

[Description](#)[Remarks and examples](#)[Also see](#)

Description

After you fit a model using one of the ERM commands, you can interpret the coefficients in the usual way. You can also use `margins` to produce counterfactuals, but you must specify one of the options `predict(base())` or `predict(fix())` on the `margins` command.

In this entry, we discuss how to interpret coefficients, how to use `margins`, and how to use `predict`.

Remarks and examples

stata.com

Remarks are presented under the following headings:

- [The problem of endogenous covariates](#)*
- [How to interpret coefficients](#)*
- [How to use margins](#)*
- [How to use margins in models without endogenous covariates](#)*
- [The two ways to use margins with endogenous covariates](#)*
- [Margins with predict\(base\(\)\)](#)*
- [Margins with predict\(fix\(\)\)](#)*
- [When to use which](#)*
- [How to use margins with predict\(base\(\)\)](#)*
- [How to use margins with predict\(fix\(\)\)](#)*
- [How to use predict](#)*

The problem of endogenous covariates

Endogenous covariates in the main equation cause problems, which means that if your model has no endogenous covariates in the main equation, you have no problems. The following models have no endogenous covariates in the main equation:

```

eregress y x1 x2
eregress y x1 x2 c.x1#c.x2
eregress y x1 x2, select(selected = x1 z1 z2) ///
                endogenous(z2 = z3 z4, nomain)

eprobit y x1 x2
eprobit y x1 x2 c.x1#c.x2
eprobit y x1 x2, select(selected = x1 z1 z2) ///
                endogenous(z2 = z3 z4, nomain)

```

We showed examples with `eregress` and `eprobit`. We could just as well have shown examples with `eintreg` and `eoprobit`. Note that the last model for each command we showed has an endogenous covariate, but it is *not* in the main equation.

In any case, if you have no endogenous covariates in the main equation, you interpret coefficients and use `margins` and `predict` just as you usually would.

In the rest of the manual entry, when we write about models with or without endogenous covariates, we mean models with or without endogenous covariates in the main equation.

Models with endogenous covariates in the main equation have issues of interpretation that arise even if you fit a model as simple as

```
eregress y x1, endogenous(x1 = z1, nomain)
```

There are four ways endogenous covariates can end up in the main equation:

1. You specify `endogenous(x1 = ...)` to add variable `x1` to the main equation.
2. You specify `endogenous(x1 = ..., nomain)` and you include `x1` in the main equation.
3. You specify `entreat(treated = ...)` to handle endogenous treatment effects. `entreat()` itself adds endogenous covariate `treated` to the main equation.
4. You specify `select(selected = ...)` to handle endogenous selection and you include `selected` in the main equation. `select()` makes variable `selected` endogenous, but it does not automatically add it to the main equation.

In what follows, we will show examples of endogenous covariates added to the main equation by option `endogenous()`, but we could have added them in any of the above ways.

In this manual entry, we depart from our usual practice of naming exogenous covariates `x1`, `x2`, ... and naming endogenous covariates `w1`, `w2`, ... We depart from this practice because we will introduce a situation and then say, “if `x1` is exogenous, do this; if it is endogenous, do something else”.

How to interpret coefficients

The rules for interpreting the coefficients from ERMs are as follows:

Rule C1 concerning models fit by `eregress` and `eintreg`.

Interpret the coefficients in the main equation in the usual way. As is usual, it does not matter whether coefficients are for endogenous or exogenous covariates.

Rule C2 concerning models fit by `eprobit` and `eoprobit` that have no endogenous covariates.

Interpret the coefficients in the main equation in the usual way. They are in standard deviation units just as they usually are. You can calculate probabilities in the usual way, too, by using the unit-normal distribution.

Rule C3 concerning models fit by `eprobit` and `eoprobit` that include endogenous covariates.

You can interpret the coefficients in the usual way—they are in standard deviation units—but you cannot calculate probabilities because the normal distribution used does not have standard deviation 1. Worse, the value of the standard deviation sometimes depends on the questions being asked. The safest approach is to use `margins`.

