# mlogit postestimation — Postestimation tools for mlogit

The following postestimation commands are available after `mlogit`:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>contrast</code></td>
<td>contrasts and ANOVA-style joint tests of estimates</td>
</tr>
<tr>
<td><code>estat ic</code></td>
<td>Akaike's and Schwarz's Bayesian information criteria (AIC and BIC)</td>
</tr>
<tr>
<td><code>estat summarize</code></td>
<td>summary statistics for the estimation sample</td>
</tr>
<tr>
<td><code>estat vce</code></td>
<td>variance–covariance matrix of the estimators (VCE)</td>
</tr>
<tr>
<td><code>estat (svy)</code></td>
<td>postestimation statistics for survey data</td>
</tr>
<tr>
<td><code>estimates</code></td>
<td>cataloging estimation results</td>
</tr>
<tr>
<td><code>etable</code></td>
<td>table of estimation results</td>
</tr>
<tr>
<td><code>*forecast</code></td>
<td>dynamic forecasts and simulations</td>
</tr>
<tr>
<td><code>*hausman</code></td>
<td>Hausman's specification test</td>
</tr>
<tr>
<td><code>lincom</code></td>
<td>point estimates, standard errors, testing, and inference for linear combinations of coefficients</td>
</tr>
<tr>
<td><code>*lrtest</code></td>
<td>likelihood-ratio test</td>
</tr>
<tr>
<td><code>margins</code></td>
<td>marginal means, predictive margins, marginal effects, and average marginal effects</td>
</tr>
<tr>
<td><code>marginsplot</code></td>
<td>graph the results from margins (profile plots, interaction plots, etc.)</td>
</tr>
<tr>
<td><code>nlcom</code></td>
<td>point estimates, standard errors, testing, and inference for nonlinear combinations of coefficients</td>
</tr>
<tr>
<td><code>predict</code></td>
<td>probabilities, linear predictions and their SEs, etc.</td>
</tr>
<tr>
<td><code>predictnl</code></td>
<td>point estimates, standard errors, testing, and inference for generalized predictions</td>
</tr>
<tr>
<td><code>pwcompare</code></td>
<td>pairwise comparisons of estimates</td>
</tr>
<tr>
<td><code>suest</code></td>
<td>seemingly unrelated estimation</td>
</tr>
<tr>
<td><code>test</code></td>
<td>Wald tests of simple and composite linear hypotheses</td>
</tr>
<tr>
<td><code>testnl</code></td>
<td>Wald tests of nonlinear hypotheses</td>
</tr>
</tbody>
</table>

*`forecast`, `hausman`, and `lrtest` are not appropriate with `svy` estimation results. `forecast` is also not appropriate with `mi` estimation results.*
predict

Description for predict

predict creates a new variable containing predictions such as probabilities, linear predictions, and standard errors.

Menu for predict

Statistics  >  Postestimation

Syntax for predict

```
predict [type] { stub* | newvar | newvarlist } [if] [in] [, statistic outcome(outcome)]
```

```
predict [type] stub* [if] [in], scores
```

<table>
<thead>
<tr>
<th>statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr</td>
<td>predicted probabilities; the default</td>
</tr>
<tr>
<td>xb</td>
<td>linear prediction</td>
</tr>
<tr>
<td>stdp</td>
<td>standard error of the linear prediction</td>
</tr>
<tr>
<td>stddp</td>
<td>standard error of the difference in two linear predictions</td>
</tr>
</tbody>
</table>

You specify one or k new variables with pr, where k is the number of outcomes. If you specify one new variable and you do not specify outcome(), then outcome(#1) is assumed.

You specify one new variable with xb, stdp, and stddp. If you do not specify outcome(), then outcome(#1) is assumed. You must specify outcome() with the stddp option.

