Title

predict — Obtain predictions, residuals, etc., after estimation

Syntax	Menu for predict	Description	Options
Remarks and examples	Methods and formulas	Also see	

Syntax

After single-equation (SE) *models*

predict [*type*] *newvar* [*if*] [*in*] [, *single_options*]

After multiple-equation (ME) *models*

predict [*type*] *newvar* [*if*] [*in*] [, *multiple_options*]

predict [*type*] { *stub** | $newvar_1$ ... $newvar_q$ } [*if*] [*in*] , scores

single_options	Description
Main	
xb	calculate linear prediction
stdp	calculate standard error of the prediction
score	calculate first derivative of the log likelihood with respect to $\mathbf{x}_j\boldsymbol{\beta}$
Options	
nooffset	ignore any offset() or exposure() variable
other_options	command-specific options

multiple_options	Description
Main	
equation(*eqno*[, *eqno*])	specify equations
xb	calculate linear prediction
stdp	calculate standard error of the prediction
stddp	calculate the difference in linear predictions
Options	
nooffset	ignore any offset() or exposure() variable
other_options	command-specific options

Menu for predict

Statistics > Postestimation > Predictions, residuals, etc.

Description

`predict` calculates predictions, residuals, influence statistics, and the like after estimation. Exactly what `predict` can do is determined by the previous estimation command; command-specific options are documented with each estimation command. Regardless of command-specific options, the actions of `predict` share certain similarities across estimation commands:

1. `predict` *newvar* creates *newvar* containing "predicted values"—numbers related to the $E(y_j|\mathbf{x}_j)$. For instance, after linear regression, `predict` *newvar* creates $\mathbf{x}_j\mathbf{b}$ and, after probit, creates the probability $\Phi(\mathbf{x}_j\mathbf{b})$.
2. `predict` *newvar*`, xb` creates *newvar* containing $\mathbf{x}_j\mathbf{b}$. This may be the same result as option 1 (for example, linear regression) or different (for example, probit), but regardless, option `xb` is allowed.
3. `predict` *newvar*`, stdp` creates *newvar* containing the standard error of the linear prediction $\mathbf{x}_j\mathbf{b}$.
4. `predict` *newvar*`,` *other_options* may create *newvar* containing other useful quantities; see `help` or the reference manual entry for the particular estimation command to find out about other available options.
5. `nooffset` added to any of the above commands requests that the calculation ignore any offset or exposure variable specified by including the `offset(`*varname*$_o$`)` or `exposure(`*varname*$_e$`)` option when you fit the model.

`predict` can be used to make in-sample or out-of-sample predictions:

6. `predict` calculates the requested statistic for all possible observations, whether they were used in fitting the model or not. `predict` does this for standard options 1–3 and generally does this for estimator-specific options 4.
7. `predict` *newvar* `if e(sample), ...` restricts the prediction to the estimation subsample.
8. Some statistics make sense only with respect to the estimation subsample. In such cases, the calculation is automatically restricted to the estimation subsample, and the documentation for the specific option states this. Even so, you can still specify `if e(sample)` if you are uncertain.
9. `predict` can make out-of-sample predictions even using other datasets. In particular, you can

```
. use ds1
.  (fit a model)
. use two                       /* another dataset         */
. predict yhat, ...             /* fill in the predictions */
```

Options

Main

`xb` calculates the linear prediction from the fitted model. That is, all models can be thought of as estimating a set of parameters b_1, b_2, ..., b_k, and the linear prediction is $\widehat{y}_j = b_1x_{1j} + b_2x_{2j} + \cdots + b_kx_{kj}$, often written in matrix notation as $\widehat{\mathbf{y}}_j = \mathbf{x}_j\mathbf{b}$. For linear regression, the values $\widehat{y}_j$ are called the predicted values or, for out-of-sample predictions, the forecast. For logit and probit, for example, $\widehat{y}_j$ is called the logit or probit index.

x_{1j}, x_{2j}, ..., x_{kj} are obtained from the data currently in memory and do not necessarily correspond to the data on the independent variables used to fit the model (obtaining b_1, b_2, ..., b_k).

