

Description

Remarks and examples

Syntax

Methods and formulas

Options

Also see

## Description

`predict` has options to predict potential-outcome means, treatment effects, and treatment effects on the treated after models fit using the `entreat()` or `extreat()` option. The `predict` options are described below.

For standard use of `predict`, see

[ERM] [eregress predict](#)

[ERM] [eintreg predict](#)

[ERM] [eprobit predict](#)

[ERM] [eoprobit predict](#)

For advanced use of `predict`, see

[ERM] [predict advanced](#)

Also see [ERM] [estat teffects](#) for reports of average treatment statistics.

## Syntax

You previously fit a model by using the `entreat()` or `extreat()` option,

```
eregress y x1 ..., ... entreat(treated = ...) ...
eintreg y1 y2 x1 ..., ... entreat(treated = ...) ...
eprobit y x1 ..., ... entreat(treated = ...) ...
eoprobit y x1 ..., ... entreat(treated = ...) ...
xteregress y x1 ..., ... entreat(treated = ...) ...
xteintreg y1 y2 x1 ..., ... entreat(treated = ...) ...
xteprobit y x1 ..., ... entreat(treated = ...) ...
xteoprobit y x1 ..., ... entreat(treated = ...) ...

eregress y x1 ..., ... extreat(treated) ...
eintreg y1 y2 x1 ..., ... extreat(treated) ...
eprobit y x1 ..., ... extreat(treated) ...
eoprobit y x1 ..., ... extreat(treated) ...
xteregress y x1 ..., ... extreat(treated) ...
xteintreg y1 y2 x1 ..., ... extreat(treated) ...
xteprobit y x1 ..., ... extreat(treated) ...
xteoprobit y x1 ..., ... extreat(treated) ...
```

In these cases, `predict` has extra features. `predict`'s extra syntax for these features is

```
predict [type] newvar [if] [in], treatstatistic [treatmodifier oprobitmodifier]
```

In some cases, more than one new variable needs to be specified:

```
predict [type] { stub* | newvarlist } [if] [in], treatstatistic
      [treatmodifier oprobitmodifier]
```

<i>treatstatistic</i>	Description
<code>pomean</code>	potential-outcome mean (POM)
<code>te</code>	treatment effect (TE)
<code>tet</code>	treatment effect on the treated (TET)

<i>treatmodifier</i>	Description
<code>tlevel(#)</code>	treatment level for which <i>treatstatistic</i> is calculated

# may be specified as a value recorded in variable `treated`, such as 1, 2, ... or such as 1, 5, ..., depending on the values recorded.

# may also be specified as #1, #2, ..., meaning the first, second, ... values recorded in `treated`.

<i>oprobitmodifier</i>	Description
<code>outlevel(#)</code>	ordered outcome for which <i>treatstatistic</i> is calculated

When used after models fit with `eoprobit` or `xteoprobit`, *treatstatistic* is calculated for the specified outcome, or for the first outcome if you do not specify otherwise.

`outlevel(#)` specifies the outcome for which statistics are to be calculated. # is specified in the same way as with `tlevel()`, but the meaning is different. In the case of `outlevel()`, you are specifying the outcome, not the treatment level.

## Options

The options for the statistic to be calculated—`pomean`, `te`, and `tet`—are mutually exclusive. You calculate one treatment statistic per `predict` command.

`pomean` calculates the POMs for each treatment level. The POMs are the expected value of *y* that would have been observed if everyone was assigned to each of the treatment levels.

If there were two treatment levels (a control and a treatment), you would type

```
. predict pom1 pom2, pomean
```

If there were three levels, you would type

```
. predict pom1 pom2 pom3, pomean
```

`pomean` can alternatively be used with `tlevel()` to produce individual POMs:

```
. predict pom1, pomean tlevel(#1)
. predict pom2, pomean tlevel(#2)
```

If you have fit the model using `eoprobit` or `xteoprobit`, the POMs calculated for the examples above would be for *y*'s first outcome. You can change that. See [Predicting treatment effects after eoprobit and xteoprobit in Remarks and examples](#) below.

`te` calculates the TEs for each treatment level. The TEs are the differences in the POMs. For instance, if there were two treatment levels—a control and a treatment—there would be one treatment effect and it would be `pom2-pom1`. If there were three levels, there would be two treatment effects, `pom2-pom1` and `pom3-pom1`.

If there were two treatment levels—a control and a treatment—you would type

```
. predict te2, te
```

If there were three levels, you would type

```
. predict te2 te3, te
```

te can alternatively be used with `tlevel()` to produce individual TEs:

```
. predict te2, te tlevel(#2)
. predict te3, te tlevel(#3)
```

If you have fit the model using `eoprobit` or `xteoprobit`, the TEs calculated for the examples above would be for  $y$ 's first outcome. You can change that. See [Predicting treatment effects after eoprobit and xteoprobit in Remarks and examples](#) below.

tet calculates the TETs. The TETs are the differences in the POMs conditioned on treatment level.

