# Title



**meoprobit** — Multilevel mixed-effects ordered probit regression



## Description

`meoprobit` fits mixed-effects probit models for ordered responses. The actual values taken on by the response are irrelevant except that larger values are assumed to correspond to "higher" outcomes. The conditional distribution of the response given the random effects is assumed to be multinomial, with success probability determined by the standard normal cumulative distribution function.

## Quick start

Two-level ordered probit regression of `y` on `x` and random intercepts by `lev2`

```
meoprobit y x || lev2:
```

Add random coefficients for `x`

```
meoprobit y x || lev2: x
```

Nested three-level ordered probit model with random intercepts by `lev2` and `lev3` for `lev2` nested within `lev3`

```
meoprobit y x || lev3: || lev2:
```

## Menu

Statistics > Multilevel mixed-effects models > Ordered probit regression

# Syntax

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

where the syntax of *fe_equation* is

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

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

for random coefficients and intercepts

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

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

*levelvar*: `R.`*varname*

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

| *fe_options* | Description |
|---|---|
| Model | |
| `offset(`*varname*`)` | include *varname* in model with coefficient constrained to 1 |

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

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

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

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

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

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

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

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

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

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

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

# Options

Model

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

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

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

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

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

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

covariance(fixed(*matname*)) and covariance(pattern(*matname*)) covariance structures provide a convenient way to impose constraints on variances and covariances of random effects. Each specification requires a *matname* that defines the restrictions placed on variances and covariances. Only elements in the lower triangle of *matname* are used, and row and column names of *matname* are ignored. A missing value in *matname* means that a given element is unrestricted. In a fixed(*matname*) covariance structure, (co)variance $(i,j)$ is constrained to equal the value specified in the $i,j$th entry of *matname*. In a pattern(*matname*) covariance structure, (co)variances $(i,j)$ and $(k,l)$ are constrained to be equal if *matname*$[i,j]$ = *matname*$[k,l]$.

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

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

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

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

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

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

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

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

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

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

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

SE/Robust

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

Reporting

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

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

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

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

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

Integration

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

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

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

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

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

Maximization

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

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

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

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

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

## Remarks and examples

stata.com

Mixed-effects ordered probit regression is ordered probit regression containing both fixed effects and random effects. An ordered response is a variable that is categorical and ordered, for instance, "poor", "good", and "excellent", which might indicate a person's current health status or the repair record of a car.

meoprobit allows for many levels of random effects. However, for simplicity, for now we consider the two-level model, where for a series of $M$ independent clusters, and conditional on a set of fixed effects $\mathbf{x}_{ij}$, a set of cutpoints $\boldsymbol{\kappa}$, and a set of random effects $\mathbf{u}_j$, the cumulative probability of the response being in a category higher than $k$ is

$$\Pr(y_{ij} > k|\mathbf{x}_{ij}, \boldsymbol{\kappa}, \mathbf{u}_j) = \Phi(\mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j - \kappa_k) \tag{1}$$

for $j = 1, \dots, M$ clusters, with cluster $j$ consisting of $i = 1, \dots, n_j$ observations. The cutpoints are labeled $\kappa_1$, $\kappa_2$, $\dots$, $\kappa_{K-1}$, where $K$ is the number of possible outcomes. $\Phi(\cdot)$ is the standard normal cumulative distribution function that represents cumulative probability.

The $1 \times p$ row vector $\mathbf{x}_{ij}$ are the covariates for the fixed effects, analogous to the covariates you would find in a standard probit regression model, with regression coefficients (fixed effects) $\boldsymbol{\beta}$. In our parameterization, $\mathbf{x}_{ij}$ does not contain a constant term because its effect is absorbed into the cutpoints. For notational convenience here and throughout this manual entry, we suppress the dependence of $y_{ij}$ on $\mathbf{x}_{ij}$.

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

From (1), we can derive the probability of observing outcome $k$ as

$$\begin{aligned}
\Pr(y_{ij} = k|\boldsymbol{\kappa}, \mathbf{u}_j) &= \Pr(\kappa_{k-1} < \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \epsilon_{ij} \le \kappa_k) \\
&= \Pr(\kappa_{k-1} - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j < \epsilon_{ij} \le \kappa_k - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j) \\
&= \Phi(\kappa_k - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j) - \Phi(\kappa_{k-1} - \mathbf{x}_{ij}\boldsymbol{\beta} - \mathbf{z}_{ij}\mathbf{u}_j)
\end{aligned}$$

where $\kappa_0$ is taken as $-\infty$ and $\kappa_K$ is taken as $+\infty$.

