Title

stata.com

eregress predict — predict after eregress

Description

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

Syntax

You previously fit the model

```
eregress y x1 ..., ...
```

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

$$\mathtt{y}_i = \beta_0 + \beta_1 \mathtt{x1}_i + \cdots + e_i.\mathtt{y}$$

`predict` calculates predictions for `y` in the main equation. The other equations in the model are called auxiliary equations or complications.

The syntax of `predict` is

`predict` [*type*] *newvar* [*if*] [*in*] [, *stdstatistics howcalculated*]

stdstatistics	Description
`mean`	linear prediction; the default
`xb`	linear prediction excluding all complications

howcalculated	Description
default	not fixed; base values from data
`fix(`*endogvars*`)`	fix specified endogenous covariates
`base(`*valspecs*`)`	specify base values of any variables
`target(`*valspecs*`)`	more convenient way to specify `fix()` and `base()`

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

`mean` specifies that the linear prediction be calculated. In each observation, the linear prediction is the expected value of the dependent variable conditioned on the covariates. Results depend on how complications are handled, which is determined by the *howcalculated* options.

`xb` specifies that the linear prediction be calculated ignoring all complications. This prediction corresponds to what would be observed in data in which all the covariates in the main equation were exogenous.

Options for how results are calculated

By default, predictions are calculated taking into account all complications. This is discussed in *Remarks and examples*.

`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* below.

`base(`*varname* `= ...)` specifies a list of variables from any equation and values for them. Those values will be used in calculating the expected value of e_i`.y`. 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 discussed further in *Remarks and examples* below.

`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 $\beta_0 + \beta_1$`x1`$_i + \cdots$. Use of `target()` is discussed in *Remarks and examples* below.

Remarks and examples

stata.com

Remarks are presented under the following headings:

How to think about the model you fit

You have fit a model, perhaps by typing

```
. eregress y x1 x2                                              (1)
```

or

```
. eregress y x1 x2, endogenous(x1 = z1 z2, nomain)              (2)
```

or

```
. eregress y x1 x2 selected, endogenous(x1 = z1 z2, nomain)     (3)
> select(selected = x1 z3 z4)
```

The equation specified immediately after the `eregress` command is called the main equation. In the models above, it is

```
. eregress y x1 x2                                              (1)
. eregress y x1 x2                                              (2)
. eregress y x1 x2 selected                                     (3)
```

The equations specified in the options are called the auxiliary equations or complications. In the models above, they are

```
none                                                            (1)
. endogenous(x1 = z1 z2, nomain)                                (2)
. endogenous(x1 = z1 z2, nomain) select(selected = x1 z3 z4)    (3)
```

The auxiliary equations arose because of complications in the data you used to fit the model. The focus of ERMs is on fitting the main equation correctly in the presence of complications.

How to think about predictions

`predict` can make different kinds of predictions. The kind is specified by the how-to-calculate options.

Option	Result
none specified	calculate $\widehat{y}_i$ for data assuming they were generated just as the data used to fit the model were generated
`fix()`	calculate $\widehat{y}_i$ for data generated with the complication for the specified variable removed
`base()`	calculate $\widehat{y}_i$ just as in the *none specified* case, but calculate correlation-of-errors effects using the values for the covariates specified

The default calculation

When you use `predict` without options, you type

```
. predict yhat
```

`predict` calculates the expected values of $\mathtt{y}_i$ that would be observed given the complications present in your data.

Let's consider the three models we mentioned earlier.

```
. eregress y x1 x2                                              (1)
. eregress y x1 x2, endogenous(x1 = z1 z2, nomain)              (2)
. eregress y x1 x2 selected, endogenous(x1 = z1 z2, nomain)     (3)
> select(selected = x2 z3 z4)
```

The result from typing `predict yhat` without options will be

1. The expected values of $\mathtt{y}_i$ given `x1` and `x2`.
2. Same as (1) and taking into account that `x1` is endogenous and predicted by `z1` and `x1`.
3. Same as (2) and taking into account that `y` is observed only if the observation is `selected` and that `selected` is endogenous and given by `x2`, `z3`, and `z4`.

