Title

stata.com

meglm — Multilevel mixed-effects generalized linear model

Syntax | Menu | Description | Options
Remarks and examples | Stored results | Methods and formulas | References
Also see

Syntax

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

where the syntax of *fe_equation* is

```
[ indepvars ] [ if ] [ in ] [ , 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

```
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

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
`collinear`	keep collinear variables
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim`, `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
`nolrtest`	do not perform likelihood-ratio test comparing with reference model
display_options	control 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
`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
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
ghermite	nonadaptive Gauss–Hermite quadrature
laplace	Laplacian approximation; the default for crossed random-effects models

indepvars 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**.

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

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

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

Menu

Statistics > Multilevel mixed-effects models > Generalized linear models (GLMs)

Description

`meglm` fits multilevel mixed-effects generalized linear models. `meglm` allows a variety of distributions for the response conditional on normally distributed random effects.

Options

Model

`noconstant` suppresses the constant (intercept) term and may be specified for the fixed-effects equation and for any or all of 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*$[k, l]$.

`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*`)`, `collinear`; 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.

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.

`nolrtest` prevents `meglm` from fitting a reference linear regression model and using this model to calculate a likelihood-ratio test comparing the mixed model with ordinary regression. This option may also be specified upon replay to suppress this test from the output.

display_options: `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 and variance adaptive Gauss–Hermite quadrature; `mcaghermite` performs mode and 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 `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)` 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.

`startvalues(iv)` builds on `startvalues(fixedonly)` by using instrumental-variable methods with generalized residuals to obtain variances of random 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 residual 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)`.

`coeflegend`; see [R] **estimation options**.

Remarks and examples

stata.com

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**.

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

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, accepts frequency and sampling weights and allows for modeling of the structure of the residual 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 $\mathbf{G} = \text{Var}(\mathbf{u})$.

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); Demidenko (2004); Hedeker and Gibbons (2006); McCulloch, Searle, and Neuhaus (2008); and Rabe-Hesketh and Skrondal (2012).

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

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

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

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 http://www.stata-press.com/data/r13/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 points =         7
                                                 Wald chi2(1)       =  25337.48
Log likelihood = -1014.9268                      Prob > chi2        =    0.0000
```

weight	Coef.	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 regression:   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: $\widehat{\sigma}_\epsilon^2 = 4.38$. This is the variance of the level-one errors, that is, the residuals.
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 the model from example 2 of `meqrlogit`.

▷ Example 2

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.

We fit a standard logistic regression model, amended to have a random intercept for each district and a random slope on the indicator variable `urban`. We fit the model by typing

```
. use http://www.stata-press.com/data/r13/bangladesh
(Bangladesh Fertility Survey, 1989)
. meglm c_use urban age child* || district: urban, family(bernoulli) link(logit)
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      =      1934
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 points =         7
                                                Wald chi2(5)       =     97.30
Log likelihood = -1205.0025                     Prob > chi2        =    0.0000
```

c_use	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
urban	.7143927	.1513595	4.72	0.000	.4177335	1.011052
age	-.0262261	.0079656	-3.29	0.001	-.0418384	-.0106138
child1	1.128973	.1599347	7.06	0.000	.815507	1.442439
child2	1.363165	.1761804	7.74	0.000	1.017857	1.708472
child3	1.352238	.1815608	7.45	0.000	.9963853	1.708091
_cons	-1.698137	.1505019	-11.28	0.000	-1.993115	-1.403159
district						
var(urban)	.2741013	.2131525			.059701	1.258463
var(_cons)	.2390807	.0857012			.1184191	.4826891

```
LR test vs. logistic regression:     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 $\widehat{\sigma}^2_{u1} = 0.27$ and $\widehat{\sigma}^2_{u0} = 0.24$. You can request a different covariance structure by specifying the `covariance()` option. See *Two-level models* in [ME] **meqrlogit** 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