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

Options for predict

- **pr**, the default, computes the predicted probabilities for all outcomes or for a specific outcome. To compute probabilities for all outcomes, you specify k new variables, where k is the number of categories of the dependent variable. Alternatively, you can specify stub*; in which case, pr will store predicted probabilities in variables stub1, stub2, ..., stubk. To compute the probability for a specific outcome, you specify one new variable and, optionally, the outcome value in option outcome(); if you omit outcome(), the first outcome value, outcome(#1), is assumed.

Say that you fit a model by typing estimation_cmd y x1 x2, and y takes on four values. Then, you could type predict p1 p2 p3 p4 to obtain all four predicted probabilities; alternatively, you could type predict p* to generate the four predicted probabilities. To compute specific probabilities one at a time, you can type predict p1, outcome(#1) (or simply predict p1), predict p2, outcome(#2), and so on. See option outcome() for other ways to refer to outcome values.

- **xb** calculates the linear prediction. You must also specify the outcome(outcome) option.
stdp calculates the standard error of the linear prediction. You must also specify the outcome(outcome) option.

stddp calculates the standard error of the difference in two linear predictions. You must specify the outcome(outcome) option, and here you specify the two particular outcomes of interest inside the parentheses, for example, predict sed, stddp outcome(1,3).

outcome(outcome) specifies for which outcome the predicted probabilities are to be calculated. outcome() should contain either one value of the dependent variable or one of #1, #2, ..., with #1 meaning the first category of the dependent variable, #2 meaning the second category, etc. outcome() is not allowed with scores.

scores calculates equation-level score variables. The number of score variables created will be one less than the number of outcomes in the model. If the number of outcomes in the model were k, then

- the first new variable will contain $\frac{\partial \ln L}{\partial (x_j \beta_1)}$;
- the second new variable will contain $\frac{\partial \ln L}{\partial (x_j \beta_2)}$;
- ...
- the $(k - 1)$th new variable will contain $\frac{\partial \ln L}{\partial (x_j \beta_{k-1})}$.

margins

Description for margins

margins estimates margins of response for probabilities and linear predictions.

Menu for margins

Statistics > Postestimation

Syntax for margins

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

<table>
<thead>
<tr>
<th>statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>probabilities for each outcome</td>
</tr>
<tr>
<td>pr</td>
<td>probability for a specified outcome</td>
</tr>
<tr>
<td>xb</td>
<td>linear prediction for a specified outcome</td>
</tr>
<tr>
<td>stdp</td>
<td>not allowed with margins</td>
</tr>
<tr>
<td>stddp</td>
<td>not allowed with margins</td>
</tr>
</tbody>
</table>

pr and xb default to the first outcome.

Statistics not allowed with margins are functions of stochastic quantities other than $e(b)$.

For the full syntax, see [R] margins.
Remarks and examples

Remarks are presented under the following headings:

- Obtaining predicted values
- Calculating marginal effects
- Testing hypotheses about coefficients

Obtaining predicted values

Example 1: Obtaining predicted probabilities

After estimation, we can use `predict` to obtain predicted probabilities, index values, and standard errors of the index, or differences in the index. For instance, in example 4 of [R] `mlogit`, we fit a model of insurance choice on various characteristics. We can obtain the predicted probabilities for outcome 1 by typing

```stata
. use https://www.stata-press.com/data/r17/sysdsn1
    (Health insurance data)
. mlogit insure age i.male i.nonwhite i.site
    (output omitted)
. predict p1 if e(sample), outcome(1)
    (option `pr' assumed; predicted probability)
    (29 missing values generated)
. summarize p1
```

```
Variable |       Obs        Mean    Std. dev.      Min      Max
----------|-------------|-----------|-------------|---------|---------|
p1        |       615 | .4764228 | .1032279 | .1698142 | .71939 |
```

We added the `i.` prefix to the `male`, `nonwhite`, and `site` variables to explicitly identify them as factor variables. That makes no difference in the estimated results, but we will take advantage of it in later examples. We also included `if e(sample)` to restrict the calculation to the estimation sample. In example 4 of [R] `mlogit`, the multinomial logit model was fit on 615 observations, so there must be missing values in our dataset.