For rule C1, it does not matter whether you fit simple models such as

```
. eregress y x1 x2
```

or complicated ones such as

```
. eregress y x1 x2, endogenous(x1 = z1, nomain)
. eintreg y1 y2 x1 x2, endogenous(x1 = x2 z1, nomain)   ///
                      endogenous(z1 = x2 z2, nomain)   ///
                      select(selected = x2 z3 z4)
```

The coefficients on `x1` and `x2` in the main equation will have the usual interpretation. That usual interpretation is that they are the expected change in `y` for a one-unit change in `x1` or `x2`.

Rule C2 is not surprising at first glance. It states that `eprobit` and `eoprobit` models without endogenous covariates are ordinary probit and ordered probit models, and so their coefficients are interpreted in the ordinary way. However, the rule says more than that. It says that if you fit

```
. eprobit y x1 x2, select(selected = x2 z1)
```

or

```
. eoprobit y x1 x2, select(selected = x2 z1)
```

then you can interpret the fitted coefficients in the usual way too.

Rule C3 is a disappointment. It says that if you fit `eprobit` or `eoprobit` models with endogenous covariates in the main equation, such as

```
. eprobit y x1 x2, endogenous(x1 = z1, nomain)
```

then you must be careful in interpreting the coefficients. You can interpret them in almost the usual way because they are in standard-deviation units, but they are not in units for which the standard deviation is 1. In the above model, if the estimate of the coefficient on x_1 , $\hat{\beta}_1$, is 2 and x_1 has an average value of 1, then x_1 has a big effect. It is even bigger than usual because the standard deviation of the normal is less than 1. How much bigger? Well, that depends, which we admit is hardly a satisfactory answer.

If the coefficients on x_1 and x_2 , $\hat{\beta}_1$ and $\hat{\beta}_2$, are almost equal, and if x_1 and x_2 are on roughly the same scale, then the effects of x_1 and x_2 are roughly the same, and that conclusion does not depend on the standard deviation of the normal distribution being used.

More helpfully, rule C3 suggests that you use the `margins` command.

How to use margins

We warn you that two of the rules for using `margins`—rules M3 and M4—are a mouthful. Think of them as a reminder about the issues rather than an explanation of them. We will explain them. In any case, the rules are as follows:

Rule M1 concerning models with no endogenous covariates.

`margins` can be used just as you ordinarily would use it.

Rule M2 concerning models with endogenous covariates.

You must specify one of the `margins` options `predict(base())` or `predict(fix())`. Do that, and `margins` produces the results you expect. If you do not specify one of the options, results will be based on reduced-form predictions, which are not what you want.

Rule M3 concerning models with endogenous covariates.

Often, you will want to specify `predict(base())`. This produces levels and comparisons (averages and differences) conditional on observing all the covariates in your model. These include the covariates outside of the main equation. The averages and differences that `margins` reports will be the best predictions possible for subjects with the same characteristics as the subjects in your data.

Rule M4 concerning models with endogenous covariates.

Sometimes, you will need to specify option `predict(fix())`. This produces levels and comparisons (averages and differences) conditional on observing only the exogenous covariates in the main equation, and with the endogenous covariates set to the values specified. The averages and differences that `margins` reports will be the best predictions possible for subjects with the same limited set of characteristics as the subjects in your data.

How to use margins in models without endogenous covariates

Rule M1 was reassuring. It says that if your models include no endogenous covariates in the main equation, you can use margins in the ordinary way. Here is how you would ordinarily use `margins`. The following model has no endogenous covariates:

```
. use http://www.stata-press.com/data/r15/ermexample
(Artificial ERM example data)
. eregress y x1 x2 c.x1#c.x2
(output omitted)
```

The model fit is

$$y_i = \beta_0 + \beta_1 x1_i + \beta_2 x2_i + \beta_3 x1_i x2_i + e_i \cdot y$$

Assume that our interest is in the effect of `x1`. One way to interpret the effect is to interpret the coefficients: A one-unit increase in `x1` increases `y` by $\beta_1 + \beta_3 x2$. Another way to interpret the effect is by using counterfactuals. In these data, what would be the average change in `y` if `x1` were increased by 1? `margins` will tell us if we type

```
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r) nowald)
Contrasts of predictive margins
Model VCE      : OIM
Expression    : mean of y, predict()
1._at        : x1          = x1
2._at        : x1          = x1+1
```

	Delta-method			
	Contrast	Std. Err.	[95% Conf. Interval]	
_at (2 vs 1)	1.109641	.1750625	.7665246	1.452757

You can learn about `margins`, its features, and its syntax in [\[R\] margins](#). We will tell you enough, however, so that everything we say will make sense.