We use the data from the Television, School, and Family Smoking Prevention and Cessation Project (Flay et al. 1988; Rabe-Hesketh and Skrondal 2012, 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 http://www.stata-press.com/data/r13/tvsfpors
. 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      =      1600
Family:                  ordinal
Link:                      logit
```

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 points =         7
                                                Wald chi2(4)       =    124.39
Log likelihood = -2114.5881                     Prob > chi2        =    0.0000
```

thk	Coef.	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	-0.57	0.570	-.4269815	.2350896
/cut2	1.177478	.1704946	6.91	0.000	.8433151	1.511642
/cut3	2.383672	.1786736	13.34	0.000	2.033478	2.733865
school						
var(_cons)	.0448735	.0425387			.0069997	.2876749
school>class						
var(_cons)	.1482157	.0637521			.063792	.3443674

```
LR test vs. ologit regression:       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 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 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

We use the salamander cross-breeding data from Karim and Zeger (1992) as analyzed in Rabe-Hesketh and Skrondal (2012, chap. 16.10). 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 http://www.stata-press.com/data/r13/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 = -209.32221
Iteration 2:   log likelihood = -209.31084
Iteration 3:   log likelihood = -209.27663
Iteration 4:   log likelihood = -209.27659
Iteration 5:   log likelihood = -209.27659  (backed up)
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)       =     37.54
Log likelihood = -209.27659                     Prob > chi2        =    0.0000
```

