# eoprobit predict — predict after eoprobit and xteoprobit

## Description

In this entry, we show how to create new variables containing observation-by-observation predictions after fitting a model with `eoprobit` or `xteoprobit`.

## Syntax

You previously fit the model

```plaintext
eoprobit y x1 ... ...
```

The equation specified immediately after the `eoprobit` command is called the main equation. It is

\[
Pr(y_i = m) = Pr(c_{m-1} \leq x_i \beta + e_i \cdot y \leq c_m)
\]

Or perhaps you had panel data and you fit the model with `xteoprobit` by typing

```plaintext
xteoprobit y x1 ... ...
```

Then the main equation would be

\[
Pr(y_{ij} = m) = Pr(c_{m-1} \leq x_{ij} \beta + u_i \cdot y + v_{ij} \cdot y \leq c_m)
\]

In either case, note that the equation produces a probability for each outcome \( m, m = 1 \) to \( M \). predict calculates predictions for the probabilities in the main equation. The other equations in the model are called auxiliary equations or complications. Our discussion follows the cross-sectional case with a single error term, but it applies to the panel-data case when we collapse the random effects and observation-level error terms, \( e_{ij} \cdot y = u_i \cdot y + v_{ij} \cdot y \).

The syntax of predict is

```plaintext
predict [ type ] { stub* | newvarlist } [ if ] [ in ] [ , stdstatistics howcalculated ]
```

### stdStatistics

<table>
<thead>
<tr>
<th>stdstatistics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr</td>
<td>probability of each outcome; the default</td>
</tr>
<tr>
<td>outlevel(#)</td>
<td>calculate probability for ( m = # ) only</td>
</tr>
<tr>
<td>xb</td>
<td>linear prediction excluding all complications</td>
</tr>
</tbody>
</table>
**Description**

<table>
<thead>
<tr>
<th>howcalculated</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>not fixed; base values from data</td>
</tr>
<tr>
<td>fix(endogvars)</td>
<td>fix specified endogenous covariates</td>
</tr>
<tr>
<td>base(valspecs)</td>
<td>specify base values of any variables</td>
</tr>
<tr>
<td>target(valspecs)</td>
<td>more convenient way to specify fix() and base()</td>
</tr>
</tbody>
</table>

Note: The `fix()` and `base()` options affect results only in models with endogenous variables in the main equation. The `target()` option is sometimes a more convenient way to specify the `fix()` and `base()` options.

*endogvars* are names of one or more endogenous variables appearing in the main equation.

*valspecs* specify the values for variables at which predictions are to be evaluated. Each *valspec* is of the form

```
varname = #
varname = (exp)
varname = othervarname
```

For instance, `base(valspecs)` could be `base(w1=0)` or `base(w1=0 w2=1)`.

Notes:

(1) `predict` can also calculate treatment-effect statistics. See [ERM] `predict treatment`.
(2) `predict` can also make predictions for the other equations in addition to the main-equation predictions discussed here. See [ERM] `predict advanced`.

### Options for statistics

- **pr** calculates the predicted probability for each outcome. In each observation, the predictions are the probabilities conditioned on the covariates. Results depend on how complications are handled, which is determined by the *howcalculated* options.

- **outlevel(#)** specifies to calculate only the probability for outcome \( m = # \) rather than calculating \( M \) probabilities. If you do not specify this option, \( y \) records three outcomes. You type

  ```
  . predict p1 p2 p3
  ```

  to obtain the probabilities for each outcome. If you want only the probability of the third outcome, you can type

  ```
  . predict p3, outlevel(#3)
  ```

  If the third outcome corresponded to \( y==3 \), you could instead type

  ```
  . predict p3, outlevel(3)
  ```

  If the third outcome corresponded to \( y==57 \), you could instead type

  ```
  . predict p3, outlevel(57)
  ```

  Most users number the outcomes 1, 2, and 3. Some users number them 0, 1, and 2. You could even number them 3, 5, and 57. Stata does not care how they are numbered.

- **xb** specifies that the linear prediction be calculated ignoring all complications.
Options for how results are calculated

By default, predictions are calculated taking into account all complications. This is discussed in Remarks and examples of [ERM] eregress predict.

fix(varname ...) specifies a list of endogenous variables from the main equation to be treated as if they were exogenous. This was discussed in [ERM] Intro 3 and is discussed further in Remarks and examples of [ERM] eregress predict.

base(varname = ...) specifies a list of variables from any equation and values for them. If eoprobit and xteoprobit were fitting linear models, we would tell you those values will be used in calculating the expected value of $e_i.y$ (or $e_{ij}.y$ in the panel case). That thinking will not mislead you but is not formally correct in the case of eoprobit and xteoprobit. Linear or nonlinear, errors from other equations spill over into the main equation because of correlations between errors. The correlations were estimated when the model was fit. The amount of spillover depends on those correlations and the values of the errors. This issue was discussed in [ERM] Intro 3 and is further discussed in Remarks and examples of [ERM] eregress predict.

target(varname = ...) is sometimes a more convenient way to specify the fix() and base() options. You specify a list of variables from the main equation and values for them. Those values override the values of the variables calculating $\beta_0 + \beta_1 x_i + \cdots$. Use of target() is discussed in Remarks and examples of [ERM] eregress predict.

Remarks and examples

Remarks are presented under the following headings:

Using predict after eoprobit and xteoprobit

How to think about nonlinear models

Using predict after eoprobit and xteoprobit

eoprobit and xteoprobit fit ordinal probit models. The outcome variable $y$ takes on various values such as 1, 2, 3, and 4, and each represents an ordered category, such as cannot walk, walks with difficulty, walks with few problems, and walks well. When you use predict after eoprobit or xteoprobit, remember to specify variables corresponding to each category.

.predict p1 p2 p3 p4

Alternatively, specify the outlevel(#) option.

With this exception, predictions after fitting models with eoprobit and xteoprobit are handled the same as they are after fitting models with eregress and xteregress. The issues are the same. See [ERM] eregress predict.

How to think about nonlinear models

What we wrote in [ERM] eprobit predict applies equally to the use of predict after eoprobit and xteoprobit. We wrote

Probit is a nonlinear model, and yet we just said that predictions after fitting models with eoprobit and xteoprobit are handled the same as they are after fitting models with eregress and xteregress. That statement is partly true, not misleading, but false in its details.
The regression-base discussion that we routed you to is framed in terms of expected values. In the nonlinear models, it needs to be framed in terms of distributional assumptions about the errors. For instance, predict after eprobit does not predict the expected value (mean) of $e_i.y$. It calculates the probability that $e_i.y$ exceeds $-x_i\beta$. These details matter hugely in implementation but can be glossed over for understanding the issues. For a full treatment of the issues, see Methods and formulas of [ERM] eprobit.

Methods and formulas

See Methods and formulas of [ERM] eoprobit postestimation.

Also see

[ERM] eoprobit postestimation — Postestimation tools for eoprobit and xteoprobit
[ERM] eoprobit — Extended ordered probit regression