Assume that the data comprise three subgroups in which we have a special interest. For instance, we want to know how an increase in `x1` would affect each subgroup. `margins` can tell us that too.

```

. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r) nowald)
> over(group)

Contrasts of predictive margins
Model VCE      : OIM
Expression     : mean of y, predict()
over           : group
1._at         : 0.group
                x1             = x1
                1.group
                x1             = x1
                2.group
                x1             = x1
2._at         : 0.group
                x1             = x1+1
                1.group
                x1             = x1+1
                2.group
                x1             = x1+1

```

	Delta-method			
	Contrast	Std. Err.	[95% Conf. Interval]	
._at@group				
(2 vs 1) 0	.5561469	.1960937	.1718103	.9404835
(2 vs 1) 1	1.123401	.1754062	.7796108	1.46719
(2 vs 1) 2	1.641114	.2153742	1.218988	2.063239

`margins` helps us to understand changes that are different in each observation. If we had the simple model `eregress y x1 x2`, we know the effect of incrementing `x1` is to increase `y` by $\hat{\beta}_1$, which might be 3. The change would be 3 in every observation. In the model we have, however, the effect of incrementing `x1` is to increase `y` by $\beta_1 + \beta_3 x_2$. The average effect depends on the distribution of `x2`.

`margins` helps us to understand how a change affects the average in our data and subgroups of our data. We are using our sample as a proxy for the population and subpopulations, but that is what we usually do in statistics. We assume that our sample is representative. The issues are the same as we discussed in [\[ERM\] intro 5](#).

If our sample is representative but we want `margins` to report population-based standard errors, we need to specify `vce(robust)` when we fit the model:

```
. eregress y x1 x2 c.x1#c.x2, vce(robust)
```

If our sample is not representative, we can weight it with the inverse probability that its observations were sampled from the underlying population. If we want `margins` to report population-based standard errors, we can type

```
. eregress y x1 x2 c.x1#c.x2 [pw = weight], vce(robust)
```

or type

```
. eregress y x1 x2 c.x1#c.x2 [pw = weight]
```

We can type either because specifying `[pw=weight]` implies `vce(robust)`.

Even when we do specify or imply `vce(robust)`, `margins` will report sample standard errors by default. To obtain population-based standard errors, we must specify or imply `vce(robust)` when we fit the model and, when we use `margins`, we must specify its `vce(unconditional)` option:

```
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r)) ///
    vce(unconditional)
```

In the linear regression example we have been discussing, we included an interaction in the model and used `margins` to report averages. We used `margins` because the interaction caused changes to vary observation by observation. Probit and ordered probit models produce predictions that vary observation by observation even in models with no interactions. Consider the following probit model, which is almost the simplest one possible:

```
. eprobit y_p x1
```

The model is

$$\Pr(\text{positive outcome}) = \Pr(\beta_0 + \beta_1 x_{1i} + e_i \cdot y > 0) = \text{normal}(\beta_0 + \beta_1 x_1)$$

Assume that our interest is in `x1` just as it was previously. The effect of a one-unit increase in `x1` is to increase the normal index by $\hat{\beta}_1$. Simple, right? No, it is not. The effect in probabilities of that change varies observation by observation. Here is how the results vary if $\hat{\beta}_1$ were 0.5 and we incremented `x1` by 1. The effect depends on each subject's initial probability of a positive outcome:

Subject's original Pr(pos. outcome)	Increment by	Subject's new Pr(pos. outcome)	Difference
0.01	0.5 s.d.	0.03	0.02
0.10	0.5 s.d.	0.22	0.12
0.20	0.5 s.d.	0.37	0.17
0.40	0.5 s.d.	0.60	0.20
0.50	0.5 s.d.	0.69	0.19
0.60	0.5 s.d.	0.77	0.17
0.90	0.5 s.d.	0.96	0.06
0.99	0.5 s.d.	1.00	0.01

A subject whose original probability was 0.40 experiences an increase of 0.20 when `x1` is incremented by 1. Meanwhile, a subject whose probability was 0.90 experiences a mere 0.06 increase.