If there were two treatment levels—a control and a treatment—you would type

```
. predict tet2, tet
```

If there were three levels, you would type

```
. predict tet2 tet3, tet
```

tet can alternatively be used with `tlevel()` to produce individual TETs:

```
. predict tet2, tet tlevel(#2)
. predict tet3, tet tlevel(#3)
```

If you have fit the model using `eoprobit` or `xteoprobit`, the TETs calculated for the examples above would be for  $y$ 's first outcome. You can change that. See [Predicting treatment effects after eoprobit and xteoprobit in Remarks and examples](#) below.

`tlevel(#)` is optionally used with `pomean`, `te`, or `tet`. Its use is illustrated above.

`outlevel(#)` is optionally used with `pomean`, `te`, or `tet` with models fit by `eoprobit` and `xteoprobit`.

See [Predicting treatment effects after eoprobit and xteoprobit in Remarks and examples](#) below.

## Remarks and examples

For an example of `predict` with treatment effects, see [ERM] [Intro 9](#).

Remarks are presented under the following headings:

*Predicting treatment effects after eregress, eintreg, xteregress, and xteintreg*  
*Predicting treatment effects after eprobit and xteprobit*  
*Predicting treatment effects after eoprobit and xteoprobit*

## Predicting treatment effects after eregress, eintreg, xteregress, and xteintreg

`eregress`, `eintreg`, `xteregress`, and `xteintreg` concern models with a continuous outcome variable. In `eregress` and `xteregress` models,  $y_i$  is observed. In `eintreg` and `xteintreg` models,  $y_i$  is not observed directly, but it is known that  $y_{l_i} \leq y_i \leq y_{u_i}$ .

Thus, the treatment statistics are expressed in the units of  $y$ . If  $y$  is blood pressure, the units are presumably mmHG. POMs are in mmHG. TEs and TETs are differences in blood pressure expressed in mmHG.

## Predicting treatment effects after eprobit and xteprobit

eprobit and xteprobit concern models with binary outcomes, and predictions are in terms of the probability of a positive outcome. Thus, POMs are probabilities. TEs and TETs are differences in probabilities.

## Predicting treatment effects after eoprobit and xteoprobit

eoprobit and xteoprobit concern models with ordinal outcome variables, and predictions are in terms of the probabilities—the probability of each outcome.

Treatment statistics are calculated on the basis of probabilities of outcomes. Thus, POMs are probabilities. TEs and TETs are differences in probabilities.

We want probabilities and differences in probabilities, but you need to specify which probability. The probability for the first outcome? The second?

If you do not specify which and simply type

```
. predict pom1 pom2 pom3, pomean
```

then the POMs are calculated for the first outcome, what eoprobit and xteoprobit call `outlevel(#1)`. If you wanted to obtain the POMs for `outlevel(#2)`, you would type

```
. predict pom1 pom2 pom3, pomean outlevel(#2)
```

If you wanted them for `outlevel(#3)`, you would type

```
. predict pom1 pom2 pom3, pomean outlevel(#3)
```

The same logic applies to calculating TE and TET with the `te` and `tet` options. `outlevel(#1)` is used unless you specify otherwise.

## Methods and formulas

See *Methods and formulas* in [\[ERM\] eintreg](#), [\[ERM\] eoprobit](#), [\[ERM\] eprobit](#), and [\[ERM\] eregress](#).

## Also see

[\[ERM\] eintreg postestimation](#) — Postestimation tools for `eintreg` and `xteintreg`

[\[ERM\] eintreg predict](#) — predict after `eintreg` and `xteintreg`

[\[ERM\] eoprobit postestimation](#) — Postestimation tools for `eoprobit` and `xteoprobit`

[\[ERM\] eoprobit predict](#) — predict after `eoprobit` and `xteoprobit`

[\[ERM\] eprobit postestimation](#) — Postestimation tools for `eprobit` and `xteprobit`

[\[ERM\] eprobit predict](#) — predict after `eprobit` and `xteprobit`

[\[ERM\] eregress postestimation](#) — Postestimation tools for `eregress` and `xteregress`

[\[ERM\] eregress predict](#) — predict after `eregress` and `xteregress`

Stata, Stata Press, and Mata are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow and NetCourseNow are trademarks of StataCorp LLC. Other brand and product names are registered trademarks or trademarks of their respective companies. Copyright © 1985–2025 StataCorp LLC, College Station, TX, USA. All rights reserved.

For suggested citations, see the FAQ on [citing Stata documentation](#).