From the above, we may also write the model in terms of a latent linear response, where observed ordinal responses $y_{ij}$ are generated from the latent continuous responses, such that

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

and

$$y_{ij} = \begin{cases} 1 & \text{if} \quad y_{ij}^* \le \kappa_1 \\ 2 & \text{if} \quad \kappa_1 < y_{ij}^* \le \kappa_2 \\ \vdots & \\ K & \text{if} \quad \kappa_{K-1} < y_{ij}^* \end{cases}$$

The errors $\epsilon_{ij}$ are distributed as standard normal with mean 0 and variance 1 and are independent of $\mathbf{u}_j$.

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

## ▷ Example 1: Two-level random-intercept model

We use the data from the Television, School, and Family Smoking Prevention and Cessation Project (Flay et al. 1988; Rabe-Hesketh and Skrondal 2022, chap. 11), where schools were randomly assigned into one of four groups defined by two treatment variables. Students within each school are nested in classes, and classes are nested in schools. In this example, we ignore the variability of classes within schools and fit a two-level model; we incorporate classes in a three-level model in example 2. The dependent variable is the tobacco and health knowledge (THK) scale score collapsed into four ordered categories. We regress the outcome on the treatment variables and their interaction and control for the pretreatment score.

```
. use https://www.stata-press.com/data/r18/tvsfpors
(Television, School, and Family Project)
. meoprobit thk prethk cc##tv || school:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood = -2127.8111
Iteration 2:  Log likelihood = -2127.7612
Iteration 3:  Log likelihood = -2127.7612
Refining starting values:
Grid node 0:  Log likelihood = -2149.7302
Fitting full model:
Iteration 0:  Log likelihood = -2149.7302  (not concave)
Iteration 1:  Log likelihood = -2129.6838  (not concave)
Iteration 2:  Log likelihood = -2123.5143
Iteration 3:  Log likelihood = -2122.2896
Iteration 4:  Log likelihood = -2121.7949
Iteration 5:  Log likelihood = -2121.7716
Iteration 6:  Log likelihood = -2121.7715
Mixed-effects oprobit regression                Number of obs     =      1,600
Group variable: school                          Number of groups  =         28
                                                Obs per group:
                                                              min =         18
                                                              avg =       57.1
                                                              max =        137
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(4)      =     128.05
Log likelihood = -2121.7715                     Prob > chi2       =     0.0000
```

| thk | Coefficient | Std. err. | z | P>\|z\| | [95% conf. interval] | |
|---|---|---|---|---|---|---|
| prethk | .2369804 | .0227739 | 10.41 | 0.000 | .1923444 | .2816164 |
| 1.cc | .5490957 | .1255108 | 4.37 | 0.000 | .303099 | .7950923 |
| 1.tv | .1695405 | .1215889 | 1.39 | 0.163 | -.0687693 | .4078504 |
| cc#tv | | | | | | |
| 1 1 | -.2951837 | .1751969 | -1.68 | 0.092 | -.6385634 | .0481959 |
| /cut1 | -.0682011 | .1003374 | | | -.2648587 | .1284565 |
| /cut2 | .67681 | .1008836 | | | .4790817 | .8745382 |
| /cut3 | 1.390649 | .1037494 | | | 1.187304 | 1.593995 |
| school | | | | | | |
| var(_cons) | .0288527 | .0146201 | | | .0106874 | .0778937 |

```
LR test vs. oprobit model: chibar2(01) = 11.98        Prob >= chibar2 = 0.0003
```

The estimation table reports the fixed effects, the estimated cutpoints $(\kappa_1, \kappa_2, \kappa_3)$, and the estimated variance components. The fixed effects can be interpreted just as you would the output from `oprobit`. We find that students with higher preintervention scores tend to have higher postintervention scores. Because of their interaction, the impact of the treatment variables on the knowledge score is not straightforward; we defer this discussion to example 1 of [ME] **meoprobit postestimation**.

Underneath the fixed effects and the cutpoints, the table shows the estimated variance components. The random-effects equation is labeled `school`, meaning that these are random effects at the `school` level. Because we have only one random effect at this level, the table shows only one variance component. The estimate of $\sigma_u^2$ is 0.03 with standard error 0.01. The reported likelihood-ratio test shows that there is enough variability between schools to favor a mixed-effects ordered probit regression over a standard ordered probit regression; see *Distribution theory for likelihood-ratio test* in [ME] **me** for a discussion of likelihood-ratio testing of variance components.

We now store our estimates for later use.

```
. estimates store r_2
```

◁

Two-level models extend naturally to models with three or more levels with nested random effects. Below we continue with example 1.