`stdp` calculates the standard error of the linear prediction. Here the prediction means the same thing as the "index", namely, $\mathbf{x}_j\mathbf{b}$. The statistic produced by `stdp` can be thought of as the standard error of the predicted expected value, or mean index, for the observation's covariate pattern. The standard error of the prediction is also commonly referred to as the standard error of the fitted value. The calculation can be made in or out of sample.

`stddp` is allowed only after you have previously fit a multiple-equation model. The standard error of the difference in linear predictions $(\mathbf{x}_{1j}\mathbf{b} - \mathbf{x}_{2j}\mathbf{b})$ between equations 1 and 2 is calculated. This option requires that `equation(`*eqno*$_1$`,`*eqno*$_2$`)` be specified.

`score` calculates the equation-level score, $\partial \ln L/\partial(\mathbf{x}_j\boldsymbol{\beta})$. Here $\ln L$ refers to the log-likelihood function.

`scores` is the ME model equivalent of the `score` option, resulting in multiple equation-level score variables. An equation-level score variable is created for each equation in the model; ancillary parameters—such as $\ln\sigma$ and atanhρ—make up separate equations.

`equation(`*eqno*[`,`*eqno*]`)`—synonym `outcome()`—is relevant only when you have previously fit a multiple-equation model. It specifies the equation to which you are referring.

`equation()` is typically filled in with one *eqno*—it would be filled in that way with options `xb` and `stdp`, for instance. `equation(#1)` would mean the calculation is to be made for the first equation, `equation(#2)` would mean the second, and so on. You could also refer to the equations by their names. `equation(income)` would refer to the equation named income and `equation(hours)` to the equation named hours.

If you do not specify `equation()`, results are the same as if you specified `equation(#1)`.

Other statistics, such as `stddp`, refer to between-equation concepts. In those cases, you might specify `equation(#1,#2)` or `equation(income,hours)`. When two equations must be specified, `equation()` is required.

Options

`nooffset` may be combined with most statistics and specifies that the calculation should be made, ignoring any offset or exposure variable specified when the model was fit.

This option is available, even if it is not documented for `predict` after a specific command. If neither the `offset(`*varname*$_o$`)` option nor the `exposure(`*varname*$_e$`)` option was specified when the model was fit, specifying `nooffset` does nothing.

other_options refers to command-specific options that are documented with each command.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Most of the examples are presented using linear regression, but the general syntax is applicable to all estimators.

You can think of any estimation command as estimating a set of coefficients b_1, b_2, …, b_k corresponding to the variables x_1, x_2, …, x_k, along with a (possibly empty) set of ancillary statistics γ_1, γ_2, …, γ_m. All estimation commands store the b_is and γ_is. `predict` accesses that stored information and combines it with the data currently in memory to make various calculations. For instance, `predict` can calculate the linear prediction, $\widehat{y}_j = b_1 x_{1j} + b_2 x_{2j} + \cdots + b_k x_{kj}$. The data on which `predict` makes the calculation can be the same data used to fit the model or a different dataset—it does not matter. `predict` uses the stored parameter estimates from the model, obtains the corresponding values of x for each observation in the data, and then combines them to produce the desired result.