Using `margins`, we can obtain the average changes in probabilities in the data due to incrementing `x1` by 1. We type

```
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r) nowald)
Contrasts of adjusted predictions
Model VCE      : OIM
Expression     : Pr(y_p==1), predict()
1._at         : x1           = x1
2._at         : x1           = x1+1
```

	Delta-method		
	Contrast	Std. Err.	[95% Conf. Interval]
_at (2 vs 1)	.2961685	.0287644	.2397912 .3525458

We can obtain the changes for each of the three subgroups too:

```
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r) nowald)
> over(group)

Contrasts of adjusted predictions
Model VCE      : OIM
Expression    : Pr(y_p==1), predict()
over          : group
1._at        : 0.group
               x1           = x1
               1.group
               x1           = x1
               2.group
               x1           = x1
2._at        : 0.group
               x1           = x1+1
               1.group
               x1           = x1+1
               2.group
               x1           = x1+1
```

	Delta-method		
	Contrast	Std. Err.	[95% Conf. Interval]
._at@group			
(2 vs 1) 0	.3857775	.051078	.2856664 .4858885
(2 vs 1) 1	.2944176	.0294406	.2367152 .3521201
(2 vs 1) 2	.2096478	.0202614	.1699363 .2493594

Counterfactuals are useful in complicated linear models—we had an interaction in ours—and in nonlinear models whether simple or complicated.

The two ways to use margins with endogenous covariates

You may remember that rules M3 and M4 were mouthfuls. These rules said to use `margins` with the `predict(base())` option in one case and `predict(fix())` in another. Moreover, each option was “best”, albeit under different assumptions. We apologize for that. We can make the distinction clear in a reasonably simple model, namely

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

The model is

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + e_i.y$$

$$x_{1i} = \gamma_0 + \gamma_1 z_{1i} + e_i.x1$$

where $\rho = \text{corr}(e.x1, e.y)$ and is nonzero.

Let’s imagine that y is a health outcome and $x1$ is a 0/1 variable indicating whether a treatment was administered that is expected to improve the outcome. Observations are people, and people choose for themselves whether to have the treatment. Given the story, we *should* fit the model by typing

```
. eregress y i.x1 x2, endogenous(x1 = z1, probit nomain)
```

Nonetheless, we are going to fit the model without the `probit` specification and factor-variable notation for endogenous covariate $x1$:

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

We omit `probit` only because it will be easier for us to explain the difference between `predict(base())` and `predict(fix())`. We need to show you some math, and the math will be simpler in the linear model case.

What is important is that ρ is likely to be nonzero, no matter how the model is fit. ρ is the correlation between $e.y$ and $e.x1$. $e.y$ includes all the unobserved things that affect how well the treatment works. $e.x1$ includes all the unobserved things that affect whether individuals choose the treatment. ρ is likely to be nonzero and positive because people who believe that they are more likely to benefit from the treatment ($e.y > 0$) should be more likely to choose the treatment ($e.x1 > 0$).

As a result, the best prediction of y that we can make for people like person 1 in our data—people who have the same value of $x1$, $x2$, and $z1$ —includes the effect of $\hat{\rho}$, albeit indirectly. The best prediction of y we can make for people like person 1 is that their expected value of y will be

$$\hat{y}_1 = \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{21} + \hat{e}_{1.y}$$

$\hat{e}_{1.y}$ is our estimate of the expected value of $e.y$ in the first observation. Expected values of errors are often 0, but not in this case. This one depends on ρ . Given that we know the values x_{11} and z_{11} , we have an estimate of $e_{1.x1}$, namely

$$\hat{e}_{1.x1} = x_{11} - \hat{\gamma}_0 - \hat{\gamma}_1 z_{11}$$

Because $e.x1$ and $e.y$ are correlated, we can produce an estimate of $e_{1.y}$ given $\hat{e}_{1.x1}$ and $\hat{\rho}$. It is a detail, but the formula is

$$\hat{e}_{1.y} = \frac{\rho \times \text{s.d.}(e.y)}{\text{s.d.}(e.x1)} \times \hat{e}_{1.x1}$$

The value of $\hat{e}_{1.y}$ can be calculated, and the best prediction we can make for people like person 1 includes it, and is

$$\hat{y}_1 = \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{21} + \hat{e}_{1.y}$$

Margins with `predict(base())`

Here, we temporarily consider $x1$ to be continuous because we want to consider what happens if we add 1 to $x1$.

