon^2$; define $\boldsymbol{\gamma}$ to be the vector containing these elements.

Once maximization with respect to $\boldsymbol{\gamma}$ is completed, $(\boldsymbol{\gamma}, \sigma_\epsilon^2)$ is reparameterized to $\{\boldsymbol{\alpha},\ \log(\sigma_\epsilon)\}$, where $\boldsymbol{\alpha}$ is a vector containing the unique elements of $\boldsymbol{\Sigma}$, expressed as logarithms of standard deviations for the diagonal elements and hyperbolic arctangents of the correlations for off-diagonal elements. This last step is necessary 1) to obtain a joint variance–covariance estimate of the elements of $\boldsymbol{\Sigma}$ and σ_ϵ^2; 2) to obtain a parameterization under which parameter estimates can be interpreted individually, rather than as elements of a matrix square root (or logarithm); and 3) to parameterize these elements such that their ranges each encompass the entire real line.

Obtaining a joint variance–covariance matrix for the estimated $\{\boldsymbol{\alpha},\ \log(\sigma_\epsilon)\}$ requires the evaluation of the log likelihood (or log-restricted likelihood) with only $\boldsymbol{\beta}$ profiled out. For ML, we have

$$L^*\{\boldsymbol{\alpha},\ \log(\sigma_\epsilon)\} = L\{\boldsymbol{\Delta}(\boldsymbol{\alpha}, \sigma_\epsilon^2), \sigma_\epsilon^2\} = -\frac{n}{2}\log(2\pi\sigma_\epsilon^2) - \frac{||\mathbf{c}_1||^2}{2\sigma_\epsilon^2} + \sum_{j=1}^{M}\log\left|\frac{\det(\boldsymbol{\Delta})}{\det(\mathbf{R}_{11j})}\right|$$

with the analogous expression for REML.

The variance–covariance matrix of $\widehat{\boldsymbol{\beta}}$ is estimated as

$$\widehat{\mathrm{Var}}(\widehat{\boldsymbol{\beta}}) = \hat{\sigma}_\epsilon^2\mathbf{R}_{00}^{-1}\left(\mathbf{R}_{00}^{-1}\right)'$$

but this does not mean that $\widehat{\mathrm{Var}}(\widehat{\boldsymbol{\beta}})$ is identical under both ML and REML because $\mathbf{R}_{00}$ depends on $\boldsymbol{\Delta}$. Because $\widehat{\boldsymbol{\beta}}$ is asymptotically uncorrelated with $\{\widehat{\boldsymbol{\alpha}},\ \log(\hat{\sigma}_\epsilon)\}$, the covariance of $\widehat{\boldsymbol{\beta}}$ with the other estimated parameters is treated as 0.

Parameter estimates are stored in `e(b)` as $\{\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\alpha}},\ \log(\hat{\sigma}_\epsilon)\}$, with the corresponding (block-diagonal) variance–covariance matrix stored in `e(V)`. Parameter estimates can be displayed in this metric by specifying the `estmetric` option. However, in `mixed` output, variance components are most often displayed either as variances and covariances or as standard deviations and correlations.

EM iterations are derived by considering the $\mathbf{u}_j$ in (2) as missing data. Here we describe the procedure for maximizing the log likelihood via EM; the procedure for maximizing the restricted log likelihood is similar. The log likelihood for the full data $(\mathbf{y}, \mathbf{u})$ is

$$L_F(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \sigma_\epsilon^2) = \sum_{j=1}^{M}\left\{\log f_1(\mathbf{y}_j|\mathbf{u}_j, \boldsymbol{\beta}, \sigma_\epsilon^2) + \log f_2(\mathbf{u}_j|\boldsymbol{\Sigma})\right\}$$

where $f_1(\cdot)$ is the density function for multivariate normal with mean $\mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{u}_j$ and variance $\sigma_\epsilon^2\mathbf{I}_{n_j}$, and $f_2(\cdot)$ is the density for multivariate normal with mean $\mathbf{0}$ and $q \times q$ covariance matrix $\boldsymbol{\Sigma}$. As before, we can profile $\boldsymbol{\beta}$ and σ_ϵ^2 out of the optimization, yielding the following EM iterative procedure:

1. For the current iterated value of $\boldsymbol{\Sigma}^{(t)}$, fix $\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{\beta}}(\boldsymbol{\Sigma}^{(t)})$ and $\hat{\sigma}^2_\epsilon = \hat{\sigma}^2_\epsilon(\boldsymbol{\Sigma}^{(t)})$ according to (12).
2. Expectation step: Calculate

$$\begin{aligned} D(\boldsymbol{\Sigma}) &\equiv E\left\{L_F(\widehat{\boldsymbol{\beta}}, \boldsymbol{\Sigma}, \hat{\sigma}^2_\epsilon)|\mathbf{y}\right\} \\ &= C - \frac{M}{2}\log\det(\boldsymbol{\Sigma}) - \frac{1}{2}\sum_{j=1}^{M} E\left(\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j|\mathbf{y}\right) \end{aligned}$$

   where C is a constant that does not depend on $\boldsymbol{\Sigma}$, and the expected value of the quadratic form $\mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j$ is taken with respect to the conditional density $f(\mathbf{u}_j|\mathbf{y}, \widehat{\boldsymbol{\beta}}, \boldsymbol{\Sigma}^{(t)}, \hat{\sigma}^2_\epsilon)$.
3. Maximization step: Maximize $D(\boldsymbol{\Sigma})$ to produce $\boldsymbol{\Sigma}^{(t+1)}$.

For general, symmetric $\boldsymbol{\Sigma}$, the maximizer of $D(\boldsymbol{\Sigma})$ can be derived explicitly, making EM iterations quite fast.

For general, within-group error covariance structure,

$$\text{Var}(\boldsymbol{\epsilon}_j) = \sigma^2_\epsilon \boldsymbol{\Lambda}_j$$

where the subscript j merely represents that $\boldsymbol{\epsilon}_j$ and $\boldsymbol{\Lambda}_j$ vary in dimension in unbalanced data, the data are first transformed according to

$$\mathbf{y}^*_j = \widehat{\boldsymbol{\Lambda}}_j^{-1/2}\mathbf{y}_j; \qquad \mathbf{X}^*_j = \widehat{\boldsymbol{\Lambda}}_j^{-1/2}\mathbf{X}_j; \qquad \mathbf{Z}^*_j = \widehat{\boldsymbol{\Lambda}}_j^{-1/2}\mathbf{Z}_j;$$

and the likelihood-evaluation techniques described above are applied to $\mathbf{y}^*_j$, $\mathbf{X}^*_j$, and $\mathbf{Z}^*_j$ instead. The unique elements of $\boldsymbol{\Lambda}$, $\boldsymbol{\rho}$, are estimated along with the fixed effects and variance components. Because σ^2_ϵ is always estimated and multiplies the entire $\boldsymbol{\Lambda}_j$ matrix, $\hat{\boldsymbol{\rho}}$ is parameterized to take this into account.

In the presence of sampling weights, following Rabe-Hesketh and Skrondal (2006), the weighted log pseudolikelihood for a two-level model is given as

$$L(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \sigma^2_\epsilon) = \sum_{j=1}^{M} w_j \log\left[\int \exp\left\{\sum_{i=1}^{n_j} w_{i|j}\log f_1(y_{ij}|\mathbf{u}_j, \boldsymbol{\beta}, \sigma^2_\epsilon)\right\} f_2(\mathbf{u}_j|\boldsymbol{\Sigma}) d\mathbf{u}_j\right] \tag{14}$$

where w_j is the inverse of the probability of selection for the jth cluster, $w_{i|j}$ is the inverse of the conditional probability of selection of individual i given the selection of cluster j, and $f_1(\cdot)$ and $f_2(\cdot)$ are the multivariate normal densities previously defined.

Weighted estimation is achieved through incorporating w_j and $w_{i|j}$ into the matrix decomposition methods detailed above to reflect replicated clusters for w_j and replicated observations within clusters for $w_{i|j}$. Because this estimation is based on replicated clusters and observations, frequency weights are handled similarly.

Rescaling of sampling weights can take one of three available forms:

Under `pwscale(size)`,

$$w^*_{i|j} = n_j w_{i|j}\left\{\sum_{i=1}^{n_j} w_{i|j}\right\}^{-1}$$

Under `pwscale(effective)`,

$$w^*_{i|j} = w_{i|j}\left\{\sum_{i=1}^{n_j} w_{i|j}\right\}\left\{\sum_{i=1}^{n_j} w^2_{i|j}\right\}^{-1}$$

Under both the above, w_j remains unchanged. For method `pwscale(gk)`, however, both weights are modified:

$$w_j^* = n_j^{-1} \sum_{i=1}^{n_j} w_{i|j} w_j \qquad w_{i|j}^* = 1$$

Under ML estimation, robust standard errors are obtained in the usual way (see [P] **_robust**) with the one distinction being that in multilevel models, robust variances are, at a minimum, clustered at the highest level. This is because given the form of the log likelihood, scores aggregate at the top-level clusters. For a two-level model, scores are obtained as the partial derivatives of $L_j(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \sigma_\epsilon^2)$ with respect to $\{\boldsymbol{\beta}, \boldsymbol{\alpha}, \log(\sigma_\epsilon)\}$, where L_j is the log likelihood for cluster j and $L = \sum_{j=1}^{M} L_j$. Robust variances are not supported under REML estimation because the form of the log restricted likelihood does not lend itself to separation by highest-level clusters.

EM iterations always assume equal weighting and an independent, homoskedastic error structure. As such, with weighted data or when error structures are more complex, EM is used only to obtain starting values.

For extensions to models with three or more levels, see Bates and Pinheiro (1998) and Rabe-Hesketh and Skrondal (2006).

Denominator degrees of freedom

When the `dfmethod()` option is specified, `mixed` uses a t distribution with ν_{ddf} degrees of freedom to perform single-hypothesis tests for fixed effects H_0: $\beta_i = 0$ for $i = 1, 2, \ldots, p$ or an F distribution with model numerator degrees of freedom and ν_{ddf_m} DDF for a model (joint) test of all coefficients (except the constant) being equal to zero. Denominator degrees of freedom ν_{ddf} and ν_{ddf_m} are computed according to the specified DDF method.

Residual DDF

This method uses the residual degrees of freedom as the DDF, $\nu_{\text{ddf}} = n - p$, where n is the total number of observations, and p is the rank of the design matrix $\mathbf{X}$.

Repeated DDF

This method partitions the residual degrees of freedom into the between-subject degrees of freedom and the within-subject degrees of freedom. This partitioning of the degrees of freedom arises from balanced repeated-measures ANOVA analysis. If levels of a fixed effect change within a subject, then the within-subject degrees of freedom is assigned to the fixed effect of interest; otherwise, the between-subject degrees of freedom is assigned to that fixed effect. See Schluchter and Elashoff (1990) for more computational details and, specifically, for the expressions of between-subject and within-subject degrees of freedom.

ANOVA DDF

This method determines the DDF for a fixed effect depending on whether the corresponding covariate is contained in any of the random-effects equations. If the covariate is contained in a random-effects equation, the DDF ν_{ddf} for the fixed effect is computed as the number of levels of the level variable from that equation minus one. If the covariate is specified in more than one random-effects equation, the DDF

ν_{ddf} for the fixed effect is computed as the smallest number of levels of the level variables from those equations minus one and is a conservative estimate of the true DDF. If the covariate is specified only in the fixed-effects equation, the DDF is computed as $\nu_{\text{ddf}} = n - \text{rank}(\mathbf{X}, \mathbf{Z})$.