Estimation-sample predictions

▷ Example 1

We have a 74-observation dataset on automobiles, including the mileage rating (`mpg`), the car's weight (`weight`), and whether the car is foreign (`foreign`). We fit the model

```
. use http://www.stata-press.com/data/r13/auto
(1978 Automobile Data)

. regress mpg weight if foreign
      Source │       SS           df       MS              Number of obs =       22
─────────────┼──────────────────────────────────           F(  1,    20) =    17.47
       Model │  427.990298         1  427.990298           Prob > F      =   0.0005
    Residual │  489.873338        20  24.4936669           R-squared     =   0.4663
─────────────┼──────────────────────────────────           Adj R-squared =   0.4396
       Total │  917.863636        21  43.7077922           Root MSE      =   4.9491

─────────────┬────────────────────────────────────────────────────────────────
         mpg │      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
─────────────┼────────────────────────────────────────────────────────────────
      weight │   -.010426   .0024942    -4.18   0.000    -.0156287   -.0052232
       _cons │    48.9183   5.871851     8.33   0.000     36.66983    61.16676
─────────────┴────────────────────────────────────────────────────────────────
```

If we were to type `predict pmpg` now, we would obtain the linear predictions for all 74 observations. To obtain the predictions just for the sample on which we fit the model, we could type

```
. predict pmpg if e(sample)
(option xb assumed; fitted values)
(52 missing values generated)
```

Here `e(sample)` is true only for foreign cars because we typed `if foreign` when we fit the model and because there are no missing values among the relevant variables. If there had been missing values, `e(sample)` would also account for those.

By the way, the `if e(sample)` restriction can be used with any Stata command, so we could obtain summary statistics on the estimation sample by typing

```
. summarize if e(sample)
  (output omitted)
```

◁

Out-of-sample predictions

By out-of-sample predictions, we mean predictions extending beyond the estimation sample. In the example above, typing `predict pmpg` would generate linear predictions using all 74 observations.

`predict` will work on other datasets, too. You can use a new dataset and type `predict` to obtain results for that sample.

▷ Example 2

Using the same auto dataset, assume that we wish to fit the model

$$\texttt{mpg} = \beta_1\texttt{weight} + \beta_2\ln(\texttt{weight}) + \beta_3\texttt{foreign} + \beta_4$$

We first create the ln(`weight`) variable, and then type the `regress` command:

```
. use http://www.stata-press.com/data/r13/auto, clear
(1978 Automobile Data)

. generate lnweight = ln(weight)

. regress mpg weight lnweight foreign

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  3,    70) =   52.36
       Model |  1690.27997     3  563.426657           Prob > F      =  0.0000
    Residual |  753.179489    70   10.759707           R-squared     =  0.6918
-------------+------------------------------           Adj R-squared =  0.6785
       Total |  2443.45946    73  33.4720474           Root MSE      =  3.2802

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |    .003304   .0038995     0.85   0.400    -.0044734    .0110813
    lnweight |  -29.59133   11.52018    -2.57   0.012     -52.5676   -6.615061
     foreign |  -2.125299   1.052324    -2.02   0.047    -4.224093   -.0265044
       _cons |   248.0548   80.37079     3.09   0.003     87.76035    408.3493
------------------------------------------------------------------------------
```

If we typed `predict pmpg` now, we would obtain predictions for all 74 cars in the current data. Instead, we are going to use a new dataset.

The dataset `newautos.dta` contains the make, weight, and place of manufacture of two cars, the Pontiac Sunbird and the Volvo 260. Let's use the dataset and create the predictions:

```
. use http://www.stata-press.com/data/r13/newautos, clear
(New Automobile Models)

. list

     +-----------------------------------+
     |         make   weight    foreign  |
     |-----------------------------------|
  1. | Pont. Sunbird     2690   Domestic  |
  2. |     Volvo 260     3170    Foreign  |
     +-----------------------------------+

. predict mpg
(option xb assumed; fitted values)
variable lnweight not found
r(111);
```

Things did not work. We typed `predict mpg`, and Stata responded with the message "variable lnweight not found". `predict` can calculate predicted values on a different dataset only if that dataset contains the variables that went into the model. Here our dataset does not contain a variable called `lnweight`. `lnweight` is just the log of weight, so we can create it and try again:

```
. generate lnweight = ln(weight)
. predict mpg
(option xb assumed; fitted values)
. list

     +----------------------------------------------------------+
     |          make   weight    foreign   lnweight         mpg |
     |----------------------------------------------------------|
  1. | Pont. Sunbird     2690   Domestic   7.897296    23.25097 |
  2. |     Volvo 260     3170    Foreign   8.061487    17.85295 |
     +----------------------------------------------------------+
```

We obtained our predicted values. The Pontiac Sunbird has a predicted mileage rating of 23.3 mpg, whereas the Volvo 260 has a predicted rating of 17.9 mpg.