What is the best prediction we can make for people like person 1 if $x1$ were incremented by 1? It is

$$\hat{y}_1 = \hat{\beta}_0 + \hat{\beta}_1 (x_{11} + 1) + \hat{\beta}_2 x_{21} + \hat{e}_{1.y}$$

The above is how `margins` with option `predict(base())` makes the calculation for each observation in the data. Observation by observation, it calculates

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 (x_{1i} + 1) + \hat{\beta}_2 x_{2i} + \hat{e}_{i.y} \quad (1)$$

`predict(base())` tells `margins` to include $\hat{e}_{i.y}$ in the calculations. This is the best prediction for people like the people in our population conditioned on everything we know about them.

Now, we return to considering $x1$ to be binary.

Margins with predict(fix())

`predict(base())` uses (1) and makes its predictions given how the world currently operates. People choose their value of x_1 , and the choice they make is correlated with the outcomes they expect.

`predict(fix())` makes predictions for a world that operates differently. In the alternative world, x_1 is fixed at a value such as 1. This means that the population of people like person 1 is expanded from being all people like person 1 who made the same treatment choice to being all people like person 1 regardless of the treatment choice they made. In the expanded definition of people like person 1, the correlation between $e.y$ and $e.x_1$ is broken. The correlation is now 0, and the best prediction for people like person 1 sets $\hat{e}_{1.y}$ to 0:

$$\hat{y}_1 = \hat{\beta}_0 + \hat{\beta}_1 x_{11} + \hat{\beta}_2 x_{21} \quad (2)$$

In the jargon of statistics, x_1 is no longer endogenous—it is fixed, and the entire equation for x_1 becomes irrelevant.

When you specify `predict(fix())`, `margins()` makes the calculation for each person by using the approach used for person 1 in (2). It uses

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{2i} + \hat{\beta}_2 x_{2i}$$

These observation-by-observation predictions are called potential outcomes when applied to treatment models. The averages based on them that `margins` reports are called potential-outcome means (POMs). These averages correspond to what would be observed in a world in which x_1 is fixed at a particular value.

We considered x_1 being fixed at a constant value. x_1 can just as well be fixed at different values for different observations.

When to use which

`margins` can produce counterfactuals in two ways.

When you specify `predict(base())`, `margins` uses

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i} + \hat{e}_{i.y}$$

for the values of x_1 and x_2 specified. The predictions are a function of x_1 and x_2 and the covariates appearing in the x_1 equation. Those covariates along with $\hat{\rho}$ go into the calculation of $\hat{e}_{i.y}$. These predictions correspond to how the current world operates.

When you specify `predict(fix())`, `margins` uses

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i}$$

where x_1 is fixed at the value specified. These predictions are based on the exogenous covariates in the main equation (x_2 in this case) and the value to which the fixed variable (x_1) is set. These predictions correspond to a different world in which x_1 is no longer endogenous but is fixed to a particular value.

We have shown results for linear models. The formulas are more complicated when models are nonlinear but the assumptions and their implications are the same.

How to use margins with predict(base())

When we used margins with models in which there were no endogenous covariates, one of the comparisons we ran was

```
. eregress y x1 x2 c.x1#c.x2
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r))
```

The `at()` option ran two counterfactuals, although the first was more factual than counterfactual. Margins ran the factual `at(x1=x1)`. It ran the counterfactual `at(x1=generate(x1+1))`. Had we omitted `contrast(at(r))`, margins would have reported the means of `y` under the two scenarios. Because we specified `contrast(at(r))`, margins instead reported the average value of the difference.

To produce counterfactuals based on changing `x1`, you must specify option `predict(base())` in models containing any endogenous covariates in the main equation. You must include the option even if `x1` itself is not endogenous.