For example, suppose we have the following mixed model,

```
mixed y A B C || D: A || E: A B
```

where `A`, `B`, and `C` are fixed effects, and `D` and `E` are nested random effects. For the fixed effect `A`, ν_{ddf} is the smaller number of levels of variables `D` and `E` minus one because `A` is included in random-effects equations at both levels `D` and `E`. For the fixed effect `B`, ν_{ddf} is the number of levels of level variable `E` minus one because `B` is included in the random-effects equation at the level `E`. For the fixed effect `C`, $\nu_{\text{ddf}} = n - \text{rank}(\mathbf{X}, \mathbf{Z})$ because `C` is not included in any of the random-effects equations.

For the three methods above, the DDF for a model test of H_0: $\boldsymbol{\beta} = \mathbf{0}$ is computed as follows. If all corresponding single-hypothesis tests H_0: $\beta_i = 0$ have the same DDF ν_{ddf}, then model DDF $\nu_{\text{ddf}_m} = \nu_{\text{ddf}}$. If the single-hypothesis DDF differs, then ν_{ddf_m} is not defined, and the large-sample χ^2 test is reported instead of the F test.

To provide formulas for the Satterthwaite and Kenward–Roger methods, consider a general linear-hypotheses test of fixed effects H_0: $\mathbf{C}'\boldsymbol{\beta} = \mathbf{b}$ with a $p \times l$ matrix of linear hypotheses $\mathbf{C}$ of rank l.

The variance–covariance matrix of $\mathbf{y}$ is $\text{Var}(\mathbf{y}) = \mathbf{V} = \mathbf{ZGZ}' + \mathbf{R} = \mathbf{V}(\boldsymbol{\sigma})$ and can be viewed as a function of variance components $\boldsymbol{\sigma}$ $(r \times 1)$. Suppose that the first two partial derivatives of $\mathbf{V}(\boldsymbol{\sigma})$ with respect to $\boldsymbol{\sigma}$ exist.

Let $\widehat{\boldsymbol{\sigma}}$ be the REML estimator of $\boldsymbol{\sigma}$. Then, the REML estimator of the fixed effects $\boldsymbol{\beta}$ is the generalized least-squares estimator

$$\widehat{\boldsymbol{\beta}} = \left\{\mathbf{X}'\mathbf{V}^{-1}(\widehat{\boldsymbol{\sigma}})\mathbf{X}\right\}^{-1}\mathbf{X}'\mathbf{V}^{-1}(\widehat{\boldsymbol{\sigma}})\mathbf{Y}$$

where $\widehat{\text{Var}}(\widehat{\boldsymbol{\beta}}) = \widehat{\boldsymbol{\Phi}} = \boldsymbol{\Phi}(\widehat{\boldsymbol{\sigma}}) = \left\{\mathbf{X}'\mathbf{V}^{-1}(\widehat{\boldsymbol{\sigma}})\mathbf{X}\right\}^{-1}$ is the conventional estimator of the variance–covariance matrix of the fixed effects $\widehat{\boldsymbol{\beta}}$, and $\mathbf{V}(\widehat{\boldsymbol{\sigma}})$ is the estimator of the covariance matrix of $\mathbf{y}$.

Under the null H_0: $\mathbf{C}'\boldsymbol{\beta} = \mathbf{b}$, the F test statistic is

$$F = \frac{1}{l}(\mathbf{C}'\widehat{\boldsymbol{\beta}} - \mathbf{b})'(\mathbf{C}'\widehat{\boldsymbol{\Phi}}\mathbf{C})^{-1}(\mathbf{C}'\widehat{\boldsymbol{\beta}} - \mathbf{b})$$

and it has an F distribution with l numerator and ν_{ddf_C} DDF.

Satterthwaite DDF

This method is derived from the DDF formula of the original approximation attributable to Satterthwaite (1946):

$$\text{ddf} = \frac{2(\mathbf{C}'\widehat{\boldsymbol{\Phi}}\mathbf{C})^2}{\text{Var}(\mathbf{C}'\widehat{\boldsymbol{\Phi}}\mathbf{C})}$$

For a single-hypothesis test of H_0: $\mathbf{c}'\boldsymbol{\beta} = \mathbf{b}$, where $\mathbf{c}$ and $\mathbf{b}$ are vectors of known constants, Giesbrecht and Burns (1985) proposed using

$$\nu_{\text{ddf}} = \frac{2(\mathbf{c}'\widehat{\boldsymbol{\Phi}}\mathbf{c})^2}{\text{Var}(\mathbf{c}'\widehat{\boldsymbol{\Phi}}\mathbf{c})} = \frac{2(\mathbf{c}'\widehat{\boldsymbol{\Phi}}\mathbf{c})^2}{\mathbf{d}'\mathbf{W}\mathbf{d}} \tag{15}$$

where $\mathbf{d}$ is a vector of partial derivatives of $\mathbf{c}'\boldsymbol{\Phi}(\boldsymbol{\sigma})\mathbf{c}$ with respect to $\boldsymbol{\sigma}$ evaluated at $\widehat{\boldsymbol{\sigma}}$, and $\widehat{\mathrm{Var}}(\widehat{\boldsymbol{\sigma}}) = \mathbf{W}$ is the estimator of the variance–covariance matrix of $\widehat{\boldsymbol{\sigma}}$ computed based on the expected information matrix $\mathbf{I}_\mathbf{E}$ in (17) or on the observed information matrix if the `oim` suboption of `dfmethod()` is specified.

For a multiple-hypotheses test (when the rank of $\mathbf{C}$ is greater than 1), Fai and Cornelius (1996) proposed an extension of the Giesbrecht–Burns single-degree-of-freedom method. Their method involves the spectral decomposition $\mathbf{C}'\widehat{\boldsymbol{\Phi}}\mathbf{C} = \mathbf{P}'\mathbf{D}\mathbf{P}$, where $\mathbf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_l)$ is an orthogonal matrix of eigenvectors, and $\mathbf{D} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_l)$ is a diagonal matrix of the corresponding eigenvalues. Using this decomposition, we can write the F-test statistic as a sum of l independent approximate t random variates, $F = Q/l$ with

$$Q = \sum_{k=1}^{l} \frac{\{\mathbf{p}_k'(\mathbf{C}'\widehat{\boldsymbol{\beta}} - \mathbf{b})\}^2}{\lambda_k} = \sum_{k=1}^{l} t_{v_k}^2$$

where v_k is computed using (15). Because t_{v_k}s are independent and have approximate t distributions with v_k degrees of freedom,

$$E(Q) = \sum_{k=1}^{l} \frac{v_k}{v_k - 2} I(v_k > 2)$$

Then, the DDF for a multiple-hypotheses test can be approximately written as

$$\nu_{\mathrm{ddf}_C} = \frac{2E(Q)}{E(Q) - l}$$

For more computational details of the Satterthwaite method, see Fai and Cornelius (1996).

Kenward–Roger DDF

This method was developed by Kenward and Roger (1997). It is based on adjusting the conventional variance–covariance estimator of fixed effects $\widehat{\boldsymbol{\Phi}}$ for small-sample bias and introducing a scaled F test that improves the small-sample performance of the conventional F test of fixed effects.

Kenward and Roger (1997) propose the adjusted estimator,

$$\widehat{\boldsymbol{\Phi}}_A = \widehat{\boldsymbol{\Phi}} + 2\widehat{\boldsymbol{\Phi}} \left\{ \sum_{i=1}^{r} \sum_{j=1}^{r} W_{ij} (\mathbf{Q}_{ij} - \mathbf{P}_i \widehat{\boldsymbol{\Phi}} \mathbf{P}_j - \frac{1}{4}\mathbf{R}_{ij}) \right\} \widehat{\boldsymbol{\Phi}} \tag{16}$$

where $\mathbf{P}_i = \mathbf{X}'\{\partial \mathbf{V}^{-1}(\boldsymbol{\sigma})/\partial\sigma_i\}\mathbf{X}$, $\mathbf{Q}_{ij} = \mathbf{X}'\{\partial \mathbf{V}^{-1}(\boldsymbol{\sigma})/\partial\sigma_i\}\mathbf{V}(\boldsymbol{\sigma})\{\partial \mathbf{V}^{-1}(\boldsymbol{\sigma})/\partial\sigma_j\}\mathbf{X}$, and $\mathbf{R}_{ij} = \mathbf{X}'\mathbf{V}^{-1}(\boldsymbol{\sigma})\{\partial^2 \mathbf{V}(\boldsymbol{\sigma})/\partial\sigma_i\partial\sigma_j\}\mathbf{V}^{-1}(\boldsymbol{\sigma})\mathbf{X}$ evaluated at $\widehat{\boldsymbol{\sigma}}$ and W_{ij} is the (i, j)th element of $\mathbf{W}$, the estimator of the variance–covariance matrix of $\widehat{\boldsymbol{\sigma}}$ computed from the inverse of the expected information matrix $\mathbf{I}_E$, where the element I_E^{ij} of $\mathbf{I}_E$ is defined as

$$2I_E^{ij} = \mathrm{tr}\left(\frac{\partial \mathbf{V}^{-1}}{\partial\sigma_i} \mathbf{V} \frac{\partial \mathbf{V}^{-1}}{\partial\sigma_j} \mathbf{V} \right) - \mathrm{tr}(2\boldsymbol{\Phi}\mathbf{Q}_{ij} - \boldsymbol{\Phi}\mathbf{P}_i\boldsymbol{\Phi}\mathbf{P}_j) \tag{17}$$

Alternatively, you can use the observed information matrix as $\mathbf{W}$ by specifying the `oim` suboption of `dfmethod()`.

All terms in (16), except those involving $\mathbf{R}_{ij}$, are invariant under reparameterization of the covariance structures. Also, the second derivative requires more computational resources and may not be numerically stable. For these reasons, the $\mathbf{R}_{ij}$ terms are ignored in the computation of $\widehat{\boldsymbol{\Phi}}_A$ in (16).

For multiple-hypotheses testing, Kenward and Roger (1997) propose the scaled F-test statistic, which under the null hypothesis can be written as

$$F_{\text{KR}} = \frac{\lambda}{l}(\mathbf{C}'\widehat{\boldsymbol{\beta}} - \mathbf{b})'(\mathbf{C}'\widehat{\boldsymbol{\Phi}}_A\mathbf{C})^{-1}(\mathbf{C}'\widehat{\boldsymbol{\beta}} - \mathbf{b})$$

and has an F distribution with l numerator and ν_{ddf_C} DDF. The scale factor $\lambda = \nu_{\text{ddf}_C}/(l - 1 + \nu_{\text{ddf}_C})$.

The DDF ν_{ddf_C} and λ are approximated as

$$\nu_{\text{ddf}_C} = 4 + \frac{l+2}{l \times \rho - 1} \qquad \text{and} \qquad \lambda = \frac{\nu_{\text{ddf}_C}}{E^*(\nu_{\text{ddf}_C} - 2)}$$

where $\rho = V^*/2(E^*)^2$ and E^* and V^* are the respective approximate mean and variance of the F_{KR} statistic; see Kenward and Roger (1997, 987) for expressions for E^* and V^*.