◁

Residuals

▷ Example 3

With many estimators, `predict` can calculate more than predicted values. With most regression-type estimators, we can, for instance, obtain residuals. Using our regression example, we return to our original data and obtain residuals by typing

```
. use http://www.stata-press.com/data/r13/auto, clear
(1978 Automobile Data)
. generate lnweight = ln(weight)
. regress mpg weight lnweight foreign
  (output omitted)
. predict double resid, residuals
. summarize resid
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+----------------------------------------------------------
       resid |         74   -1.51e-15    3.212091  -5.453078   13.83719
```

We could do this without refitting the model. Stata always remembers the last set of estimates, even as we use new datasets.

It was not necessary to type the `double` in `predict double resid, residuals`, but we wanted to remind you that you can specify the type of a variable in front of the variable's name; see [U] **11.4.2 Lists of new variables**. We made the new variable `resid` a `double` rather than the default `float`.

If you want your residuals to have a mean as close to zero as possible, remember to request the extra precision of `double`. If we had not specified `double`, the mean of `resid` would have been roughly 10^{-9} rather than 10^{-14}. Although 10^{-14} sounds more precise than 10^{-9}, the difference really does not matter.

◁

For linear regression, `predict` can also calculate standardized residuals and Studentized residuals with the options `rstandard` and `rstudent`; for examples, see [R] **regress postestimation**.

Single-equation (SE) models

If you have not read the discussion above on using `predict` after linear regression, please do so. And `predict`'s default calculation almost always produces a statistic in the same metric as the dependent variable of the fitted model—for example, predicted counts for Poisson regression. In any case, `xb` can always be specified to obtain the linear prediction.

`predict` can calculate the standard error of the prediction, which is obtained by using the covariance matrix of the estimators.

▷ Example 4

After most binary outcome models (for example, `logistic`, `logit`, `probit`, `cloglog`, `scobit`), `predict` calculates the probability of a positive outcome if we do not tell it otherwise. We can specify the `xb` option if we want the linear prediction (also known as the logit or probit index). The odd abbreviation `xb` is meant to suggest $\mathbf{x}\boldsymbol{\beta}$. In logit and probit models, for example, the predicted probability is $p = F(\mathbf{x}\boldsymbol{\beta})$, where $F()$ is the logistic or normal cumulative distribution function, respectively.

```
. logistic foreign mpg weight
  (output omitted )
. predict phat
(option pr assumed; Pr(foreign))
. predict idxhat, xb
. summarize foreign phat idxhat
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
     foreign |         74    .2972973    .4601885          0          1
        phat |         74    .2972973    .3052979    .000729   .8980594
      idxhat |         74   -1.678202    2.321509  -7.223107   2.175845
```

Because this is a logit model, we could obtain the predicted probabilities ourselves from the predicted index

```
. generate phat2 = exp(idxhat)/(1+exp(idxhat))
```

but using `predict` without options is easier.

◁

▷ Example 5

For all models, `predict` attempts to produce a predicted value in the same metric as the dependent variable of the model. We have seen that for dichotomous outcome models, the default statistic produced by `predict` is the probability of a success. Similarly, for Poisson regression, the default statistic produced by `predict` is the predicted count for the dependent variable. You can always specify the `xb` option to obtain the linear combination of the coefficients with an observation's x values (the inner product of the coefficients and x values). For `poisson` (without an explicit exposure), this is the natural log of the count.

```
. use http://www.stata-press.com/data/r13/airline, clear
. poisson injuries XYZowned
  (output omitted )
```

```
. predict injhat
(option n assumed; predicted number of events)
. predict idx, xb
. generate exp_idx = exp(idx)
. summarize injuries injhat exp_idx idx
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    injuries |          9    7.111111    5.487359          1         19
      injhat |          9    7.111111    .8333333          6   7.666667
     exp_idx |          9    7.111111    .8333333          6   7.666667
         idx |          9    1.955174    .1225612   1.791759   2.036882
```

We note that our "hand-computed" prediction of the count (`exp_idx`) matches what was produced by the default operation of `predict`.