Although we typed `outcome(1)`, specifying 1 for the indemnity outcome, we could have typed `outcome(Indemnity)`. For instance, to obtain the probabilities for prepaid, we could type

```stata
. predict p2 if e(sample), outcome(Prepaid)
    (option `pr' assumed; predicted probability)
    (29 missing values generated)
. summarize p2
```

```
Variable |       Obs        Mean    Std. dev.      Min      Max
----------|-------------|-----------|-------------|---------|---------|
p2        |       615 | .4504065 | .1125962 | .1964103 | .7885724 |
```

We must specify the label exactly as it appears in the underlying value label (or how it appears in the `mlogit` output), including capitalization.

Here we have used `predict` to obtain probabilities for the same sample on which we estimated. That is not necessary. We could use another dataset that had the independent variables defined (in our example, `age`, `male`, `nonwhite`, and `site`) and use `predict` to obtain predicted probabilities; here, we would not specify `if e(sample)`.

```stata

```
Example 2: Obtaining index values

predict can also be used to obtain the index values—the $\sum x_i \hat{\beta}_i^{(k)}$—as well as the probabilities:

. predict idx1, outcome(Indemnity) xb  
  (1 missing value generated)

. summarize idx1

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>idx1</td>
<td>643</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The indemnity outcome was our base outcome—the outcome for which all the coefficients were set to 0—so the index is always 0. For the prepaid and uninsured outcomes, we type

. predict idx2, outcome(Prepaid) xb  
  (1 missing value generated)

. predict idx3, outcome(Uninsure) xb  
  (1 missing value generated)

. summarize idx2 idx3

<table>
<thead>
<tr>
<th>Variable</th>
<th>Obs</th>
<th>Mean</th>
<th>Std. dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>idx2</td>
<td>643</td>
<td>-.0566113</td>
<td>.4962973</td>
<td>-1.298198</td>
<td>1.700719</td>
</tr>
<tr>
<td>idx3</td>
<td>643</td>
<td>-1.980747</td>
<td>.6018139</td>
<td>-3.112741</td>
<td>-.8258458</td>
</tr>
</tbody>
</table>

We can obtain the standard error of the index by specifying the stdp option:

. predict se2, outcome(Prepaid) stdp  
  (1 missing value generated)

. list p2 idx2 se2 in 1/5

<table>
<thead>
<tr>
<th>p2</th>
<th>idx2</th>
<th>se2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>.3709022</td>
<td>-.4831167</td>
</tr>
<tr>
<td>2.</td>
<td>.4977667</td>
<td>.055111</td>
</tr>
<tr>
<td>3.</td>
<td>.4113073</td>
<td>-.1712106</td>
</tr>
<tr>
<td>4.</td>
<td>.5424927</td>
<td>.3788345</td>
</tr>
<tr>
<td>5.</td>
<td>-.0925817</td>
<td>.1452616</td>
</tr>
</tbody>
</table>

We obtained the probability, p2, in the previous example.
Finally, `predict` can calculate the standard error of the difference in the index values between two outcomes with the `stddp` option:

```stata
. predict se_2_3, outcome(Prepaid,Uninsure) stddp
(1 missing value generated)
. list idx2 idx3 se_2_3 in 1/5
```

<table>
<thead>
<tr>
<th>idx2</th>
<th>idx3</th>
<th>se_2_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.483</td>
<td>-3.073</td>
<td>0.547</td>
</tr>
<tr>
<td>0.055</td>
<td>-2.716</td>
<td>0.433</td>
</tr>
<tr>
<td>-1.712</td>
<td>-1.579</td>
<td>0.306</td>
</tr>
<tr>
<td>0.378</td>
<td>-1.462</td>
<td>0.449</td>
</tr>
<tr>
<td>-0.093</td>
<td>-2.814</td>
<td>0.402</td>
</tr>
</tbody>
</table>

In the first observation, the difference in the indexes is $-0.483 - (-3.073) = 2.59$. The standard error of that difference is 0.547.