Fixed-effects constraints

Fixed-effects constraints $\mathbf{R}\boldsymbol{\beta} = \mathbf{r}$ are computed by first generating the $\mathbf{T}$ and $\mathbf{a}$ matrices via the eigenvalue decomposition described in [P] **makecns**. The fixed-effects model matrix is adjusted by $\mathbf{X}_c = \mathbf{XT}$ and the dependent variable by $\mathbf{y}_c = \mathbf{y} - \mathbf{Xa}'$. Computations then proceed with unconstrained optimization using $\mathbf{X}_c$ and $\mathbf{y}_c$. On convergence, we solve for the reduced-form fixed effects $\widehat{\boldsymbol{\beta}}_c$ and then solve for the constrained fixed effects $\widehat{\boldsymbol{\beta}} = \mathbf{T}\widehat{\boldsymbol{\beta}}_c + \mathbf{a}'$. (Here, $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{\beta}}_c$ correspond to $\mathbf{b}'$ and $\mathbf{b}_c'$ in [P] **makecns**.)

Acknowledgments

We thank Badi Baltagi of the Department of Economics at Syracuse University and Ray Carroll of the Department of Statistics at Texas A&M University for each providing us with a dataset used in this entry.

We also thank Mike Kenward (retired) of the Medical Statistics Unit at the London School of Hygiene and Tropical Medicine and James Roger (retired) of the Research Statistics Unit at GlaxoSmithKline for answering our questions about their methods.

Charles Roy Henderson (1911–1989) was born in Iowa and grew up on the family farm. His education in animal husbandry, animal nutrition, and statistics at Iowa State was interspersed with jobs in the Iowa Extension Service, Ohio University, and the US Army. After completing his PhD, Henderson joined the Animal Science faculty at Cornell. He developed and applied statistical methods in the improvement of farm livestock productivity through genetic selection, with particular focus on dairy cattle. His methods are general and have been used worldwide in livestock breeding and beyond agriculture. Henderson's work on variance components and best linear unbiased predictions has proved to be one of the main roots of current mixed-model methods.

References

Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

Baldwin, S. 2019. *Psychological Statistics and Psychometrics Using Stata*. College Station, TX: Stata Press.

Baltagi, B. H., S. H. Song, and B. C. Jung. 2001. The unbalanced nested error component regression model. *Journal of Econometrics* 101: 357–381. https://doi.org/10.1016/S0304-4076(00)00089-0.

Bates, D. M., and J. C. Pinheiro. 1998. "Computational methods for multilevel modelling". In *Technical Memorandum BL0112140-980226-01TM*. Murray Hill, NJ: Bell Labs, Lucent Technologies.

Brown, H., and R. J. Prescott. 2015. *Applied Mixed Models in Medicine*. 3rd ed. Chichester, UK: Wiley.

Cameron, A. C., and P. K. Trivedi. 2022. *Microeconometrics Using Stata*. 2nd ed. College Station, TX: Stata Press.

Canette, I. 2011. Including covariates in crossed-effects models. *The Stata Blog: Not Elsewhere Classified*. https://blog.stata.com/2010/12/22/including-covariates-in-crossed-effects-models/.

———. 2014. Using gsem to combine estimation results. *The Stata Blog: Not Elsewhere Classified*. https://blog.stata.com/2014/08/18/using-gsem-to-combine-estimation-results/.

Carle, A. C. 2009. Fitting multilevel models in complex survey data with design weights: Recommendations. *BMC Medical Research Methodology* 9: art. 49. https://doi.org/10.1186/1471-2288-9-49.

Chen, X., and L. Wei. 2003. A comparison of recent methods for the analysis of small-sample cross-over studies. *Statistics in Medicine* 22: 2821–2833. https://doi.org/10.1002/sim.1537.

Cressie, N. 1993. *Statistics for Spatial Data*. Rev. ed. New York: Wiley. https://doi.org/10.1002/9781119115151.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B ser., 39: 1–38. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Diggle, P. J., P. J. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.

Elston, D. A. 1998. Estimation of denominator degrees of freedom of *F*-distributions for assessing Wald statistics for fixed-effect factors in unbalanced mixed models. *Biometrics* 54: 1085–1096. https://doi.org/10.2307/2533859.

Fai, A. H.-T., and P. L. Cornelius. 1996. Approximate *F*-tests of multiple degree of freedom hypotheses in generalized least squares analyses of unbalanced split-plot experiments. *Journal of Statistical Computation and Simulation* 54: 363–378. https://doi.org/10.1080/00949659608811740.

Fitzmaurice, G. M., N. M. Laird, and J. H. Ware. 2011. *Applied Longitudinal Analysis*. 2nd ed. Hoboken, NJ: Wiley.

Gambino, A. J., and D. B. McCoach. 2025. r2_mlm: A command for computing R-squared measures for models fit by mixed. *Stata Journal* 25: 719–742.

Giesbrecht, F. G., and J. C. Burns. 1985. Two-stage analysis based on a mixed model: Large-sample asymptotic theory and small-sample simulation results. *Biometrics* 41: 477–486. https://doi.org/10.2307/2530872.

Giesecke, J., and U. Kohler. 2024. Two-step analysis of hierarchical data. *Stata Journal* 24: 213–249.

Goldstein, H. 1986. Efficient statistical modelling of longitudinal data. *Annals of Human Biology* 13: 129–141. https://doi.org/10.1080/03014468600008271.

Graubard, B. I., and E. L. Korn. 1996. Modelling the sampling design in the analysis of health surveys. *Statistical Methods in Medical Research* 5: 263–281. https://doi.org/10.1177/096228029600500304.

Halbmeier, C., A.-K. Kreutzmann, T. Schmid, and C. Schröder. 2019. The fayherriot command for estimating small-area indicators. *Stata Journal* 19: 626–644.

Harville, D. A. 1977. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association* 72: 320–338. https://doi.org/10.2307/2286796.

Henderson, C. R. 1953. Estimation of variance and covariance components. *Biometrics* 9: 226–252. https://doi.org/10.2307/3001853.

Hocking, R. R. 1985. *The Analysis of Linear Models*. Monterey, CA: Brooks/Cole.

Horton, N. J. 2011. Stata tip 95: Estimation of error covariances in a linear model. *Stata Journal* 11: 145–148.

Huber, C. 2013a. Multilevel linear models in Stata, part 1: Components of variance. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2013/02/04/multilevel-linear-models-in-stata-part-1-components-of-variance/.

———. 2013b. Multilevel linear models in Stata, part 2: Longitudinal data. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2013/02/18/multilevel-linear-models-in-stata-part-2-longitudinal-data/.

———. 2014. How to simulate multilevel/longitudinal data. *The Stata Blog: Not Elsewhere Classified.* https://blog.stata.com/2014/07/18/how-to-simulate-multilevellongitudinal-data/.

Kackar, R. N., and D. A. Harville. 1984. Approximations for standard errors of estimators of fixed and random effects in mixed linear models. *Journal of the American Statistical Association* 79: 853–862. https://doi.org/10.2307/2288715.

Kenward, M. G., and J. H. Roger. 1997. Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics* 53: 983–997. https://doi.org/10.2307/2533558.

———. 2009. An improved approximation to the precision of fixed effects from restricted maximum likelihood. *Computational Statistics and Data Analysis* 53: 2583–2595. https://doi.org/10.1016/j.csda.2008.12.013.

Khuri, A. I., T. Mathew, and B. K. Sinha. 1998. *Statistical Tests for Mixed Linear Models.* New York: Wiley.

Laird, N. M., and J. H. Ware. 1982. Random-effects models for longitudinal data. *Biometrics* 38: 963–974. https://doi.org/10.2307/2529876.

LaMotte, L. R. 1973. Quadratic estimation of variance components. *Biometrics* 29: 311–330. https://doi.org/10.2307/2529395.

Marchenko, Y. V. 2006. Estimating variance components in Stata. *Stata Journal* 6: 1–21.

McCulloch, C. E., S. R. Searle, and J. M. Neuhaus. 2008. *Generalized, Linear, and Mixed Models.* 2nd ed. Hoboken, NJ: Wiley.

Munnell, A. H. 1990. Why has productivity growth declined? Productivity and public investment. *New England Economic Review* Jan./Feb.: 3–22.

Nichols, A. 2007. Causal inference with observational data. *Stata Journal* 7: 507–541.

Palmer, T. M., C. M. Macdonald-Wallis, D. A. Lawlor, and K. Tilling. 2014. Estimating adjusted associations between random effects from multilevel models: The reffadjust package. *Stata Journal* 14: 119–140.

Pantazis, N., and G. Touloumi. 2010. Analyzing longitudinal data in the presence of informative drop-out: The jmre1 command. *Stata Journal* 10: 226–251.

Pfeffermann, D., C. J. Skinner, D. J. Holmes, H. Goldstein, and J. Rasbash. 1998. Weighting for unequal selection probabilities in multilevel models. *Journal of the Royal Statistical Society,* B ser., 60: 23–40. https://doi.org/10.1111/1467-9868.00106.

Pierson, R. A., and O. J. Ginther. 1987. Follicular population dynamics during the estrous cycle of the mare. *Animal Reproduction Science* 14: 219–231. https://doi.org/10.1016/0378-4320(87)90085-6.

Pinheiro, J. C., and D. M. Bates. 2000. *Mixed-Effects Models in S and S-PLUS.* New York: Springer. https://doi.org/10.1007/b98882.

Prosser, R., J. Rasbash, and H. Goldstein. 1991. *ML3 Software for 3-Level Analysis: User's Guide for V. 2.* London: Institute of Education, University of London.

Rabe-Hesketh, S., and A. Skrondal. 2006. Multilevel modelling of complex survey data. *Journal of the Royal Statistical Society,* A ser., 169: 805–827. https://doi.org/10.1111/j.1467-985X.2006.00426.x.

———. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Rao, C. R. 1973. *Linear Statistical Inference and Its Applications*. 2nd ed. New York: Wiley. https://doi.org/10.1002/9780470316436.

Raudenbush, S. W., and A. S. Bryk. 2002. *Hierarchical Linear Models: Applications and Data Analysis Methods*. 2nd ed. Thousand Oaks, CA: Sage.

Rikken, M. G. J., and R. P. G. Van Rijn. 1993. Soil pollution with heavy metals—an inquiry into spatial variation, cost of mapping and the risk evaluation of copper, cadmium, lead and zinc in the floodplains of the Meuse west of Stein, the Netherlands. Doctoraalveldwerkverslag, Dept. of Physical Geography, Utrecht University.

Robson, K., and D. Pevalin. 2016. *Multilevel Modeling in Plain Language.* London: Sage.

Ruppert, D., M. P. Wand, and R. J. Carroll. 2003. *Semiparametric Regression*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511755453.

Satterthwaite, F. E. 1946. An approximate distribution of estimates of variance components. *Biometrics Bulletin* 2: 110–114. https://doi.org/10.2307/3002019.

Schaalje, G. B., J. B. McBride, and G. W. Fellingham. 2002. Adequacy of approximations to distributions of test statistics in complex mixed linear models. *Journal of Agricultural, Biological, and Environmental Statistics* 7: 512–524. https://doi.org/10.1198/108571102726.