Let's imagine that we fit one of the models

```
. eregress y x1 x2 c.x1#c.x2, endogenous(x1 = z1, nomain)
. eregress y x1 x2 c.x1#c.x2, endogenous(x2 = z1, nomain)
. eregress y x1 x2 c.x1#c.x2, endogenous(x1 = z1, nomain) ///
                          endogenous(x2 = z2, nomain)
```

and now we want to produce the same counterfactual we produced when the model had no endogenous covariate, that is, when we typed

```
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r))
```

To produce the same counterfactual, we type

```
. generate x1orig = x1
. margins, at(x1=generate(x1)) at(x1=generate(x1+1)) contrast(at(r)) ///
  predict(base(x1=x1orig))
```

We did two things differently:

1. We created a new variable containing a copy of `x1`.
2. We added `predict(base(x1=x1orig))` to the margins command, which includes the copied variable.

If we wanted a comparison of `x1+1` with `x1+2`, we would type

```
. generate x1orig = x1
. margins, at(x1=generate(x1+1)) at(x1=generate(x1+2)) contrast(at(r)) ///
  predict(base(x1=x1orig))
```

If we requested counterfactuals that involved changing `x1` and `x2`, we would type

```
. generate x1orig = x1
. generate x2orig = x2
. margins, at(x1=generate(x1) x2=generate(x2)) ///
  at(x1=generate(x1+1) x2=generate(x2+1)) ///
  contrast(at(r)) predict(base(x1=x1orig x2=x2orig))
```

That is,

1. You must copy all variables that will be temporarily changed by margins.
2. You must specify the name of each original variable and the name of each copied variable in the `predict(base(original=copied))` option.

The variables that `margins` changes appear in its `at()` option. They can also appear in `varlist` following the `margins` command, such as

```
. eregress y x1 i.x2, endogenous(x1 = z1)
. generate x2 = x2orig
. margins r.x2, predict(base(x2=x2orig))
. drop x2orig
```

`margins r.x2` compares average values of `y` for each level `x2`. It reports them as differences in average values from the first level.

For examples using `margins` with `predict(base())`, see *Interpreting effects* in [ERM] [intro 8](#) and see [ERM] [example 1a](#).

How to use margins with predict(fix())

You have fit the model

```
. eregress y i.x1 x2, endogenous(x1 = z1, probit nomain)
```

This is the same example we discussed in *The two ways to use margins with endogenous covariates* except that now we specify the equation for `x1` as a probit equation.

To run the counterfactuals that `x1` is fixed at 0 and fixed at 1, type

```
. margins, predict(fix(x1)) at(x1=0 x1=1)
```

Averages of `y` will be reported based on predictions of `y` given the values of the exogenous covariates in the main equation (`x2` in this case) holding the fixed variable (`x1`) fixed first at 0 and then at 1.

If the model had two endogenous covariates in the main equation,

```
. eregress y x1 x2 x3, endogenous(x1 = z1, probit nomain) ///
                        endogenous(x2 = z2 z3, probit nomain) ///
                        endogenous(z3 = z4, nomain)
```

you could fix both of them by typing

```
. margins, predict(fix(x1 x2)) at(x1=1 x2=0)
```

The average of `y` will be reported given all the values of the exogenous covariates in the main equation (`x3` in this case) holding `x1` and `x2` fixed at the values specified.

You could fix `x1` only:

```
. margins, predict(fix(x1)) at(x1=1)
```

The average of `y` will be reported given

- the values of all the exogenous covariates in the main equation (`x3` in this case), plus
- the values of `x2` and of all the covariates in its equation, whether endogenous or exogenous (`x2`, `z2`, and `z3` in this case), plus
- all the covariates necessary to predict `x2`'s endogenous covariates (`z4` in this case).

Had `z4` been endogenous and had an equation, we would have added that equation's variables, and so on.

In this case, the average of `y` will be reported given `x3`, `x2`, `z2`, `z3`, and `z4`. `x1` will be fixed.

If the model had been

```
. eregress y x1 x2 x3, endogenous(x1 = z1, probit nomain)      ///  
                        endogenous(x2 = x1 z2, probit nomain)
```

then `margins` would refuse to fix just `x1`:

```
. margins, predict(fix(x1)) at(x1=1)  
endogenous x2 depends on fixed x1  
r(498);
```

We tried to fix `x1` and `x1` also affects `x2`. `margins, predict(fix())` cannot do this.