### Example 3: Interpreting results using predictive margins

It is more difficult to interpret the results from `mlogit` than those from `clogit` or `logit` because there are multiple equations. For example, suppose that one of the independent variables in our model takes on the values 0 and 1, and we are attempting to understand the effect of this variable. Assume that the coefficient on this variable for the second outcome, $\beta^{(2)}$, is positive. We might then be tempted to reason that the probability of the second outcome is higher if the variable is 1 rather than 0. Most of the time, that will be true, but occasionally we will be surprised. The probability of some other outcome could increase even more (say, $\beta^{(3)} > \beta^{(2)}$), and thus the probability of outcome 2 would actually fall relative to that outcome. We can use `predict` to help interpret such results.

Continuing with our previously fit insurance-choice model, we wish to describe the model’s predictions by race. For this purpose, we can use the method of predictive margins (also known as recycled predictions), in which we vary characteristics of interest across the whole dataset and average the predictions. That is, we have data on both whites and nonwhites, and our individuals have other characteristics as well. We will first pretend that all the people in our data are white but hold their other characteristics constant. We then calculate the probabilities of each outcome. Next we will pretend that all the people in our data are nonwhite, still holding their other characteristics constant. Again, we calculate the probabilities of each outcome. The difference in those two sets of calculated probabilities, then, is the difference due to race, holding other characteristics constant.

```stata
. gen byte nonwhold=nonwhite // save real race
. replace nonwhite=0 // make everyone white
(126 real changes made)
. predict wpind, outcome(Indemnity) // predict probabilities
(option pr assumed; predicted probability)
(1 missing value generated)
. predict wpp, outcome(Prepaid)
(option pr assumed; predicted probability)
(1 missing value generated)
. predict wpnoi, outcome(Uninsure)
(option pr assumed; predicted probability)
(1 missing value generated)
. replace nonwhite=1 // make everyone nonwhite
(644 real changes made)
```
In example 1 of [R] mlogit, we presented a cross-tabulation of insurance type and race. Those values were unadjusted. The means reported above are the values adjusted for age, sex, and site. Combining the results gives

<table>
<thead>
<tr>
<th></th>
<th>Unadjusted</th>
<th></th>
<th>Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>white</td>
<td>nonwhite</td>
<td>white</td>
</tr>
<tr>
<td>Indemnity</td>
<td>0.51</td>
<td>0.36</td>
<td>0.51</td>
</tr>
<tr>
<td>Prepaid</td>
<td>0.42</td>
<td>0.57</td>
<td>0.41</td>
</tr>
<tr>
<td>Uninsured</td>
<td>0.07</td>
<td>0.07</td>
<td>0.08</td>
</tr>
</tbody>
</table>

We find, for instance, after adjusting for age, sex, and site, that although 57% of nonwhites in our data had prepaid plans, 63% of nonwhites chose prepaid plans.

Computing predictive margins by hand was instructive, but we can compute these values more easily using the margins command (see [R] margins). The two margins for the indemnity outcome can be estimated by typing

```
margins nonwhite, predict(outcome(Indemnity)) noesample
```

margins also estimates the standard errors and confidence intervals of the margins. By default, margins uses only the estimation sample. We added the noesample option so that margins would use the entire sample and produce results comparable with our earlier analysis.
We can use `marginsplot` to graph the results from `margins`:

```
. marginsplot
Variables that uniquely identify margins: nonwhite
```

![Predictive margins of nonwhite with 95% CIs](image)

The margins for the other two outcomes can be computed by typing

```
. margins nonwhite, predict(outcome(Prepaid)) noesample
(output omitted)
. margins nonwhite, predict(outcome(Uninsure)) noesample
(output omitted)
```

The margins for each outcome is computed when no outcome is specified. For example,

```
. margins nonwhite, noesample
(output omitted)
```