Schluchter, M. D., and J. D. Elashoff. 1990. Small-sample adjustments to tests with unbalanced repeated measures assuming several covariance structures. *Journal of Statistical Computation and Simulation* 37: 69–87. https://doi.org/10.1080/00949659008811295.

Schunck, R. 2013. Within and between estimates in random-effects models: Advantages and drawbacks of correlated random effects and hybrid models. *Stata Journal* 13: 65–76.

Searle, S. R. 1989. Obituary: Charles Roy Henderson 1911–1989. *Biometrics* 45: 1333–1335.

Searle, S. R., G. Casella, and C. E. McCulloch. 1992. *Variance Components*. New York: Wiley. https://doi.org/10.1002/9780470316856.

Thompson, W. A., Jr. 1962. The problem of negative estimates of variance components. *Annals of Mathematical Statistics* 33: 273–289. https://doi.org/10.1214/aoms/1177704731.

Vallejo, G., P. Fernández, F. J. Herrero, and N. M. Conejo. 2004. Alternative procedures for testing fixed effects in repeated measures designs when assumptions are violated. *Psicothema* 16: 498–508.

Verbeke, G., and G. Molenberghs. 2000. *Linear Mixed Models for Longitudinal Data*. New York: Springer. https://doi.org/10.1007/978-1-4419-0300-6.

West, B. T., K. B. Welch, and A. T. Gałecki. 2022. *Linear Mixed Models: A Practical Guide Using Statistical Software*. 3rd ed. Boca Raton, FL: CRC Press.

Wiggins, V. L. 2011. Multilevel random effects in xtmixed and sem—the long and wide of it. *The Stata Blog: Not Elsewhere Classified*. https://blog.stata.com/2011/09/28/multilevel-random-effects-in-xtmixed-and-sem-the-long-and-wide-of-it/.

Winer, B. J., D. R. Brown, and K. M. Michels. 1991. *Statistical Principles in Experimental Design*. 3rd ed. New York: McGraw–Hill.

Also see

[ME] **mixed postestimation** — Postestimation tools for mixed

[ME] **meglm** — Multilevel mixed-effects generalized linear models

[ME] **menl** — Nonlinear mixed-effects regression

[ME] **me** — Introduction to multilevel mixed-effects models

[MI] **Estimation** — Estimation commands for use with mi estimate

[BAYES] **bayes: mixed** — Bayesian multilevel linear regression

[SEM] **Intro 5** — Tour of models (*Multilevel mixed-effects models*)

[XT] **xtrc** — Random-coefficients model

[XT] **xtreg** — Linear models for panel data+

[U] 20 Estimation and postestimation commands

mixed postestimation — Postestimation tools for mixed

Postestimation commands | predict | margins | test and testparm
lincom | contrast | pwcompare | Remarks and examples
Stored results | Methods and formulas | References | Also see

Postestimation commands

The following postestimation commands are of special interest after `mixed`:

Command	Description
`estat df`	calculate and display degrees of freedom for fixed effects
`estat group`	summarize the composition of the nested groups
`estat icc`	estimate intraclass correlations
`estat recovariance`	display the estimated random-effects covariance matrices
`estat sd`	display variance components as standard deviations and correlations
`estat wcorrelation`	display within-cluster correlations and standard deviations

The following standard postestimation commands are also available:

Command	Description
`contrast`	contrasts and ANOVA-style joint tests of parameters
`estat ic`	Akaike's, consistent Akaike's, corrected Akaike's, and Schwarz's Bayesian information criteria (AIC, CAIC, AICc, and BIC, respectively)
`estat summarize`	summary statistics for the estimation sample
`estat vce`	variance–covariance matrix of the estimators (VCE)
`estimates`	cataloging estimation results
`etable`	table of estimation results
`hausman`	Hausman's specification test
`lincom`	point estimates, standard errors, testing, and inference for linear combinations of parameters
`lrtest`	likelihood-ratio test
`margins`	marginal means, predictive margins, marginal effects, and average marginal effects
`marginsplot`	graph the results from margins (profile plots, interaction plots, etc.)
`nlcom`	point estimates, standard errors, testing, and inference for nonlinear combinations of parameters
`predict`	predictions and their SEs, residuals, etc.
`predictnl`	point estimates, standard errors, testing, and inference for generalized predictions
`pwcompare`	pairwise comparisons of parameters
`test`	Wald tests of simple and composite linear hypotheses
`testnl`	Wald tests of nonlinear hypotheses

predict

Description for predict

`predict` creates a new variable containing predictions such as linear predictions, standard errors, fitted values, residuals, and standardized residuals.

Menu for predict

Statistics > Postestimation

Syntax for predict

Syntax for obtaining predictions of the outcome and other statistics

```
predict [type] newvar [if] [in] [, statistic relevel(levelvar)]
```

Syntax for obtaining BLUPs of random effects and the BLUPs' standard errors

```
predict [type] {stub* | newvars} [if] [in], reffects [relevel(levelvar)
  reses(stub* | newvars)]
```

Syntax for obtaining scores after ML estimation

```
predict [type] stub* [if] [in], scores
```

statistic	Description
Main	
`xb`	linear prediction for the fixed portion of the model only; the default
`stdp`	standard error of the fixed-portion linear prediction
`fitted`	fitted values, fixed-portion linear prediction plus contributions based on predicted random effects
`residuals`	residuals, response minus fitted values
* `rstandard`	standardized residuals

Unstarred statistics are available both in and out of sample; type `predict ... if e(sample) ...` if wanted only for the estimation sample. Starred statistics are calculated only for the estimation sample, even when `if e(sample)` is not specified.

Options for predict

Main

`xb`, the default, calculates the linear prediction $\mathbf{x}\boldsymbol{\beta}$ based on the estimated fixed effects (coefficients) in the model. This is equivalent to fixing all random effects in the model to their theoretical mean value of 0.

`stdp` calculates the standard error of the linear predictor $\mathbf{x}\boldsymbol{\beta}$.

`fitted` calculates fitted values, which are equal to the fixed-portion linear predictor *plus* contributions based on predicted random effects, or in mixed-model notation, $\mathbf{x}\beta+\mathbf{Zu}$. By default, the fitted values take into account random effects from all levels in the model; however, if the `relevel(`*levelvar*`)` option is specified, then the fitted values are fit beginning with the topmost level down to and including level *levelvar*. For example, if `classes` are nested within `schools`, then typing

```
. predict yhat_school, fitted relevel(school)
```

would produce school-level predictions. That is, the predictions would incorporate school-specific random effects but not those for each class nested within each school.

`residuals` calculates residuals, equal to the responses minus fitted values. By default, the fitted values take into account random effects from all levels in the model; however, if the `relevel(`*levelvar*`)` option is specified, then the fitted values are fit beginning at the topmost level down to and including level *levelvar*.

`rstandard` calculates standardized residuals, equal to the residuals multiplied by the inverse square root of the estimated error covariance matrix.

`reffects` calculates best linear unbiased predictions (BLUPs) of the random effects. By default, BLUPs for all random effects in the model are calculated. However, if the `relevel(`*levelvar*`)` option is specified, then BLUPs for only level *levelvar* in the model are calculated. For example, if `classes` are nested within `schools`, then typing

```
. predict b*, reffects relevel(school)
```

would produce BLUPs at the school level. You must specify q new variables, where q is the number of random-effects terms in the model (or level). However, it is much easier to just specify *stub*`*` and let Stata name the variables *stub*`1`, *stub*`2`, …, *stub*q for you.

Rabe-Hesketh and Skrondal (2022, sec. 2.11.2) discuss the link between the empirical Bayes predictions and BLUPs and how these predictions are unbiased. They are unbiased when the groups associated with the random effects are expected to vary in repeated samples. If you expect the groups to be fixed in repeated samples, then these predictions are no longer unbiased.

`scores` calculates the parameter-level scores, one for each parameter in the model including regression coefficients and variance components. The score for a parameter is the first derivative of the log likelihood (or log pseudolikelihood) with respect to that parameter. One score per highest-level group is calculated, and it is placed on the last record within that group. Scores are calculated in the estimation metric as stored in `e(b)`.

`scores` is not available after restricted maximum-likelihood (REML) estimation.

`relevel(`*levelvar*`)` specifies the level in the model at which predictions involving random effects are to be obtained; see the options above for the specifics. *levelvar* is the name of the model level and is either the name of the variable describing the grouping at that level or is `_all`, a special designation for a group comprising all the estimation data.

`reses(`*stub*`*` | *newvars*`)` calculates the standard errors of the BLUPs of the random effects. By default, standard errors for all BLUPs in the model are calculated. However, if the `relevel(`*levelvar*`)` option is specified, then standard errors for only level *levelvar* in the model are calculated; see the `reffects` option.

You must specify q new variables, where q is the number of random-effects terms in the model (or level). However, it is much easier to just specify *stub*`*` and let Stata name the variables *stub*`1`, *stub*`2`, ..., *stubq* for you. The new variables will have the same storage type as the corresponding random-effects variables.

The `reffects` and `reses()` options often generate multiple new variables at once. When this occurs, the random effects (or standard errors) contained in the generated variables correspond to the order in which the variance components are listed in the output of `mixed`. Still, examining the variable labels of the generated variables (with the `describe` command, for instance) can be useful in deciphering which variables correspond to which terms in the model.

margins

Description for margins

`margins` estimates margins of response for linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

`margins` [*marginlist*] [`,` *options*]

`margins` [*marginlist*] `,` `predict(`*statistic* `...)` [*options*]

statistic	Description
`xb`	linear predictor for the fixed portion of the model only; the default
`stdp`	not allowed with `margins`
`fitted`	not allowed with `margins`
`residuals`	not allowed with `margins`
`rstandard`	not allowed with `margins`
`reffects`	not allowed with `margins`
`scores`	not allowed with `margins`

Statistics not allowed with `margins` are functions of stochastic quantities other than `e(b)`.

For the full syntax, see [R] **margins**.

test and testparm

Description for test and testparm

`test` and `testparm`, by default, perform χ^2 tests of simple and composite linear hypotheses about the parameters for the most recently fit `mixed` model. They also support F tests with a small-sample adjustment for fixed effects.

Menu for test and testparm

Statistics > Postestimation

Syntax for test and testparm

`test` (*spec*) [(*spec*) ...] [, *test_options* `small`]

`testparm` *varlist* [, *testparm_options* `small`]

Options for test and testparm

Options

test_options; see [R] **test** options. Options `df()`, `common`, and `nosvyadjust` may not be specified together with `small`.

testparm_options; see options of `testparm` in [R] **test**. Options `df()` and `nosvyadjust` may not be specified together with `small`.

`small` specifies that F tests for fixed effects be carried out with the denominator degrees of freedom (DDF) obtained by the same method used in the most recently fit `mixed` model. If option `dfmethod()` is not specified in the previous `mixed` command, option `small` is not allowed. For certain methods, the DDF for some tests may not be available. See *Small-sample inference for fixed effects* in [ME] **mixed** for more details.

lincom

Description for lincom

`lincom`, by default, computes point estimates, standard errors, z statistics, p-values, and confidence intervals for linear combinations of parameters after `mixed`. `lincom` also provides t statistics for linear combinations of the fixed effects, with the degrees of freedom calculated by the DF method specified in option `dfmethod()` of `mixed`.

