**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

```
eoprobit y x1 ..., ...
```

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

$$\Pr(y_i = m) = \Pr(c_{m-1} \leq \mathbf{x}_i\boldsymbol{\beta} + e_i.\mathrm{y} \leq c_m)$$

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

```
xteoprobit y x1 ..., ...
```

Then the main equation would be

$$\Pr(y_{ij} = m) = \Pr(c_{m-1} \leq \mathbf{x}_{ij}\boldsymbol{\beta} + u_i.\mathrm{y} + v_{ij}.\mathrm{y} \leq c_m)$$

In either case, note that the equation produces a probability for each ordered category recorded in y. We will call the ordered categories outcomes. If there are $M$ outcomes, a probability is produced for each of $m$, $m = 1$ to $M$ outcomes. `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, $e_{ij}.\mathrm{y} = u_i.\mathrm{y} + v_{ij}.\mathrm{y}$.

All predictions after `xteoprobit` assume the panel-level random effects ($u_i.\mathrm{y}$) are zero. Put another way, predictions condition on random effects being set to their mean.

The syntax of `predict` is

`predict` [*type*] { *newvar* | *stub*`*` | *newvarlist* } [*if*] [*in*]
[ , *statistic asfmethod counterfactual* ]

| *statistic* | Description |
|---|---|
| Main | |
| `pr` | probability of each outcome; the default |
| `xb` | linear prediction excluding all complications |
| `outlevel(#)` | calculate probability for $m = \#$ only |

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

You specify one new variable with `xb`.

| *asfmethod* | Description |
|---|---|
| Main | |
| `asf` | average structural function; the default |
| `fixedasf` | fixed average structural function |
| `noasf` | no average structural function adjustment |

| *counterfactual* | Description |
|---|---|
| Main | |
| `target(`*valspecs*`)` | specify counterfactuals |

*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, `target(`*valspecs*`)` could be `target(w1=0)` or `target(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. It can also compute some rarely used statistics. See [ERM] **predict advanced**.

## Options for statistics

Main

`pr`, the default, calculates the predicted probability for all outcomes or for a specific outcome. To compute probabilities for all outcomes, you specify $M$ new variables, where $M$ is the number of categories of the dependent variable. Alternatively, you can specify *stub*`*`, in which case `pr` will store predicted probabilities in variables *stub*`1`, *stub*`2`, …, *stub*$M$. To compute the probability for a specific outcome, you specify one new variable and, optionally, the outcome value in option `outlevel()`; if you omit `outlevel()`, the first outcome value, `outlevel(#1)`, is assumed.

In each observation, the predictions are the probabilities conditioned on the covariates. Results depend on how complications are handled, which is determined by the *asfmethod* and *counterfactual* options.

`xb` specifies that the linear prediction be calculated ignoring all complications.

`outlevel(#)` specifies to calculate only the probability for outcome $m = \#$ rather than calculating $M$ probabilities, one for each ordered category (outcome) recorded in the dependent variable. If you do not specify this option and $y$ 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.

## Options for asfmethod

Main

`asf`, `fixedasf`, and `noasf` determine whether and how the average structural function (ASF) of the specified statistic is computed. These options are not allowed with `xb`.

`asf`, the default, calculates the ASF of the statistic. Thus, the default when no *statistic* is specified is the ASF of the probability of one or more ordered categories (outcomes) recorded in the dependent variable.

`asf` computes the statistic conditional on the errors of the endogenous variable equations. Put another way, it is the statistic accounting for the correlation of the endogenous covariates with the errors of the outcome equation. Derivatives and contrasts based on `asf` have a structural interpretation. See `margins` for computing derivatives and contrasts.

`fixedasf` calculates a fixed ASF. It is the specified statistic computed using only the coefficients and variables of the outcome equation. `fixedasf` does not condition on the errors of the endogenous variable equations. Contrasts between two fixed counterfactuals averaged over the whole sample have a potential-outcome interpretation. Intuitively, it is as if the values of the covariates were fixed at a value exogenously by fiat. See `margins` for computing derivatives and contrasts.

To be clear, derivatives and contrasts between two fixed counterfactuals using the default `asf` option also have a potential-outcome interpretation. And, unlike `fixedasf`, they retain that interpretation when computed over subpopulations for both linear and nonlinear models.

`noasf` calculates the statistic using the linear prediction with no adjustment. For extended regression models, this is computationally equivalent to `fixedasf`. So `fixedasf` and `noasf` are synonyms.

## Option for counterfactuals

Main

`target(`*valspecs*`)` specifies counterfactual predictions. You specify a list of variables from the main equation and values for them. Those values override the values of the variables calculating $\beta_0 + \beta_1\texttt{x1}_i + \cdots$. 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 `eprobit` and `xteprobit` 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-based 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 $-\mathbf{x}_i\boldsymbol{\beta}$. 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