If our model has an exposure-time variable, we can use `predict` to obtain the linear prediction with or without the exposure. Let's verify what we are getting by obtaining the linear prediction with and without exposure, transforming these predictions to count predictions and comparing them with the default count prediction from `predict`. We must remember to multiply by the exposure time when using `predict ..., nooffset`.

```
. use http://www.stata-press.com/data/r13/airline, clear
. poisson injuries XYZowned, exposure(n)
  (output omitted)
. predict double injhat
(option n assumed; predicted number of events)
. predict double idx, xb
. gen double exp_idx = exp(idx)
. predict double idxn, xb nooffset
. gen double exp_idxn = exp(idxn)*n
. summarize injuries injhat exp_idx exp_idxn idx idxn
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    injuries |          9    7.111111    5.487359          1         19
      injhat |          9    7.111111     3.10936   2.919621   12.06158
     exp_idx |          9    7.111111     3.10936   2.919621   12.06158
    exp_idxn |          9    7.111111     3.10936   2.919621   12.06158
         idx |          9    1.869722    .4671044   1.071454   2.490025
-------------+---------------------------------------------------------
        idxn |          9     4.18814    .1904042   4.061204   4.442013
```

Looking at the identical means and standard deviations for `injhat`, `exp_idx`, and `exp_idxn`, we see that we can reproduce the default computations of `predict` for `poisson` estimations. We have also demonstrated the relationship between the count predictions and the linear predictions with and without exposure.

◁

SE model scores

▷ Example 6

With most maximum likelihood estimators, `predict` can calculate equation-level scores. The first derivative of the log likelihood with respect to $\mathbf{x}_j\boldsymbol{\beta}$ is the equation-level score.

```
. use http://www.stata-press.com/data/r13/auto, clear
(1978 Automobile Data)
. logistic foreign mpg weight
  (output omitted )
. predict double sc, score
. summarize sc
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
          sc |        74   -1.37e-12    .3533133  -.8760856   .8821309
```

See [P] _robust and [SVY] variance estimation for details regarding the role equation-level scores play in linearization-based variance estimators.

◁

❑ Technical note

predict after some estimation commands, such as regress and cnsreg, allows the score option as a synonym for the residuals option.

❑

Multiple-equation (ME) models

If you have not read the above discussion on using predict after SE models, please do so. With the exception of the ability to select specific equations to predict from, the use of predict after ME models follows almost the same form that it does for SE models.

▷ Example 7

The details of prediction statistics that are specific to particular ME models are documented with the estimation command. If you are using ME commands that do not have separate discussions on obtaining predictions, read *Obtaining predicted values* in [R] **mlogit postestimation**, even if your interest is not in multinomial logistic regression. As a general introduction to the ME models, we will demonstrate predict after sureg:

```
. use http://www.stata-press.com/data/r13/auto, clear
(1978 Automobile Data)
. sureg (price foreign displ) (weight foreign length)
Seemingly unrelated regression
----------------------------------------------------------------------
Equation          Obs  Parms        RMSE    "R-sq"       chi2        P
----------------------------------------------------------------------
price              74      2    2202.447    0.4348      45.21   0.0000
weight             74      2    245.5238    0.8988     658.85   0.0000
----------------------------------------------------------------------

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
price        |
     foreign |   3137.894   697.3805     4.50   0.000     1771.054    4504.735
displacement |   23.06938   3.443212     6.70   0.000     16.32081    29.81795
       _cons |   680.8438   859.8142     0.79   0.428    -1004.361    2366.049
-------------+----------------------------------------------------------------
weight       |
     foreign |   -154.883    75.3204    -2.06   0.040    -302.5082   -7.257674
      length |   30.67594   1.531981    20.02   0.000     27.67331    33.67856
       _cons |  -2699.498   302.3912    -8.93   0.000    -3292.173   -2106.822
------------------------------------------------------------------------------
```