Menu for lincom

Statistics > Postestimation

Syntax for lincom

`lincom` *exp* [, *lincom_options* `small`]

Options for lincom

lincom_options; see [R] **lincom** options. Option `df()` may not be specified together with `small`.

`small` specifies that t statistics for linear combinations of fixed effects be displayed with the degrees of freedom obtained by the same method used in the most recently fit `mixed` model. If option `dfmethod()` is not specified in the previous `mixed` command, option `small` is not allowed. For certain methods, the degrees of freedom for some linear combinations may not be available. See *Small-sample inference for fixed effects* in [ME] **mixed** for more details.

contrast

Description for contrast

`contrast`, by default, performs χ^2 tests of linear hypotheses and forms contrasts involving factor variables and their interactions for the most recently fit `mixed` model. `contrast` also supports tests with small-sample adjustments after `mixed, dfmethod()`.

Menu for contrast

Statistics > Postestimation

Syntax for contrast

`contrast` *termlist* [, *contrast_options* `small`]

Options for contrast

contrast_options; see [R] **contrast** options. Options `df()` and `nosvyadjust` may not be specified together with `small`.

`small` specifies that tests for contrasts be carried out with the DDF obtained by the same method used in the most recently fit `mixed` model. If option `dfmethod()` is not specified in the previous `mixed` command, option `small` is not allowed. For certain methods, the DDF for some contrasts may not be available. See *Small-sample inference for fixed effects* in [ME] **mixed** for more details.

pwcompare

Description for pwcompare

`pwcompare` performs pairwise comparisons across the levels of factor variables from the most recently fit `mixed` model. `pwcompare`, by default, reports the comparisons as contrasts (differences) of margins along with z tests or confidence intervals for the pairwise comparisons. `pwcompare` also supports t tests with small-sample adjustments after `mixed, dfmethod()`.

Menu for pwcompare

Statistics > Postestimation

Syntax for pwcompare

`pwcompare` *marginlist* [, *pwcompare_options* `small`]

Options for pwcompare

pwcompare_options; see [R] **pwcompare** options. Option `df()` may not be specified together with `small`.

`small` specifies that t tests for pairwise comparisons be carried out with the degrees of freedom obtained by the same method used in the most recently fit `mixed` model with the `dfmethod()` option. If option `dfmethod()` is not specified in the previous `mixed` command, option `small` is not allowed. For certain methods, the degrees of freedom for some pairwise comparisons may not be available. See *Small-sample inference for fixed effects* in [ME] **mixed** for more details.

Remarks and examples

Various predictions, statistics, and diagnostic measures are available after fitting a mixed model using `mixed`. For the most part, calculation centers around obtaining BLUPs of the random effects. Random effects are not estimated when the model is fit but instead need to be predicted after estimation. Calculation of intraclass correlations, estimating the dependence between responses for different levels of nesting, may also be of interest.

▷ Example 1: Obtaining predictions of random effects and checking model fit

In example 3 of [ME] **mixed**, we modeled the weights of 48 pigs measured on nine successive weeks as

$$\texttt{weight}_{ij} = \beta_0 + \beta_1 \texttt{week}_{ij} + u_{0j} + u_{1j}\texttt{week}_{ij} + \epsilon_{ij} \tag{1}$$

for $i = 1, \ldots, 9$, $j = 1, \ldots, 48$, $\epsilon_{ij} \sim N(0, \sigma^2_\epsilon)$, and u_{0j} and u_{1j} normally distributed with mean 0 and variance–covariance matrix

$$\mathbf{\Sigma} = \text{Var}\begin{bmatrix} u_{0j} \\ u_{1j} \end{bmatrix} = \begin{bmatrix} \sigma^2_{u0} & \sigma_{01} \\ \sigma_{01} & \sigma^2_{u1} \end{bmatrix}$$

```
. use https://www.stata-press.com/data/r19/pig
(Longitudinal analysis of pig weights)
. mixed weight week || id: week, covariance(unstructured)
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -868.96185
Iteration 1:  Log likelihood = -868.96185
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =        432
Group variable: id                              Number of groups  =         48
                                                Obs per group:
                                                              min =          9
                                                              avg =        9.0
                                                              max =          9
                                                Wald chi2(1)      =    4649.17
Log likelihood = -868.96185                     Prob > chi2       =     0.0000
```

weight	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
week	6.209896	.0910745	68.18	0.000	6.031393	6.388399
_cons	19.35561	.3996387	48.43	0.000	18.57234	20.13889

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Unstructured				
var(week)	.3715251	.0812958	.2419532	.570486
var(_cons)	6.823363	1.566194	4.351297	10.69986
cov(week,_cons)	-.0984378	.2545767	-.5973991	.4005234
var(Residual)	1.596829	.123198	1.372735	1.857505

```
LR test vs. linear model: chi2(3) = 764.58                Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Rather than see the estimated variance components listed as variance and covariances as above, we can instead see them as correlations and standard deviations using `estat sd`; see [ME] **estat sd**.

```
. estat sd
```

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Unstructured				
sd(week)	.6095286	.0666874	.4918874	.7553052
sd(_cons)	2.612157	.2997895	2.085976	3.271064
corr(week,_cons)	-.0618257	.1575911	-.3557072	.243182
sd(Residual)	1.263657	.0487466	1.171638	1.362903

We can use `estat recovariance` to display the estimated variance components $\widehat{\Sigma}$ as a correlation matrix; see [ME] **estat recovariance**.

```
. estat recovariance, correlation
Random-effects correlation matrix for level id
```

	week	_cons
week	1	
_cons	-.0618257	1

Finally, we can use `estat wcorrelation` to display the within-cluster marginal standard deviations and correlations for one of the clusters; see [ME] **estat wcorrelation**.

```
. estat wcorrelation, format(%4.2g)
Standard deviations and correlations for id = 1:
Standard deviations:
```

obs	1	2	3	4	5	6	7	8	9
sd	2.9	3.1	3.3	3.7	4.1	4.5	5	5.5	6.1

Correlations:

obs	1	2	3	4	5	6	7	8	9
1	1								
2	.8	1							
3	.77	.83	1						
4	.72	.81	.86	1					
5	.67	.78	.85	.89	1				
6	.63	.75	.83	.88	.91	1			
7	.59	.72	.81	.87	.91	.93	1		
8	.55	.69	.79	.86	.9	.93	.94	1	
9	.52	.66	.77	.85	.89	.92	.94	.95	1

Because within-cluster correlations can vary between clusters, `estat wcorrelation` by default displays the results for the first cluster. In this example, each cluster (pig) has the same number of observations, and the timings of measurements (`week`) are the same between clusters. Thus the within-cluster correlations are the same for all the clusters. In example 1 of [ME] **estat wcorrelation**, we fit a model where different clusters have different within-cluster correlations and show how to display these correlations.

We can also obtain BLUPs of the pig-level random effects (u_{0j} and u_{1j}). We need to specify the variables to be created in the order `u1 u0` because that is the order in which the corresponding variance components are listed in the output (`week _cons`). We obtain the predictions and list them for the first 10 pigs.

```
. predict u1 u0, reffects
. by id, sort: generate tolist = (_n==1)
. list id u0 u1 if id <=10 & tolist
```

	id	u0	u1
1.	1	.2369444	-.3957636
10.	2	-1.584127	.510038
19.	3	-3.526551	.3200372
28.	4	1.964378	-.7719702
37.	5	1.299236	-.9241479
46.	6	-1.147302	-.5448151
55.	7	-2.590529	.0394454
64.	8	-1.137067	-.1696566
73.	9	-3.189545	-.7365507
82.	10	1.160324	.0030772

If you forget how to order your variables in `predict`, or if you use `predict` *stub*`*`, remember that `predict` labels the generated variables for you to avoid confusion.

```
. describe u0 u1
Variable      Storage   Display    Value
    name         type    format    label      Variable label
------------------------------------------------------------------------------
u0              float   %9.0g                 BLUP r.e. for id: _cons
u1              float   %9.0g                 BLUP r.e. for id: week
```

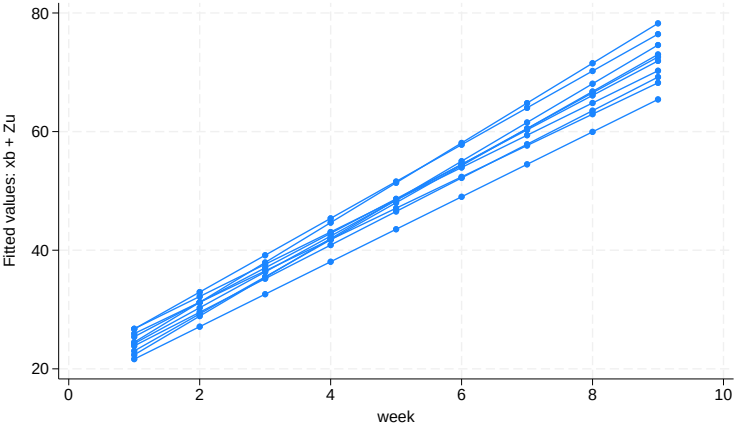
Examining (1), we see that within each pig, the successive weight measurements are modeled as simple linear regression with intercept $\beta_0 + u_{j0}$ and slope $\beta_1 + u_{j1}$. We can generate estimates of the pig-level intercepts and slopes with

```
. generate intercept = _b[_cons] + u0
. generate slope = _b[week] + u1
. list id intercept slope if id<=10 & tolist
```

	id	interc~t	slope
1.	1	19.59256	5.814132
10.	2	17.77149	6.719934
19.	3	15.82906	6.529933
28.	4	21.31999	5.437926
37.	5	20.65485	5.285748
46.	6	18.20831	5.665081
55.	7	16.76509	6.249341
64.	8	18.21855	6.040239
73.	9	16.16607	5.473345
82.	10	20.51594	6.212973

Thus we can plot estimated regression lines for each of the pigs. Equivalently, we can just plot the fitted values because they are based on both the fixed and the random effects:

```
. predict fitweight, fitted
. twoway connected fitweight week if id<=10, connect(L)
```

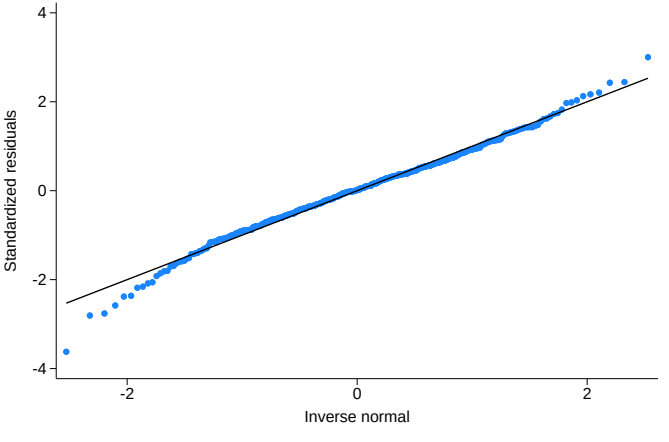


We can also generate standardized residuals and see whether they follow a standard normal distribution, as they should in any good-fitting model:

```
. predict rs, rstandard
. summarize rs
```

Variable	Obs	Mean	Std. dev.	Min	Max
rs	432	1.01e-09	.8929356	-3.621446	3.000929