## ▷ Example 2: Three-level random-intercept model

In this example, we fit a three-level model incorporating classes nested within schools. The fixed-effects part remains the same. Our model now has two random-effects equations, separated by `||`. The first is a random intercept (constant only) at the `school` level (level three), and the second is a random intercept at the `class` level (level two). The order in which these are specified (from left to right) is significant—`meoprobit` assumes that `class` is nested within `school`.

```
. meoprobit thk prethk cc##tv || school: || class:
Fitting fixed-effects model:
Iteration 0:  Log likelihood =  -2212.775
Iteration 1:  Log likelihood = -2127.8111
Iteration 2:  Log likelihood = -2127.7612
Iteration 3:  Log likelihood = -2127.7612
Refining starting values:
Grid node 0:  Log likelihood = -2195.6424
Fitting full model:
Iteration 0:  Log likelihood = -2195.6424  (not concave)
Iteration 1:  Log likelihood = -2167.9576  (not concave)
Iteration 2:  Log likelihood = -2140.2644  (not concave)
Iteration 3:  Log likelihood = -2128.6948  (not concave)
Iteration 4:  Log likelihood = -2119.9225
Iteration 5:  Log likelihood = -2117.0947
Iteration 6:  Log likelihood = -2116.7004
Iteration 7:  Log likelihood = -2116.6981
Iteration 8:  Log likelihood = -2116.6981
Mixed-effects oprobit regression                Number of obs     =      1,600
        Grouping information
```

| Group variable | No. of groups | Observations per group Minimum | Average | Maximum |
|---|---|---|---|---|
| school | 28 | 18 | 57.1 | 137 |
| class | 135 | 1 | 11.9 | 28 |

```
Integration method: mvaghermite                 Integration pts.  =          7
                                                Wald chi2(4)      =     124.20
Log likelihood = -2116.6981                     Prob > chi2       =     0.0000
```

| thk | Coefficient | Std. err. | z | P>\|z\| | [95% conf. | interval] |
|---|---|---|---|---|---|---|
| prethk | .238841 | .0231446 | 10.32 | 0.000 | .1934784 | .2842036 |
| 1.cc | .5254813 | .1285816 | 4.09 | 0.000 | .2734659 | .7774967 |
| 1.tv | .1455573 | .1255827 | 1.16 | 0.246 | -.1005803 | .3916949 |
| cc#tv | | | | | | |
| 1 1 | -.2426203 | .1811999 | -1.34 | 0.181 | -.5977656 | .1125251 |
| /cut1 | -.074617 | .1029791 | | | -.2764523 | .1272184 |
| /cut2 | .6863046 | .1034813 | | | .4834849 | .8891242 |
| /cut3 | 1.413686 | .1064889 | | | 1.204972 | 1.622401 |
| school | | | | | | |
| var(_cons) | .0186456 | .0160226 | | | .0034604 | .1004695 |
| school>class | | | | | | |
| var(_cons) | .0519974 | .0224014 | | | .0223496 | .1209745 |

```
LR test vs. oprobit model: chi2(2) = 22.13            Prob > chi2 = 0.0000
Note: LR test is conservative and provided only for reference.
```

We see that we have 135 classes from 28 schools. The variance-component estimates are now organized and labeled according to level. The variance component for `class` is labeled `school>class` to emphasize that classes are nested within schools.

Compared with the two-level model from example 1, the estimate of the random intercept at the school level dropped from 0.03 to 0.02. This is not surprising because we now use two random components versus one random component to account for unobserved heterogeneity among students. We can use lrtest and our stored estimation result from example 1 to see which model provides a better fit:

```
. lrtest r_2 .
Likelihood-ratio test
Assumption: r_2 nested within .
 LR chi2(1) =  10.15
Prob > chi2 = 0.0014
Note: The reported degrees of freedom assumes the null hypothesis is not on
      the boundary of the parameter space. If this is not true, then the
      reported test is conservative.
```

The likelihood-ratio test favors the three-level model. For more information about the likelihood-ratio test in the context of mixed-effects models, see *Distribution theory for likelihood-ratio test* in [ME] **me**.

◁

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

## Stored results

meoprobit stores the following in e():

Scalars

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

Macros

| | |
|---|---|
| e(cmd) | meglm |
| e(cmd2) | meoprobit |
| e(cmdline) | command as typed |
| e(depvar) | name of dependent variable |
| e(wtype) | weight type |
| e(wexp) | weight expression (first-level weights) |
| e(fweight*k*) | fweight variable for *k*th highest level, if specified |
| e(iweight*k*) | iweight variable for *k*th highest level, if specified |