You could, however, fix `x2` because `x2` does not affect `x1`:

```
. margins, predict(fix(x2)) at(x2=1)
```

For examples using `margins` with `predict(fix())`, see [Interpreting effects](#) in [ERM] [intro 8](#) and see [ERM] [example 1a](#).

How to use predict

Regardless of how or why a model was fit, Stata's postestimation `predict` command is used in three ways:

In-sample prediction.

`predict` is used to obtain predicted values from the data used to fit the model.

Out-of-sample prediction.

`predict` is used to obtain predicted values from other data, data not used to fit the model.

Counterfactual prediction.

`predict` is used to obtain what the predicted values would be if the values of a covariate or covariates were changed. Counterfactual prediction can be performed with the data used to fit the model or other data.

The rules for using `predict` after ERM depend on the way `predict` is being used. The rules are the following:

Rule P1 for models with no endogenous covariates.

`predict` is used for in-sample, out-of-sample, and counterfactual prediction just as you would ordinarily use it. This rule applies to all models with no endogenous covariates in the main equation.

Rule P2 for models with endogenous covariates.

`predict` is used for in-sample and out-of-sample prediction just as you would ordinarily use it.

Rule P3 for models with endogenous covariates.

You must specify option `base()` or `fix()` when using `predict` for counterfactual prediction.

Here is how `predict` is ordinarily used. You have fit the model

```
. eregress y x1 x2
```

The model you fit could just as well be fit by `eintreg`, `eoprobit`, or `eoprobbit`.

To make in-sample predictions, you type (with the same dataset in memory)

```
. predict yhat
```

New variable `yhat` will contain the predicted values based on the fitted model.

To make out-of-sample predictions, you type

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

New variable `yhat` will contain predicted values based on the fitted model.

You could also use one part of the original dataset to fit the model and make predictions simultaneously both in and outside of the data used to fit the model:

```
. eregress y x1 x2 if subset==1
. predict yhat
```

New variable `yhat` would contain in-sample predictions in observations for which `subset==1` and would contain out-of-sample predictions in the other observations.

You can make counterfactual predictions. You have fit the model

```
. eregress y x1 x2
```

To obtain predicted values of `y` if `x1` were 1 in all observations, you type

```
. eregress y x1 x2
. replace x1 = 1
. predict yhat1
```

New variable `yhat1` will contain predicted values conditional on `x1` being 1.

You use `predict` in models with endogenous covariates in the main equation just as we have shown when making in-sample or out-of-sample predictions. To make counterfactual predictions in models with endogenous covariates in the main equation, such as

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

you type

```
. generate x1orig = x1
. replace x1 = 1
. predict yhat1, base(x1=x1orig)
. replace x1 = x1orig
. drop x1orig
```

or you type

```
. generate x1orig = x1
. replace x1 = 1
. predict yhat1, fix(x1)
. replace x1 = x1orig
. drop x1orig
```

You must specify option `base()` or `fix()` with `predict` for the same reasons you must specify option `predict(base())` or `predict(fix())` with `margins`. You make the decision about which to specify in the same way you make the decision with `margins`.

Note that `predict` used for making counterfactual predictions works just like `predict` used for making in-sample or out-of-sample predictions in one respect. `predict` uses the values of the variables in memory. To make counterfactual predictions, you must change the contents of those variables.

Using `predict` to predict counterfactuals reproduces results produced by `margins`. Typing

```
. generate x1orig = x1
. margins, predict(base(x1=x1orig)) at(x1=generate(x1+1))
```

is equivalent to typing

```
. generate x1orig = x1
. replace x1 = 1
. predict yhat1, base(x1=x1orig)
. summarize yhat1
```

Both will report the same result for the average of y if x_1 were incremented by 1.

Typing

```
. margins, predict(fix(x1)) at(x1=1)
```

is equivalent to typing

```
. generate x1orig = x1
. replace x1 = 1
. predict yhat1, fix(x1)
. summarize yhat1
```

For your information, `margins` uses `predict` in making its calculations, and that explains why the options on `margins` are named `predict(base())` and `predict(fix())`. When `margins` uses `predict`, it specifies to predict the options you specified in option `predict()`.

Also see

[ERM] [intro 8](#) — Conceptual introduction via worked example

[ERM] [example 1a](#) — Linear regression with continuous endogenous covariate