```
. qnorm rs
```



Example 2: Estimating the intraclass correlation

Following Rabe-Hesketh and Skrondal (2022, chap. 2), we fit a two-level random-effects model for human peak-expiratory-flow rate. The subjects were each measured twice with the Mini-Wright peak-flow meter. It is of interest to determine how reliable the meter is as a measurement device. The intraclass correlation provides a measure of reliability. Formally, in a two-level random-effects model, the intraclass correlation corresponds to the correlation of measurements within the same individual and also to the proportion of variance explained by the individual random effect.

First, we fit the two-level model with `mixed`:

```
. use https://www.stata-press.com/data/r19/pefrate, clear
(Peak-expiratory-flow rate)
. mixed wm || id:
Performing EM optimization ...
Performing gradient-based optimization:
Iteration 0:  Log likelihood = -184.57839
Iteration 1:  Log likelihood = -184.57839
Computing standard errors ...
Mixed-effects ML regression                     Number of obs     =         34
Group variable: id                              Number of groups  =         17
                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2
                                                Wald chi2(0)      =          .
Log likelihood = -184.57839                     Prob > chi2       =          .
```

wm	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
_cons	453.9118	26.18617	17.33	0.000	402.5878	505.2357

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
id: Identity				
var(_cons)	11458.94	3998.952	5782.176	22708.98
var(Residual)	396.441	135.9781	202.4039	776.4942

```
LR test vs. linear model: chibar2(01) = 46.27          Prob >= chibar2 = 0.0000
```

Now we use `estat icc` to estimate the intraclass correlation:

```
. estat icc
Intraclass correlation
```

Level	ICC	Std. err.	[95% conf.	interval]
id	.9665602	.0159495	.9165853	.9870185

This correlation is close to 1, indicating that the Mini-Wright peak-flow meter is reliable. But as noted by Rabe-Hesketh and Skrondal (2022), the reliability is not only a characteristic of the instrument but also of the between-subject variance. Here we see that the between-subject standard deviation, `sd(_cons)`, is much larger than the within-subject standard deviation, `sd(Residual)`.

In the presence of fixed-effects covariates, `estat icc` reports the residual intraclass correlation, the correlation between measurements conditional on the fixed-effects covariates. This is equivalent to the correlation of the model errors.

In the presence of random-effects covariates, the intraclass correlation is no longer constant and depends on the values of the random-effects covariates. In this case, `estat icc` reports conditional intraclass correlations assuming 0 values for all random-effects covariates. For example, in a two-level model, this conditional correlation represents the correlation of the errors for two measurements on the same subject, which both have random-effects covariates equal to 0. Similarly to the interpretation of intercept variances in random-coefficients models (Rabe-Hesketh and Skrondal 2022, chap. 4), interpretation of this conditional intraclass correlation relies on the usefulness of the 0 baseline values of random-effects covariates. For example, mean centering of the covariates is often used to make a 0 value a useful reference.

See [ME] **estat icc** for more information.

◁

▷ Example 3: Estimating residual intraclass correlations

In example 4 of [ME] **mixed**, we estimated a Cobb–Douglas production function with random intercepts at the region level and at the state-within-region level:

$$\mathbf{y}_{jk} = \mathbf{X}_{jk}\boldsymbol{\beta} + u_k^{(3)} + u_{jk}^{(2)} + \boldsymbol{\epsilon}_{jk}$$

```
. use https://www.stata-press.com/data/r19/productivity
(Public capital productivity)
. mixed gsp private emp hwy water other unemp || region: || state:
  (output omitted)
```

We can use `estat group` to see how the data are broken down by state and region:

```
. estat group
```

Group variable	No. of groups	Observations per group Minimum	Average	Maximum
region	9	51	90.7	136
state	48	17	17.0	17

We are reminded that we have balanced productivity data for 17 years for each state.

We can use `predict, fitted` to get the fitted values

$$\hat{\mathbf{y}}_{jk} = \mathbf{X}_{jk}\widehat{\boldsymbol{\beta}} + \hat{u}_k^{(3)} + \hat{u}_{jk}^{(2)}$$

but if we instead want fitted values at the region level, that is,

$$\hat{\mathbf{y}}_{jk} = \mathbf{X}_{jk}\widehat{\boldsymbol{\beta}} + \hat{u}_k^{(3)}$$

we need to use the relevel() option:

```
. predict gsp_region, fitted relevel(region)
. list gsp gsp_region in 1/10
```

	gsp	gsp_re~n
1.	10.25478	10.40529
2.	10.2879	10.42336
3.	10.35147	10.47343
4.	10.41721	10.52648
5.	10.42671	10.54947
6.	10.4224	10.53537
7.	10.4847	10.60781
8.	10.53111	10.64727
9.	10.59573	10.70503
10.	10.62082	10.72794

❑ Technical note

Out-of-sample predictions are permitted after mixed, but if these predictions involve BLUPs of random effects, the integrity of the estimation data must be preserved. If the estimation data have changed since the mixed model was fit, predict will be unable to obtain predicted random effects that are appropriate for the fitted model and will give an error. Thus to obtain out-of-sample predictions that contain random-effects terms, be sure that the data for these predictions are in observations that augment the estimation data.

❑

We can use estat icc to estimate residual intraclass correlations between productivity years in the same region and in the same state and region.

```
. estat icc
Residual intraclass correlation
```

Level	ICC	Std. err.	[95% conf.	interval]
region	.159893	.127627	.0287143	.5506202
state|region	.8516265	.0301733	.7823466	.9016272

estat icc reports two intraclass correlations for this three-level nested model. The first is the level-3 intraclass correlation at the region level, the correlation between productivity years in the same region. The second is the level-2 intraclass correlation at the state-within-region level, the correlation between productivity years in the same state and region.

Conditional on the fixed-effects covariates, we find that annual productivity is only slightly correlated within the same region, but it is highly correlated within the same state and region. We estimate that state and region random effects compose approximately 85% of the total error variance.

◁

▷ Example 4: Small-sample adjusted tests for fixed effects

To illustrate the use of `test` and `testparm` with the `small` option for small-sample adjusted tests for fixed effects, we refit the dental veneer data from example 16 of [ME] **mixed** using the Satterthwaite method (option `dfmethod(satterthwaite)`) to compute the DF for fixed effects.

```
. use https://www.stata-press.com/data/r19/veneer, clear
(Dental veneer data)
. mixed gcf followup base_gcf cda age
> || patient: followup, covariance(unstructured)
> || tooth:, reml nolog dfmethod(satterthwaite)
Mixed-effects REML regression                   Number of obs =    110
        Grouping information
------------------------------------------------------------------------------
 Group variable |     No. of       Observations per group
                |     groups    Minimum    Average    Maximum
----------------+-------------------------------------------------------------
        patient |         12          2        9.2         12
          tooth |         55          2        2.0          2
------------------------------------------------------------------------------
DF method: Satterthwaite                        DF:       min =  10.41
                                                          avg =  28.96
                                                          max =  50.71
                                                F(4, 16.49)   =   1.87
Log restricted-likelihood = -420.92761          Prob > F      = 0.1638
------------------------------------------------------------------------------
         gcf | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    followup |   .3009815   1.936863     0.16   0.879    -3.963754    4.565717
    base_gcf |  -.0183127   .1433094    -0.13   0.899    -.3065704     .269945
         cda |   -.329303   .5292525    -0.62   0.537     -1.39197    .7333636
         age |  -.5773932   .2139656    -2.70   0.022    -1.051598   -.1031885
       _cons |   45.73862   12.55497     3.64   0.001     19.90352    71.57372
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects parameters |   Estimate   Std. err.      [95% conf. interval]
----------------------------+-------------------------------------------------
patient: Unstructured       |
               var(followup) |   41.88772   18.79997       17.38009    100.9535
                  var(_cons) |   524.9851   253.0205       204.1287    1350.175
        cov(followup,_cons) |  -140.4229   66.57623      -270.9099   -9.935904
----------------------------+-------------------------------------------------
tooth: Identity             |
                  var(_cons) |   47.45738   16.63034        23.8792     94.3165
----------------------------+-------------------------------------------------
               var(Residual) |   48.86704   10.50523       32.06479    74.47382
------------------------------------------------------------------------------
LR test vs. linear model: chi2(4) = 91.12                 Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

Now we can, for example, test the hypotheses that all fixed effects are zero by typing

```
. testparm *, small
 ( 1)  [gcf]followup = 0
 ( 2)  [gcf]base_gcf = 0
 ( 3)  [gcf]cda = 0
 ( 4)  [gcf]age = 0
       F(  4, 16.49) =    1.87
            Prob > F =    0.1638
```

The F statistic for the overall test is 1.87, and the DDF is estimated to be 16.49. These results are different from the model test using the Kenward–Roger DDF method reported in the header of the estimation output in example 1 of [ME] **estat df** (the F statistic is 1.47, and the model DDF is 27.96).

The results differ because the Kenward–Roger method uses an adjusted F-test statistic and adjusts the fixed-effects variance–covariance estimator for a small sample. Both methods, however, lead to the same conclusion of no joint significance of the fixed effects.

Without option `small`, the commands `test` and `testparm` report large-sample χ^2 Wald tests. We can compare the small-sample and large-sample tests of the joint hypotheses that the coefficient on `followup` and the coefficient on `age` equal zero.

```
. test followup = age = 0, small
 ( 1)  [gcf]followup - [gcf]age = 0
 ( 2)  [gcf]followup = 0
       F(  2, 10.75) =    3.65
            Prob > F =    0.0617
. test followup = age = 0
 ( 1)  [gcf]followup - [gcf]age = 0
 ( 2)  [gcf]followup = 0
           chi2(  2) =    7.30
         Prob > chi2 =    0.0260
```

The DDF of the F test, which is computed using the Satterthwaite method from our posted results, is 10.75. The p-values are very different (0.0617 versus 0.0260), and they lead to different conclusions of whether we should reject the null hypotheses at the $\alpha = 0.05$ level.

Similarly, you can use the `small` option with `lincom` to perform small-sample inference for linear combinations of fixed effects.

