Title

meprobit — Multilevel mixed-effects probit regression

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
```

As above, but specify that `y` records the number of successes from 10 trials

```
meprobit y x || lev2: x, binomial(10)
```

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

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
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)
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 probit regression
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
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

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` and `svy` are allowed; see **[U] 11.1.10 Prefix commands**.

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

`fweight`s, `iweight`s, and `pweight`s 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`, 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*), 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(*#*), 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 meprobit from performing a likelihood-ratio test that compares the mixed-effects probit model with standard (marginal) probit regression. This option may also be specified upon replay to suppress this test from the output.

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**.

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

Remarks and examples

stata.com

For a general introduction to `me` commands, see [ME] **me**.

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

Remarks are presented under the following headings:

Introduction

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.

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). Guo and Zhao (2000) and Rabe-Hesketh and Skrondal (2012, chap. 10) are good introductory readings on applied multilevel modeling of binary data.

`meprobit` allows for not just one, but many levels of nested clusters of random effects. For example, in a three-level model you can specify random effects for schools and then random effects for classes nested within schools. In this model, the observations (presumably, the students) comprise the first level, the classes comprise the second level, and the schools comprise the third.

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 $j = 1, \dots, M$ clusters, with cluster j consisting of $i = 1, \dots, n_j$ 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 $\mathbf{u}_j$.

Model (1) is an example of a generalized linear mixed model (GLMM), which generalizes the linear mixed-effects (LME) model to non-Gaussian responses. You can fit LMEs in Stata by using `mixed` and fit GLMMs by using `meglm`. Because of the relationship between LMEs and GLMMs, there is insight to be gained through examination of the linear mixed model. This is especially true for Stata users because the terminology, syntax, options, and output for fitting these types of models are nearly identical. See [ME] **mixed** and the references therein, particularly in *Introduction*, for more information.

Log-likelihood calculations for fitting any generalized mixed-effects model require integrating out the random effects. 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 likelihood-ratio tests for comparing nested models. Also, if done correctly, quadrature approximations can be quite accurate, thus minimizing bias.

`meprobit` supports three types of Gauss–Hermite quadrature and the Laplacian approximation method; see *Methods and formulas* of [ME] **meglm** for details. The simplest random-effects model you can fit using `meprobit` is the two-level model with a random intercept,

$$\Pr(y_{ij} = 1|\mathbf{u}_j) = \Phi(\mathbf{x}_{ij}\boldsymbol{\beta} + u_j)$$

This model can also be fit using `xtprobit` with the `re` option; see [XT] **xtprobit**.

Below we present two short examples of mixed-effects probit regression; refer to [ME] **melogit** for additional examples including crossed random-effects models and to [ME] **me** and [ME] **meglm** for examples of other random-effects models.

Two-level models

We begin with a simple application of (1) as a two-level model, because a one-level model, in our terminology, is just standard probit regression; see [R] **probit**.

▷ Example 1

In example 1 of [ME] **melogit**, we 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. Here we refit that model with `meprobit`:

```
. use http://www.stata-press.com/data/r14/bangladesh
(Bangladesh Fertility Survey, 1989)
. meprobit c_use urban age child* || 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	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
urban	.4490191	.0727176	6.17	0.000	.3064953	.5915429
age	-.0162203	.0048005	-3.38	0.001	-.0256291	-.0068114
child1	.674377	.0947829	7.11	0.000	.488606	.8601481
child2	.8281581	.1048136	7.90	0.000	.6227272	1.033589
child3	.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
```

Comparing the estimates of `meprobit` with those of `melogit`, we observe the familiar result where the probit estimates are closer to 0 in absolute value due to the smaller variance of the error term in the probit model. Example 1 of [ME] **meprobit postestimation** shows that the marginal effect of covariates is nearly the same between the two models.

Unlike a logistic regression, coefficients from a probit regression cannot be interpreted in terms of odds ratios. Most commonly, probit regression coefficients are 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.

◁

Three-level models

Two-level models extend naturally to models with three or more levels with nested random effects. Below we replicate example 2 of [ME] **melogit** with `meprobit`.

▷ Example 2

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.

```
. use http://www.stata-press.com/data/r14/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

                  No. of       Observations per Group
 Group Variable   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)      =      83.28
Log likelihood = -304.88845                     Prob > chi2       =     0.0000

        dtlm       Coef.   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.
```

Notes:

1. 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—`meprobit` assumes that `subject` is nested within `family`.

2. 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. You can suppress this table with the `nogroup` or the `noheader` option, which will suppress the rest of the header as well.

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 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).

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)`	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(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

Methods and formulas

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, $j = 1, \dots, M$, the conditional distribution of $\mathbf{y}_j = (y_{j1}, \dots, y_{jn_j})'$, given a set of cluster-level random effects $\mathbf{u}_j$, 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

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

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

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

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 $f(\mathbf{y}_j, \mathbf{u}_j)$,

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

where

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

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. `meprobit` offers four approximation methods: mean–variance adaptive Gauss–Hermite quadrature (default unless a crossed random-effects model is fit), mode-curvature adaptive Gauss–Hermite quadrature, nonadaptive Gauss–Hermite quadrature, and Laplacian approximation (default for crossed random-effects models).

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; see *Methods and formulas* in [ME] **meglm** for details.

Gaussian quadrature relies on transforming the multivariate integral in (2) into a set of nested univariate integrals. Each univariate integral can then be evaluated using a form of Gaussian quadrature; see *Methods and formulas* in [ME] **meglm** for details.

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)`.

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

References

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

Guo, G., and H. Zhao. 2000. Multilevel modeling of binary data. *Annual Review of Sociology* 26: 441–462.

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

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

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

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

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

Searle, S. R., G. Casella, and C. E. McCulloch. 1992. *Variance Components*. New York: Wiley.

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

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

[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**