## Technical note

You can use `predict` to classify predicted values and compare them with the observed outcomes to interpret a multinomial logit model. This is a variation on the notions of sensitivity and specificity for logistic regression. Here we will classify indemnity and prepaid as definitely predicting indemnity, definitely predicting prepaid, and ambiguous.

```
. predict indem, outcome(Indemnity) index // obtain indexes
(1 missing value generated)
. predict prepaid, outcome(Prepaid) index // obtain indexes
(1 missing value generated)
. gen diff = prepaid-indem // obtain difference
(1 missing value generated)
. predict sediff, outcome(Indemnity,Prepaid) stddp // & its standard error
(1 missing value generated)
. gen type = 1 if diff/sediff < -1.96 // definitely indemnity
(504 missing values generated)
. replace type = 3 if diff/sediff > 1.96 // definitely prepaid
(100 real changes made)
```
We can see that the predictive power of this model is modest. There are many misclassifications in both directions, though there are more correctly classified observations than misclassified observations.

Also, the uninsured look overwhelmingly as though they might have come from the indemnity system rather than from the prepaid system.

Calculating marginal effects

Example 4

We have already noted that the coefficients from multinomial logit can be difficult to interpret because they are relative to the base outcome. Another way to evaluate the effect of covariates is to examine the marginal effect of changing their values on the probability of observing an outcome.

The margins command can be used for this too. We can estimate the marginal effect of each covariate on the probability of observing the first outcome—indemnity insurance—by typing

```
margins, dydx(*) predict(outcome(Indemnity))
```

Average marginal effects  Number of obs = 615
Model VCE: OIM
Expression: Pr(insure==Indemnity), predict(outcome(Indemnity))
dy/dx wrt: age 1.male 1.nonwhite 2.site 3.site

<table>
<thead>
<tr>
<th></th>
<th>Delta-method</th>
<th></th>
<th></th>
<th></th>
<th>[95% conf. interval]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dy/dx</td>
<td>std. err.</td>
<td>z</td>
<td>P&gt;</td>
<td>z</td>
</tr>
<tr>
<td>age</td>
<td>.0026655</td>
<td>.001399</td>
<td>1.91</td>
<td>0.057</td>
<td>-.0000765</td>
</tr>
<tr>
<td>1.male</td>
<td>-.1295734</td>
<td>.0450945</td>
<td>-2.87</td>
<td>0.004</td>
<td>-.2179571</td>
</tr>
<tr>
<td>1.nonwhite</td>
<td>-.2032404</td>
<td>.0482554</td>
<td>-4.21</td>
<td>0.000</td>
<td>-.2978192</td>
</tr>
<tr>
<td>site</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>.0070995</td>
<td>.0479993</td>
<td>0.15</td>
<td>0.882</td>
<td>-.0869775</td>
</tr>
<tr>
<td>3</td>
<td>.1216165</td>
<td>.0505833</td>
<td>2.40</td>
<td>0.016</td>
<td>.022475</td>
</tr>
</tbody>
</table>

Note: dy/dx for factor levels is the discrete change from the base level.

By default, margins estimates the average marginal effect over the estimation sample, and that is what we see above. Being male decreases the average probability of having indemnity insurance by 0.130. We also see, from the note at the bottom of the table, that the marginal effect was computed as a discrete change in the probability of being male rather than female. That is why we made male a factor variable when fitting the model.
The dydx(*) option requested that margins estimate the marginal effect for each regressor, dydx(age) would have produced estimates only for the effect of age. margins has many options for controlling how the marginal effect is computed, including the ability to average over subgroups or to compute estimates for specified values of the regressors; see [R] margins.

margins will compute the marginal effects on each outcome when no outcome is specified.

```
. margins, dydx(*)
(output omitted)
```

Testing hypotheses about coefficients

Example 5

test tests hypotheses about the coefficients just as after any estimation command; see [R] test. Note, however, test’s syntax for dealing with multiple-equation models. Because test bases its results on the estimated covariance matrix, we might prefer a likelihood-ratio test; see example 5 in [R] mlogit for an example of lrtest.