y	Odds Ratio	Std. Err.	z	P>|z|	[95% Conf.	Interval]
1.wsm	.4956747	.2146564	-1.62	0.105	.2121174	1.15829
1.wsf	.0548105	.0300131	-5.30	0.000	.0187397	.1603114
wsm#wsf						
1 1	36.17082	22.77918	5.70	0.000	10.52689	124.2844
_cons	2.740043	.9768565	2.83	0.005	1.362368	5.510873
_all>male						
var(_cons)	1.04091	.511001			.3976885	2.724476
_all>female						
var(_cons)	1.174448	.5420751			.4752865	2.902098

```
LR test vs. logistic regression:     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 (2012, chap. 16.10) 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. You can include factor variables in the fixed-effects specification by using standard methods; see [U] **11.4.3 Factor variables**. However, random-effects equations support only the R.*varname* factor specification. For more complex factor specifications (such as interactions) in random-effects equations, use generate to form the variables manually.

❑ 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 residual 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()`.

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)`	significance
`e(ll_c)`	log likelihood, comparison model
`e(chi2_c)`	χ^2, comparison model
`e(df_c)`	degrees of freedom, comparison model
`e(p_c)`	significance, comparison model
`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)	meglm
e(cmdline)	command as typed
e(depvar)	name of dependent variable
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

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 estimator
e(V_modelbased)	model-based variance

Functions

e(sample)	marks estimation sample

Methods and formulas

Methods and formulas are presented under the following headings:

Introduction
Gauss–Hermite quadrature
Adaptive Gauss–Hermite quadrature
Laplacian approximation

Introduction

Without a loss of generality, consider a two-level generalized mixed-effects model

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

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 (MVAGH), mode-curvature adaptive Gauss–Hermite quadrature (MCAGH), and nonadaptive Gauss–Hermite quadrature (GHQ). `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 GHQ approximation to the likelihood is

$$\begin{aligned}\mathcal{L}_j^{\mathrm{GHQ}}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) &= \sum_{k_1=1}^{r} \cdots \sum_{k_q=1}^{r} \left[\exp\left\{\log f(\mathbf{y}_j|\boldsymbol{\eta}_{j\mathbf{k}})\right\} \prod_{p=1}^{q} w_{k_p} \right] \\ &= \sum_{k_1=1}^{r} \cdots \sum_{k_q=1}^{r} \left[\exp\left\{ \sum_{i=1}^{n_j} \log f(y_{ij}|\eta_{ij\mathbf{k}}) \right\} \prod_{p=1}^{q} w_{k_p} \right]\end{aligned}$$

where

$$\boldsymbol{\eta}_{j\mathbf{k}} = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{L}\mathbf{a}_{\mathbf{k}}$$

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 MVAGH quadrature and MCAGH 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 MVAGH approximation to the likelihood is

$$\mathcal{L}_j^{\mathrm{MVAGH}}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \sum_{k_1=1}^{r} \cdots \sum_{k_q=1}^{r} \left[\exp\left\{\log f(\mathbf{y}_j|\boldsymbol{\eta}_{j\mathbf{k}})\right\} \prod_{p=1}^{q} w^*_{jk_p} \right]$$

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 MVAGH approximation, starting with a $\mathbf{0}$ mean vector and identity variance matrix. For the MCAGH 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 $\widehat{\mathbf{u}}_j$ such that $h'(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \widehat{\mathbf{u}}_j) = \mathbf{0}$. The integral in (3) is proportional to the posterior density of $\mathbf{u}_j$ given the data, so $\widehat{\mathbf{u}}_j$ is also the posterior mode.

Let

$$\widehat{\mathbf{p}}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\widehat{\mathbf{u}}_j$$

$$\mathbf{S}_1 = \frac{\partial \log f(\mathbf{y}_j|\widehat{\mathbf{p}}_j)}{\partial \widehat{\mathbf{p}}_j}$$

$$\mathbf{S}_2 = \frac{\partial \mathbf{S}_1}{\partial \widehat{\mathbf{p}}_j'} = \frac{\partial^2 \log f(\mathbf{y}_j|\widehat{\mathbf{p}}_j)}{\partial \widehat{\mathbf{p}}_j \partial \widehat{\mathbf{p}}_j'}$$

$$\mathbf{H}_j = h''(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \widehat{\mathbf{u}}_j) = \mathbf{Z}_j'\mathbf{S}_2\mathbf{Z}_j - \boldsymbol{\Sigma}^{-1}$$

then

$$\mathbf{0} = h'(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \widehat{\mathbf{u}}_j) = \mathbf{Z}_j'\mathbf{S}_1 - \boldsymbol{\Sigma}^{-1}\widehat{\mathbf{u}}_j$$

Given the above, the second-order Taylor approximation takes the form

$$h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j) \approx h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \widehat{\mathbf{u}}_j) + \frac{1}{2}(\mathbf{u}_j - \widehat{\mathbf{u}}_j)' \mathbf{H}_j (\mathbf{u}_j - \widehat{\mathbf{u}}_j)$$

because the first-order derivative term is 0. The integral is approximated by

$$\int_{\Re^q} \exp\{h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \mathbf{u}_j)\}\, d\mathbf{u}_j \approx (2\pi)^{q/2}\, |-\mathbf{H}_j|^{-1/2} \exp\{h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \widehat{\mathbf{u}}_j)\}$$

Thus the Laplacian approximated log likelihood is

$$\log \mathcal{L}_j^{\mathrm{Lap}}(\boldsymbol{\beta}, \boldsymbol{\Sigma}) = -\frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} \log |-\mathbf{H}_j| + h(\boldsymbol{\beta}, \boldsymbol{\Sigma}, \widehat{\mathbf{u}}_j)$$

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.

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. http://stat.bell-labs.com/NLME/CompMulti.pdf.

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

Cameron, A. C., and P. K. Trivedi. 2010. *Microeconometrics Using Stata*. Rev. ed. College Station, TX: Stata Press.

Canette, I. 2011. Including covariates in crossed-effects models. The Stata Blog: Not Elsewhere Classified. http://blog.stata.com/2010/12/22/including-covariates-in-crossed-effects-models/.

Demidenko, E. 2004. *Mixed Models: Theory and Applications*. Hoboken, NJ: Wiley.

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.

Harville, D. A. 1977. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association* 72: 320–338.

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.

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

LaMotte, L. R. 1973. Quadratic estimation of variance components. *Biometrics* 29: 311–330.

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.

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.

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.

Rabe-Hesketh, S., and A. Skrondal. 2012. *Multilevel and Longitudinal Modeling Using Stata*. 3rd 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.

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.

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

Verbeke, G., and G. Molenberghs. 2000. *Linear Mixed Models for Longitudinal Data*. New York: Springer.

Also see

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

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

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

[R] **glm** — Generalized linear models

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

[U] **20 Estimation and postestimation commands**