The other calculation options affect how the auxiliary equations are handled. Because model (1) has no auxiliary equations, the default prediction is the only one possible in its case.

`predict` without options can be used to calculate expected values with the data used in fitting the model and with other data that include the same complications. After fitting the model, you can type

```
. use anotherdataset
. predict yhat
```

You will sometimes use `predict` to calculate counterfactuals. If you do that, the default calculation can be used for changes in covariates that are exogenous in the main equation and appear in the main equation only.

Having fit any of the above models, you could type

```
. generate x2orig = x2
. replace x2 = 1000
. predict yhat
. replace x2 = x2orig
```

The predictions obtained would be the expected value of `y` given that each subject had `x2` set to 1,000.

A safer approach, however, is to specify the `base()` option. We will discuss `base()` in detail below, but the better solution is

```
. generate x2orig = x2
. replace x2 = 1000
. predict yhat, base(x2=x2orig)
. replace x2 = x2orig
```

If `base()` is unnecessary, it will cause no harm to specify it.

The fix() calculation

The purpose of the other calculation options is to make meaningful counterfactuals when you change the values of endogenous covariates. Option `fix(`*varname* ...`)` makes predictions as if the complications associated with *varname* were removed.

Assume you have fit model (3):

```
. eregress y x1 x2 selected, endogenous(x1 = z1 z2, nomain)  (3)
> select(selected = x2 z3 z4)
```

Then,

```
. predict yhat1, fix(x1)
```

would produce predictions that correspond to "what would have been observed" if the complication for `x1` had not been present either in the data or in the fitted model. These predicted values would correspond to a world in which the data-generating process was

```
. eregress y x1 x2 selected, select(selected = x2 z3 z4)     (3')
```

In this counterfactual world, `x1` is no longer endogenous. This switch from being endogenous to being exogenous is not a technicality. It is full of import. In the real world, $e.$`x1` is correlated with $e.$`y`. When we made the default prediction in the previous section, that correlation was taken into account. In this alternative world, there is no correlation. Perhaps `x1` records each subject's amount of health insurance coverage and `y` is a health outcome. In the world of the data used to fit the model, subjects chose to purchase health insurance, and presumably those who perceived a larger benefit would purchase more. Thus, the correlation between $e.$`x1` and $e.$`y` was positive. In the counterfactual world, perhaps purchase of health insurance is mandatory or it is free. Either way, the correlation between $e.$`x1` and $e.$`y` becomes 0.

Let's consider another prediction involving changing an endogenous variable.

```
. predict yhat2, fix(selected)
```

In this counterfactual world, selection is no longer endogenous. The predicted values would correspond to a world in which the data-generating process is

```
. eregress y x1 x2 selected, endogenous(x1 = z1 z2, nomain) (3'')
```

In this counterfactual world, `x1` is back to being endogenous, but selection no longer is. That breaks the correlation between $e.$`selected` and $e.$`y` in the same way the previous counterfactual broke the correlation between $e.$`x1` and $e.$`y`.

Another possible prediction is

```
. predict yhat2, fix(x1 selected)
```

The predicted values would correspond to a world in which the data-generating process is

```
. eregress y x1 x2 selected                                  (3''')
```

When you use `fix()`, you ordinarily change the values of the variable being fixed. You might type

```
. generate x1orig = x1
. replace x1 = 1                // $1 million
. predict yhat2, fix(x1)
. replace x1 = x1orig
```

or

```
. generate selectedorig = selected
. replace selected = 1           // or 0 as you please
. predict yhat2, fix(x1 selected)
. replace selected = selectedorig
```

or

```
. generate x1orig = x1
. generate selectedorig = selected
. replace x1 = 1                 // $1 million
. replace selected = 1           // or 0 as you please
. predict yhat2, fix(x1 selected)
. replace selected = selectedorig
. replace x1 = x1orig
```

The base() calculation

`fix()` is one way of handling predictions of counterfactuals when an endogenous variable in the main equation is changed. `base()` is the other.