If we simply list variables after the test command, we are testing that the corresponding coefficients are zero across all equations:

```
. test 2.site 3.site
( 1) [Indemnity]2o.site = 0
( 2) [Prepaid]2.o.site = 0
( 3) [Uninsure]2.o.site = 0
( 4) [Indemnity]3o.site = 0
( 5) [Prepaid]3.o.site = 0
( 6) [Uninsure]3.o.site = 0
Constraint 1 dropped
Constraint 4 dropped
   chi2( 4) = 19.74
   Prob > chi2 = 0.0006
```

We can test that all the coefficients (except the constant) in an equation are zero by simply typing the outcome in square brackets:

```
. test [Uninsure]
( 1) [Uninsure]age = 0
( 2) [Uninsure]0b.male = 0
( 3) [Uninsure]1.male = 0
( 4) [Uninsure]0b.nonwhite = 0
( 5) [Uninsure]1.nonwhite = 0
( 6) [Uninsure]1b.site = 0
( 7) [Uninsure]2.o.site = 0
( 8) [Uninsure]3.o.site = 0
Constraint 2 dropped
Constraint 4 dropped
Constraint 6 dropped
   chi2( 5) =  9.31
   Prob > chi2 = 0.0973
```

We specify the outcome just as we do with predict; we can specify the label if the outcome variable is labeled, or we can specify the numeric value of the outcome. We would have obtained the same test as above if we had typed test [3] because 3 is the value of insure for the outcome uninsured.

We can combine the two syntaxes. To test that the coefficients on the site variables are 0 in the equation corresponding to the outcome prepaid, we can type

```
. test 2.o.site 3.o.site
```
We specified the outcome and then followed that with a colon and the variables we wanted to test.

We can also test that coefficients are equal across equations. To test that all coefficients except the constant are equal for the prepaid and uninsured outcomes, we can type

```
.test [Prepaid=Uninsure]
```

```
( 1) [Prepaid]age - [Uninsure]age = 0
( 2) [Prepaid]0b.male - [Uninsure]0b.male = 0
( 3) [Prepaid]1.male - [Uninsure]1.male = 0
( 4) [Prepaid]0b.nonwhite - [Uninsure]0b.nonwhite = 0
( 5) [Prepaid]1.nonwhite - [Uninsure]1.nonwhite = 0
( 6) [Prepaid]1b.site - [Uninsure]1b.site = 0
( 7) [Prepaid]2.site - [Uninsure]2.site = 0
( 8) [Prepaid]3.site - [Uninsure]3.site = 0
Constraint 2 dropped
Constraint 4 dropped
Constraint 6 dropped
    chi2( 5) = 13.80
    Prob > chi2 = 0.0169
```

To test that only the site variables are equal, we can type

```
.test [Prepaid=Uninsure]: 2.site 3.site
```

```
( 1) [Prepaid]2.site - [Uninsure]2.site = 0
( 2) [Prepaid]3.site - [Uninsure]3.site = 0
    chi2( 2) = 12.68
    Prob > chi2 = 0.0018
```

Finally, we can test any arbitrary constraint by simply entering the equation and specifying the coefficients as described in [U] 13.5 Accessing coefficients and standard errors. The following hypothesis is senseless but illustrates the point:

```
.test ([Prepaid]age+[Uninsure]2.site)/2 = 2-[Uninsure]1.nonwhite
```

```
( 1) .5*[Prepaid]age + [Uninsure]1.nonwhite + .5*[Uninsure]2.site = 2
    chi2( 1) = 22.45
    Prob > chi2 = 0.0000
```

See [R] test for more information about test. The information there about combining hypotheses across test commands (the accumulate option) also applies after mlogit.

Reference


Also see

[R] mlogit — Multinomial (polytomous) logistic regression

[U] 20 Estimation and postestimation commands