◁

▷ Example 5: Small-sample adjusted contrasts

As we did with `test`, after fitting a mixed model with the `dfmethod()` option for small-sample adjustment, we can use the `small` option with `contrast` to adjust for a small sample when estimating contrasts. Suppose we have collected data on a vigilance performance test. This experiment has been designed to test the response latency scores of two modes of signal during a four-hour monitoring period. This is a split-plot factorial design where `signal` is the whole-plot factor, `hour` is the subplot factor, and `subject` is the block factor. The whole-plot factor and the subplot factor are fixed; the block factor is random. Also, suppose that two measurements are missing in this dataset.

```
. use https://www.stata-press.com/data/r19/vptscores, clear
(Vigilance performance test scores with missing data)
. tabdisp subject hour, cellvar(score) by(signal) concise missing
```

Signal and Subject ID	Monitoring period 1	2	3	4
Auditory				
1	3	4	7	7
2	6	5	.	8
3	3	4	7	9
4	3	3	6	8
Visual				
5	1	2	5	10
6	2	3	6	.
7	2	4	5	9
8	2	3	6	11

We start by fitting a mixed model. Because the dataset is small and unbalanced, we apply the Kenward–Roger method for small-sample adjustment:

```
. mixed score signal##hour || subject:, reml dfmethod(kroger) nolog nogroup
Mixed-effects REML regression                   Number of obs =     30
DF method: Kenward-Roger                        DF:       min =  16.02
                                                          avg =  16.76
                                                          max =  18.29
                                                F(7, 16.08)   =  43.84
Log restricted-likelihood =   -32.9724          Prob > F      = 0.0000
```

score	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
signal						
Visual	-2	.6288677	-3.18	0.005	-3.319693	-.6803071
hour						
2	.25	.5359916	0.47	0.647	-.8861371	1.386137
3	3.108222	.5911044	5.26	0.000	1.859163	4.357281
4	4.25	.5359916	7.93	0.000	3.113863	5.386137
signal#hour						
Visual#2	1	.7580066	1.32	0.206	-.6067405	2.606741
Visual#3	.6417778	.7979294	0.80	0.433	-1.046666	2.330221
Visual#4	4.044205	.7979294	5.07	0.000	2.355762	5.732649
_cons	3.75	.4446766	8.43	0.000	2.816836	4.683164

Random-effects parameters	Estimate	Std. err.	[95% conf.	interval]
subject: Identity				
var(_cons)	.2163751	.2345718	.0258477	1.811312
var(Residual)	.574574	.2062107	.2843515	1.161011

```
LR test vs. linear model: chibar2(01) = 1.55          Prob >= chibar2 = 0.1069
```

We can test the main effects and the interaction effects by typing the `contrast` command. With the `small` option, `contrast` reports small-sample adjusted F tests. Without the `small` option, `contrast` performs large-sample χ^2 Wald tests. Below is the comparison of the small-sample and the large-sample contrasts:

```
. contrast signal##hour, small
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	ddf	F	P>F
score				
signal	1	5.95	1.78	0.2307
hour	3	16.35	100.62	0.0000
signal#hour	3	16.35	9.66	0.0007

```
. contrast signal##hour
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	chi2	P>chi2
score			
signal	1	1.79	0.1810
hour	3	304.95	0.0000
signal#hour	3	29.35	0.0000

From these results, we can see that the p-values for the main effect of `signal` and the interaction effect vary between small-sample and large-sample tests. However, both tests indicate that the `hour` effect and the interaction effects are significant. We can decompose the interaction effect into separate interaction contrasts for further investigation.

```
. contrast r.signal#ar.hour, small
Contrasts of marginal linear predictions
Margins: asbalanced
```

	df	ddf	F	P>F
score				
signal#hour				
(Visual vs Auditory) (2 vs 1)	1	16.02	1.74	0.2056
(Visual vs Auditory) (3 vs 2)	1	16.37	0.20	0.6594
(Visual vs Auditory) (4 vs 3)	1	16.66	16.57	0.0008
Joint	3	16.35	9.66	0.0007

	Contrast	Std. err.	df	[95% conf.	interval]
score					
signal#hour					
(Visual vs Auditory) (2 vs 1)	1	.7580066	16.0	-.6067405	2.606741
(Visual vs Auditory) (3 vs 2)	-.3582222	.7979294	16.4	-2.046666	1.330221
(Visual vs Auditory) (4 vs 3)	3.402427	.8359478	16.7	1.635991	5.168863

From previous analysis, we already knew the overall interaction was significant. From the decomposition, we can easily see that the overall significance is driven by differences in the third and fourth hours; the change in response latency from hour three to hour four is greater for visual signals than for auditory signals.

We can also calculate the pairwise differences of the hourly marginal means by typing the `pwcompare` command. With the `small` option, `pwcompare` reports small-sample adjusted pairwise comparisons along with the degrees of freedom for each pairwise comparison.

```
. pwcompare hour, small
Pairwise comparisons of marginal linear predictions
Margins: asbalanced
```

	Contrast	Std. err.	df	Unadjusted [95% conf.	interval]
score					
hour					
2 vs 1	.75	.3790033	16.0	-.0533703	1.55337
3 vs 1	3.429111	.3989647	16.4	2.584889	4.273333
4 vs 1	6.272103	.3989647	16.4	5.427881	7.116324
3 vs 2	2.679111	.3989647	16.4	1.834889	3.523333
4 vs 2	5.522103	.3989647	16.4	4.677881	6.366324
4 vs 3	2.842991	.4179739	16.7	1.959774	3.726209

When we compare these results with the large-sample results below, we can see that the confidence interval of hour 2 versus hour 1 changes to include 0. Therefore, after adjusting for small-sample size, we would not reject the hypothesis that the means for hour 1 and hour 2 are equivalent at the 5% significance level.

```
. pwcompare hour
Pairwise comparisons of marginal linear predictions
Margins: asbalanced
```

	Contrast	Std. err.	Unadjusted [95% conf.	interval]
score				
hour				
2 vs 1	.75	.3790033	.0071672	1.492833
3 vs 1	3.429111	.3971529	2.650706	4.207516
4 vs 1	6.272103	.3971529	5.493697	7.050508
3 vs 2	2.679111	.3971529	1.900706	3.457516
4 vs 2	5.522103	.3971529	4.743697	6.300508
4 vs 3	2.842991	.4145085	2.03057	3.655413

Stored results

`pwcompare` with option `small` stores the following in `r()`:

Matrices

`r(L_df)`	degrees of freedom for each margin difference
`r(M_df)`	degrees of freedom for each margin estimate

`pwcompare` with options `post` and `small` stores the following in `e()`:

Matrices

`e(L_df)`	degrees of freedom for each margin difference
`e(M_df)`	degrees of freedom for each margin estimate

Methods and formulas

Methods and formulas are presented under the following headings:

Prediction
Small-sample inference

Prediction

Following the notation defined throughout [ME] **mixed**, BLUPs of random effects $\mathbf{u}$ are obtained as

$$\tilde{\mathbf{u}} = \widetilde{\mathbf{G}}\mathbf{Z}'\widetilde{\mathbf{V}}^{-1}\left(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}\right)$$

where $\widetilde{\mathbf{G}}$ and $\widetilde{\mathbf{V}}$ are $\mathbf{G}$ and $\mathbf{V} = \mathbf{ZGZ}' + \sigma_\epsilon^2\mathbf{R}$ with maximum likelihood (ML) or REML estimates of the variance components plugged in. Standard errors for BLUPs are calculated based on the iterative technique of Bates and Pinheiro (1998, sec. 3.3) for estimating the BLUPs themselves. If estimation is done by REML, these standard errors account for uncertainty in the estimate of $\boldsymbol{\beta}$, while for ML the standard errors treat $\boldsymbol{\beta}$ as known. As such, standard errors of REML-based BLUPs will usually be larger.

Fitted values are given by $\mathbf{X}\widehat{\boldsymbol{\beta}} + \mathbf{Z}\tilde{\mathbf{u}}$, residuals as $\hat{\boldsymbol{\epsilon}} = \mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}} - \mathbf{Z}\tilde{\mathbf{u}}$, and standardized residuals as

$$\hat{\boldsymbol{\epsilon}}_* = \hat{\sigma}_\epsilon^{-1}\widehat{\mathbf{R}}^{-1/2}\hat{\boldsymbol{\epsilon}}$$

If the `relevel(`*levelvar*`)` option is specified, fitted values, residuals, and standardized residuals consider only those random-effects terms up to and including level *levelvar* in the model.

For details concerning the calculation of scores, see *Methods and formulas* in [ME] **mixed**.

Small-sample inference

For small-sample computations performed when the `small` option is used with `test`, `testparm`, `lincom`, `contrast`, or `pwcompare`, see *Denominator degrees of freedom* in *Methods and formulas* of [ME] **mixed**.

References

Baldwin, S. 2019. *Psychological Statistics and Psychometrics Using Stata*. College Station, TX: Stata Press.

Bates, D. M., and J. C. Pinheiro. 1998. "Computational methods for multilevel modelling". In *Technical Memorandum BL0112140-980226-01TM*. Murray Hill, NJ: Bell Labs, Lucent Technologies.

Mitchell, M. N. 2015. *Stata for the Behavioral Sciences*. College Station, TX: Stata Press.

Rabe-Hesketh, S., and A. Skrondal. 2022. *Multilevel and Longitudinal Modeling Using Stata*. 4th ed. College Station, TX: Stata Press.

Also see

[ME] **mixed** — Multilevel mixed-effects linear regression+

[U] 20 Estimation and postestimation commands

Glossary

ANOVA denominator degrees of freedom (DDF) method. This method uses the traditional ANOVA for computing DDF. According to this method, the DDF for a test of a fixed effect of a given variable depends on whether that variable is also included in any of the random-effects equations. For traditional ANOVA models with balanced designs, this method provides exact sampling distributions of the test statistics. For more complex mixed-effects models or with unbalanced data, this method typically leads to poor approximations of the actual sampling distributions of the test statistics.

approximation denominator degrees of freedom (DDF) methods. The Kenward–Roger and Satterthwaite DDF methods are referred to as approximation methods because they approximate the sampling distributions of test statistics using t and F distributions with the DDF specific to the method for complicated mixed-effects models and for simple mixed models with unbalanced data. Also see *exact denominator degrees of freedom (DDF) methods*.

between–within denominator degrees of freedom (DDF) method. See *repeated denominator degrees of freedom (DDF) method*.

BLUPs. BLUPs are best linear unbiased predictions of either random effects or linear combinations of random effects. In linear models containing random effects, these effects are not estimated directly but instead are integrated out of the estimation. Once the fixed effects and variance components have been estimated, you can use these estimates to predict group-specific random effects. These predictions are called BLUPs because they are unbiased and have minimal mean squared errors among all linear functions of the response.

canonical link. Corresponding to each family of distributions in a generalized linear model (GLM) is a canonical link function for which there is a sufficient statistic with the same dimension as the number of parameters in the linear predictor. The use of canonical link functions provides the GLM with desirable statistical properties, especially when the sample size is small.

conditional hazard function. In the context of mixed-effects survival models, the conditional hazard function is the hazard function computed conditionally on the random effects. Even within the same covariate pattern, the conditional hazard function varies among individuals who belong to different random-effects clusters.

conditional hazard ratio. In the context of mixed-effects survival models, the conditional hazard ratio is the ratio of two conditional hazard functions evaluated at different values of the covariates. Unless stated differently, the denominator corresponds to the conditional hazard function at baseline, that is, with all the covariates set to zero.

conditional overdispersion. In a negative binomial mixed-effects model, conditional overdispersion is overdispersion conditional on random effects. Also see *overdispersion*.

containment denominator degrees of freedom (DDF) method. See *ANOVA denominator degrees of freedom (DDF) method*.

continuous-time autoregressive structure. A generalization of the autoregressive structure that allows for unequally spaced and noninteger time values.

covariance structure. In a mixed-effects model, covariance structure refers to the variance–covariance structure of the random effects.

crossed-effects model. A crossed-effects model is a mixed-effects model in which the levels of random effects are not nested. A simple crossed-effects model for cross-sectional time-series data would contain a random effect to control for panel-specific variation and a second random effect to control for time-specific random variation. Rather than being nested within panel, in this model a random effect due to a given time is the same for all panels.

crossed-random effects. See *crossed-effects model*.