sureg estimated two equations, one called price and the other weight; see [R] **sureg**.

```
. predict pred_p, equation(price)
(option xb assumed; fitted values)

. predict pred_w, equation(weight)
(option xb assumed; fitted values)

. summarize price pred_p weight pred_w

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       price |        74    6165.257    2949.496       3291      15906
      pred_p |        74    6165.257    1678.805    2664.81   10485.33
      weight |        74    3019.459    777.1936       1760       4840
      pred_w |        74    3019.459    726.0468   1501.602   4447.996
```

You may specify the equation by name, as we did above, or by number: equation(#1) means the same thing as equation(price) in this case.

◁

ME model scores

▷ Example 8

For ME models, predict allows you to specify a stub when generating equation-level score variables. predict generates new variables using this stub by appending an equation index. Depending upon the command, the index will start with 0 or 1. Here is an example where predict starts indexing the score variables with 0.

```
. ologit rep78 mpg weight
  (output omitted)

. predict double sc*, scores

. summarize sc*

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         sc0 |        69   -1.33e-11    .5337363  -.9854088    .921433
         sc1 |        69   -7.69e-13     .186919  -.2738537   .9854088
         sc2 |        69   -2.87e-11    .4061637  -.5188487   1.130178
         sc3 |        69   -1.04e-10    .5315368  -1.067351   .8194842
         sc4 |        69    1.47e-10     .360525   -.921433   .6140182
```

Although it involves much more typing, we could also specify the new variable names individually.

```
. predict double (sc_xb sc_1 sc_2 sc_3 sc_4), scores

. summarize sc_*

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       sc_xb |        69   -1.33e-11    .5337363  -.9854088    .921433
        sc_1 |        69   -7.69e-13     .186919  -.2738537   .9854088
        sc_2 |        69   -2.87e-11    .4061637  -.5188487   1.130178
        sc_3 |        69   -1.04e-10    .5315368  -1.067351   .8194842
        sc_4 |        69    1.47e-10     .360525   -.921433   .6140182
```

◁

Methods and formulas

Denote the previously estimated coefficient vector as $\mathbf{b}$ and its estimated variance matrix as $\mathbf{V}$. `predict` works by recalling various aspects of the model, such as $\mathbf{b}$, and combining that information with the data currently in memory. Let's write $\mathbf{x}_j$ for the jth observation currently in memory.

The *predicted value* (`xb` option) is defined as $\widehat{y}_j = \mathbf{x}_j\mathbf{b} + \text{offset}_j$

The *standard error of the prediction* (the `stdp` option) is defined as $s_{p_j} = \sqrt{\mathbf{x}_j\mathbf{V}\mathbf{x}_j'}$

The *standard error of the difference in linear predictions* between equations 1 and 2 is defined as

$$s_{dp_j} = \{(\mathbf{x}_{1j}, -\mathbf{x}_{2j}, \mathbf{0}, \ldots, \mathbf{0})\, \mathbf{V}\, (\mathbf{x}_{1j}, -\mathbf{x}_{2j}, \mathbf{0}, \ldots, \mathbf{0})'\}^{\frac{1}{2}}$$

See the individual estimation commands for information about calculating command-specific `predict` statistics.

Also see

[R] **predictnl** — Obtain nonlinear predictions, standard errors, etc., after estimation

[P] **_predict** — Obtain predictions, residuals, etc., after estimation programming command

[U] **20 Estimation and postestimation commands**