Let's assume you have fit either model (2) or model (3):

```
. eregress y x1 x2, endogenous(x1 = z1 z2, nomain)            (2)
. eregress y x1 x2 selected, endogenous(x1 = z1 z2, nomain)   (3)
> select(selected = x2 z3 z4)
```

You cannot haphazardly change the value of an endogenous variable such as `x1` and expect to produce meaningful results. Because of that, you should *not* type

```
. generate x1orig = x1
. replace x1 = x1 + 1
. predict yhat
. replace x1 = x1orig
```

What would happen if you did? In either of the above models, there is an equation for `x1`. It is

```
endogenous(x1 = z1 z2, nomain)
```

which, written mathematically, is

$$\texttt{x1}_i = \gamma_0 + \gamma_1 \texttt{z1}_i + \gamma_2 \texttt{z2}_i + e_i.\texttt{x1}$$

You increased `x1` by 1 but did not change anything else. The equation above still holds, and so incrementing `x1` increased e_i.`x1` by 1 too.

What does it mean to increase e_i.`x1`? You are assuming that `x1` increased by 1 because the subjects decided to choose `x1+1` instead of `x1`. The only way that could happen is if they were different subjects.

Here is the thought experiment you just performed. You have data on subjects. What if you had different data on different subjects, each with the same characteristics as the current subjects, but who had chosen a value of `x1` that was one unit larger. Well, if these alternate subjects had chosen a value one unit larger than the current subjects, they would have done so for good reason, and their larger e.`x1` would have passed along its effect to the e.`y` because of the correlation. The new value of `y` would be the direct effect of `x1` in the `y` equation plus the change in e.`y`.

`predict yhat` without options produces the answer to the question that you never wanted to ask. What you wanted to ask was what would be the effect on `y` for the current subjects if endogenous variable `x1` was "exogenously" incremented by 1.

`predict, base()` will answer that question.

The subjects in your data are who they are because of their errors. Errors such as $e.$`x1` are the unobserved things about them that affect their choice of `x1`. You cannot change their errors without changing those unobserved things that make them who they are. If you want to ask about the effects of changes in `x1` holding the subjects constant, you need to ask about changes in `x1` holding $e_i.$`x1` constant.

`base()` does that and here is how you use it:

```
. generate x1orig = x1
. replace x1 = x1 + 1            // or whatever new values you please
. predict yhat3, base(x1=x1orig)
. replace x1 = x1orig
```

The option says that the unobserved components about the subjects in your data—the unobserved components that make them who they are—are to be calculated ignoring the values stored in `x1` (values that you have changed) and are instead to be calculated at the original values of `x1` (the values that will produce the same endogenously chosen solution). Then, we increase `x1` by 1.

The alternative target() option for making the fix() and base() calculations

`target()` is sometimes a more convenient way to make predictions using the `fix()` and `base()` calculations.

In the section above, one of the predictions was made by typing

```
. generate x1orig = x1
. replace x1 = x1 + 1            // or whatever new values you please
. predict yhat3, base(x1=x1orig)
. replace x1 = x1orig
```

We could have made the same prediction with `target()` by typing

```
. predict yhat3, target(x1=(x1+1))
```

Using `target()`, we specify the counterfactual calculation and leave variable `x1` unchanged. The unobserved components will be calculated on the basis of the values in variable `x1`.

In the section on `fix()`, one of the predictions was made by typing

```
. generate x1orig = x1
. replace x1 = 1                 // $1 million
. predict yhat2, fix(x1)
. replace x1 = x1orig
```

We could have made the same prediction with `target()` by typing

```
. predict yhat, fix(x1) target(x1=1)
```

You can use `target()` by itself as a substitute for `base()`, and you can use `target()` with `fix()`. In both cases, `target()` specifies the counterfactual, and you do not change the data in memory.

Methods and formulas

See *Methods and formulas* in [ERM] **eregress postestimation**.

Also see

[ERM] **eregress postestimation** — Postestimation tools for eregress

[ERM] **eregress** — Extended linear regression