EB. See *empirical Bayes*.

empirical Bayes. In generalized linear mixed-effects models, empirical Bayes refers to the method of prediction of the random effects after the model parameters have been estimated. The empirical Bayes method uses Bayesian principles to obtain the posterior distribution of the random effects, but instead of assuming a prior distribution for the model parameters, the parameters are treated as given.

empirical Bayes mean. See *posterior mean*.

empirical Bayes mode. See *posterior mode*.

error covariance, **error covariance structure**. Variance–covariance structure of the errors within the lowest-level group. For example, if you are modeling random effects for classes nested within schools, then error covariance refers to the variance–covariance structure of the observations within classes, the lowest-level groups. With a slight abuse of the terminology, error covariance is sometimes also referred to as residual covariance or residual error covariance in the literature.

exact denominator degrees of freedom (DDF) methods. Residual, repeated, and ANOVA DDF methods are referred to as exact methods because they provide exact t and F sampling distributions of test statistics for special classes of mixed-effects models—linear regression, repeated-measures designs, and traditional ANOVA models—with balanced data. Also see *approximation denominator degrees of freedom (DDF) methods*.

fixed effects. In the context of multilevel mixed-effects models, fixed effects represent effects that are constant for all groups at any level of nesting. In the ANOVA literature, fixed effects represent the levels of a factor for which the inference is restricted to only the specific levels observed in the study. See also *fixed-effects model* in [XT] **Glossary**.

free parameter. Free parameters are parameters that are not defined by a linear form. Free parameters are displayed with a forward slash in front of their names or their equation names.

Gauss–Hermite quadrature. In the context of generalized linear mixed models, Gauss–Hermite quadrature is a method of approximating the integral used in the calculation of the log likelihood. The quadrature locations and weights for individual clusters are fixed during the optimization process.

generalized linear mixed-effects model. A generalized linear mixed-effects model is an extension of a generalized linear model allowing for the inclusion of random deviations (effects).

generalized linear model. The generalized linear model is an estimation framework in which the user specifies a distributional family for the dependent variable and a link function that relates the dependent variable to a linear combination of the regressors. The distribution must be a member of the exponential family of distributions. The generalized linear model encompasses many common models, including linear, probit, and Poisson regression.

GHQ. See *Gauss–Hermite quadrature*.

GLM. See *generalized linear model*.

GLME model. See *generalized linear mixed-effects model*.

GLMM. Generalized linear mixed model. See *generalized linear mixed-effects model*.

hierarchical model. A hierarchical model is one in which successively more narrowly defined groups are nested within larger groups. For example, in a hierarchical model, patients may be nested within doctors who are in turn nested within the hospital at which they practice.

intraclass correlation. In the context of mixed-effects models, intraclass correlation refers to the correlation for pairs of responses at each nested level of the model.

Kenward–Roger denominator degrees of freedom (DDF) method. This method implements the Kenward and Roger (1997) method, which is designed to approximate unknown sampling distributions of test statistics for complex linear mixed-effects models. This method is supported only with restricted maximum-likelihood estimation.

Laplacian approximation. Laplacian approximation is a technique used to approximate definite integrals without resorting to quadrature methods. In the context of mixed-effects models, Laplacian approximation is as a rule faster than quadrature methods at the cost of producing biased parameter estimates of variance components.

Lindstrom–Bates algorithm. An algorithm used by the linearization method.

linear form. A linear combination is what we call a "linear form" as long as you do not refer to its coefficients or any subset of the linear combination anywhere in the expression. Linear forms are beneficial for some nonlinear commands such as `nl` because they make derivative computation faster and more accurate. In contrast to free parameters, parameters of a linear form are displayed without forward slashes in the output. Rather, they are displayed as parameters within an equation whose name is the linear combination name. Also see *Linear forms versus linear combinations* in [ME] **menl**.

linear mixed model. See *linear mixed-effects model*.

linear mixed-effects model. A linear mixed-effects model is an extension of a linear model allowing for the inclusion of random deviations (effects).

linearization log likelihood. Objective function used by the linearization method for optimization. This is the log likelihood of the linear mixed-effects model used to approximate the specified nonlinear mixed-effects model.

linearization method, **Lindstrom–Bates method**. Method developed by Lindstrom and Bates (1990) to approximate for fitting nonlinear mixed-effects models. The linearization method uses a first-order Taylor-series expansion of the specified nonlinear mean function to approximate it with a linear function of fixed and random effects. Thus a nonlinear mixed-effects model is approximated by a linear mixed-effects model, in which the fixed-effects and random-effects design matrices involve derivatives of the nonlinear mean function with respect to fixed effects (coefficients) and random effects, respectively. Also see *Introduction* in [ME] **menl**.

link function. In a generalized linear model or a generalized linear mixed-effects model, the link function relates a linear combination of predictors to the expected value of the dependent variable. In a linear regression model, the link function is simply the identity function.

LME model. See *linear mixed-effects model*.

lowest-level group. The second level of a multilevel model with the observations composing the first level. For example, if you are modeling random effects for classes nested within schools, then classes are the lowest-level groups.

MCAGHQ. See *mode-curvature adaptive Gauss–Hermite quadrature*.

mean–variance adaptive Gauss–Hermite quadrature. In the context of generalized linear mixed models, mean–variance adaptive Gauss–Hermite quadrature is a method of approximating the integral used in the calculation of the log likelihood. The quadrature locations and weights for individual clusters are updated during the optimization process by using the posterior mean and the posterior standard deviation.

mixed model. See *mixed-effects model*.

mixed-effects model. A mixed-effects model contains both fixed and random effects. The fixed effects are estimated directly, whereas the random effects are summarized according to their (co)variances. Mixed-effects models are used primarily to perform estimation and inference on the regression coefficients in the presence of complicated within-subject correlation structures induced by multiple levels of grouping.

mode-curvature adaptive Gauss–Hermite quadrature. In the context of generalized linear mixed models, mode-curvature adaptive Gauss–Hermite quadrature is a method of approximating the integral used in the calculation of the log likelihood. The quadrature locations and weights for individual clusters are updated during the optimization process by using the posterior mode and the standard deviation of the normal density that approximates the log posterior at the mode.

MVAGHQ. See *mean–variance adaptive Gauss–Hermite quadrature*.

named substitutable expression. A named substitutable expression is a substitutable expression defined within `menl`'s `define()` option; see *Substitutable expressions* in [ME] **menl**.

nested random effects. In the context of mixed-effects models, nested random effects refer to the nested grouping factors for the random effects. For example, we may have data on students who are nested in classes that are nested in schools.

NLME model. See *nonlinear mixed-effects model*.

nonlinear mixed-effects model. A model in which the conditional mean function given random effects is a nonlinear function of fixed and random effects. A linear mixed-effects model is a special case of a nonlinear mixed-effects model.

one-level model. A one-level model has no multilevel structure and no random effects. Linear regression is a one-level model.

overdispersion. In count-data models, overdispersion occurs when there is more variation in the data than would be expected if the process were Poisson.

posterior mean. In generalized linear mixed-effects models, posterior mean refers to the predictions of random effects based on the mean of the posterior distribution.

posterior mode. In generalized linear mixed-effects models, posterior mode refers to the predictions of random effects based on the mode of the posterior distribution.

QR decomposition. QR decomposition is an orthogonal-triangular decomposition of an augmented data matrix that speeds up the calculation of the log likelihood; see *Methods and formulas* in [ME] **mixed** for more details.

quadrature. Quadrature is a set of numerical methods to evaluate a definite integral.

random coefficient. In the context of mixed-effects models, a random coefficient is a counterpart to a slope in the fixed-effects equation. You can think of a random coefficient as a randomly varying slope at a specific level of nesting.

random effects. In the context of mixed-effects models, random effects represent effects that may vary from group to group at any level of nesting. In the ANOVA literature, random effects represent the levels of a factor for which the inference can be generalized to the underlying population represented by the levels observed in the study. See also *random-effects model* in [XT] **Glossary**.

random intercept. In the context of mixed-effects models, a random intercept is a counterpart to the intercept in the fixed-effects equation. You can think of a random intercept as a randomly varying intercept at a specific level of nesting.

random-effects substitutable expression. A random-effects substitutable expression is a substitutable expression containing random-effects terms; see *Random-effects substitutable expressions* in [ME] **menl**.

REML. See *restricted maximum likelihood*.

repeated denominator degrees of freedom (DDF) method. This method uses the repeated-measures ANOVA for computing DDF. It is used with balanced repeated-measures designs with spherical correlation error structures. It partitions the residual degrees of freedom into the between-subject degrees of freedom and the within-subject degrees of freedom. The repeated method is supported only with two-level models. For more complex mixed-effects models or with unbalanced data, this method typically leads to poor approximations of the actual sampling distributions of the test statistics.

residual covariance, **residual error covariance**. See *error covariance*.

residual denominator degrees of freedom (DDF) method. This method uses the residual degrees of freedom, $n - \text{rank}(X)$, as the DDF for all tests of fixed effects. For a linear model without random effects with independent and identically distributed errors, the distributions of the test statistics for fixed effects are t or F distributions with the residual DDF. For other mixed-effects models, this method typically leads to poor approximations of the actual sampling distributions of the test statistics.

restricted maximum likelihood. Restricted maximum likelihood is a method of fitting linear mixed-effects models that involves transforming out the fixed effects to focus solely on variance–component estimation.

Satterthwaite denominator degrees of freedom (DDF) method. This method implements a generalization of the Satterthwaite (1946) approximation of the unknown sampling distributions of test statistics for complex linear mixed-effects models. This method is supported only with restricted maximum-likelihood estimation.

substitutable expression. Substitutable expressions are like any other mathematical expressions involving scalars and variables, such as those you would use with Stata's `generate` command, except that the parameters to be estimated are bound in braces. See *Substitutable expressions* in [ME] **menl**.

three-level model. A three-level mixed-effects model has one level of observations and two levels of grouping. Suppose that you have a dataset consisting of patients overseen by doctors at hospitals, and each doctor practices at one hospital. Then a three-level model would contain a set of random effects to control for hospital-specific variation, a second set of random effects to control for doctor-specific random variation within a hospital, and a random-error term to control for patients' random variation.

two-level model. A two-level mixed-effects model has one level of observations and one level of grouping. Suppose that you have a panel dataset consisting of patients at hospitals; a two-level model would contain a set of random effects at the hospital level (the second level) to control for hospital-specific random variation and a random-error term at the observation level (the first level) to control for within-hospital variation.

variance components. In a mixed-effects model, the variance components refer to the variances and covariances of the various random effects.

within-group errors. In a two-level model with observations nested within groups, within-group errors refer to error terms at the observation level. In a higher-level model, they refer to errors within the lowest-level groups.

References

Kenward, M. G., and J. H. Roger. 1997. Small sample inference for fixed effects from restricted maximum likelihood. *Biometrics* 53: 983–997. https://doi.org/10.2307/2533558.

Lindstrom, M. J., and D. M. Bates. 1990. Nonlinear mixed effects models for repeated measures data. *Biometrics* 46: 673–687. https://doi.org/10.2307/2532087.

Satterthwaite, F. E. 1946. An approximate distribution of estimates of variance components. *Biometrics Bulletin* 2: 110–114. https://doi.org/10.2307/3002019.

Subject and author index

See the combined subject index and the combined author index in the *Stata Index*.