| | |
|---|---|
| `e(pweightk)` | `pweight` variable for *k*th highest level, if specified |
| `e(covariates)` | list of covariates |
| `e(ivars)` | grouping variables |
| `e(model)` | `oprobit` |
| `e(title)` | title in estimation output |
| `e(link)` | `probit` |
| `e(family)` | `ordinal` |
| `e(clustvar)` | name of cluster variable |
| `e(offset)` | offset |
| `e(intmethod)` | integration method |
| `e(n_quad)` | number of integration points |
| `e(chi2type)` | `Wald`; type of model $\chi^2$ |
| `e(vce)` | *vcetype* specified in `vce()` |
| `e(vcetype)` | title used to label Std. err. |
| `e(opt)` | type of optimization |
| `e(which)` | `max` or `min`; whether optimizer is to perform maximization or minimization |
| `e(ml_method)` | type of `ml` method |
| `e(user)` | name of likelihood-evaluator program |
| `e(technique)` | maximization technique |
| `e(datasignature)` | the checksum |
| `e(datasignaturevars)` | variables used in calculation of checksum |
| `e(properties)` | `b V` |
| `e(estat_cmd)` | program used to implement `estat` |
| `e(predict)` | program used to implement `predict` |
| `e(marginsnotok)` | predictions disallowed by `margins` |
| `e(marginswtype)` | weight type for `margins` |
| `e(marginswexp)` | weight expression for `margins` |
| `e(marginsdefault)` | default `predict()` specification for `margins` |
| `e(asbalanced)` | factor variables `fvset` as `asbalanced` |
| `e(asobserved)` | factor variables `fvset` as `asobserved` |

Matrices

| | |
|---|---|
| `e(b)` | coefficient vector |
| `e(Cns)` | constraints matrix |
| `e(ilog)` | iteration log (up to 20 iterations) |
| `e(gradient)` | gradient vector |
| `e(N_g)` | group counts |
| `e(g_min)` | group-size minimums |
| `e(g_avg)` | group-size averages |
| `e(g_max)` | group-size maximums |
| `e(cat)` | category values |
| `e(V)` | variance–covariance matrix of the estimators |
| `e(V_modelbased)` | model-based variance |

Functions

| | |
|---|---|
| `e(sample)` | marks estimation sample |

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

Matrices

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

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

# Methods and formulas

`meoprobit` is a convenience command for `meglm` with a `probit` link and an `ordinal` family; see [ME] **meglm**.

Without a loss of generality, consider a two-level ordered probit model. The probability of observing outcome $k$ for response $y_{ij}$ is then

$$
\begin{aligned}
p_{ij} = \Pr(y_{ij} = k|\boldsymbol{\kappa}, \mathbf{u}_j) &= \Pr(\kappa_{k-1} < \boldsymbol{\eta}_{ij} + \epsilon_{it} \le \kappa_k) \\
&= \Phi(\kappa_k - \boldsymbol{\eta}_{ij}) - \Phi(\kappa_{k-1} - \boldsymbol{\eta}_{ij})
\end{aligned}
$$

where $\boldsymbol{\eta}_{ij} = \mathbf{x}_{ij}\boldsymbol{\beta} + \mathbf{z}_{ij}\mathbf{u}_j + \text{offset}_{ij}$, $\kappa_0$ is taken as $-\infty$, and $\kappa_K$ is taken as $+\infty$. Here $\mathbf{x}_{ij}$ does not contain a constant term because its effect is absorbed into the cutpoints.

For cluster $j$, $j = 1, \dots, M$, 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

$$
\begin{aligned}
f(\mathbf{y}_j|\mathbf{u}_j) &= \prod_{i=1}^{n_j} p_{ij}^{I_k(y_{ij})} \\
&= \exp \sum_{i=1}^{n_j} \left\{ I_k(y_{ij}) \log(p_{ij}) \right\}
\end{aligned}
$$

where

$$
I_k(y_{ij}) = \begin{cases} 1 & \text{if } y_{ij} = k \\ 0 & \text{otherwise} \end{cases}
$$

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

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

where

$$
h\left(\boldsymbol{\beta}, \boldsymbol{\kappa}, \boldsymbol{\Sigma}, \mathbf{u}_j\right) = \sum_{i=1}^{n_j} \left\{ I_k(y_{ij}) \log(p_{ij}) \right\} - \mathbf{u}_j'\boldsymbol{\Sigma}^{-1}\mathbf{u}_j/2
$$

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

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

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

# References


Andrews, M. J., T. Schank, and R. Upward. 2006. Practical fixed-effects estimation methods for the three-way error-components model. *Stata Journal* 6: 461–481.

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

Harbord, R. M., and P. Whiting. 2009. metandi: Meta-analysis of diagnostic accuracy using hierarchical logistic regression. *Stata Journal* 9: 211–229.

Joe, H. 2008. Accuracy of Laplace approximation for discrete response mixed models. *Computational Statistics and Data Analysis* 52: 5066–5074. https://doi.org/10.1016/j.csda.2008.05.002.

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

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

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

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

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

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

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


# Also see

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

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

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

[BAYES] **bayes: meoprobit** — Bayesian multilevel ordered probit regression

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

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

[XT] **xtoprobit** — Random-effects ordered probit models

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



For suggested citations, see the FAQ on citing Stata documentation.