ereturn — Post the estimation results

Description Syntax Options Remarks and examples Stored results Also see

Description

`ereturn local`, `ereturn scalar`, and `ereturn matrix` set the `e()` macros, scalars, and matrices other than `b`, `V`, and `Cns` returned by estimation commands. See [P] **return** for more discussion on returning results.

`ereturn clear` clears the `e()` stored results.

`ereturn list` lists the names and values of the macros and scalars stored in `e()`, and the names and sizes of the matrices stored in `e()` by the last estimation command.

`ereturn post` clears all existing e-class results and stores the coefficient vector (`b`), variance–covariance matrix (`V`), and constraint matrix (`Cns`) in Stata's system areas, making available all the postestimation features described in **[U] 20 Estimation and postestimation commands**. `b`, `V`, and `Cns` are optional for `ereturn post`; some commands (such as `factor`; see [MV] **factor**) do not have a `b`, `V`, or `Cns` but do set the estimation sample, `e(sample)`, and properties, `e(properties)`. You must use `ereturn post` before setting other `e()` macros, scalars, and matrices.

`ereturn repost` changes the `b`, `V`, or `Cns` matrix (allowed only after estimation commands that posted their results with `ereturn post`) or changes the declared estimation sample or `e(properties)`. The specified matrices cease to exist after `post` or `repost`; they are moved into Stata's system areas. The resulting `b`, `V`, and `Cns` matrices in Stata's system areas can be retrieved by reference to `e(b)`, `e(V)`, and `e(Cns)`. `ereturn post` and `repost` deal with only the coefficient and variance–covariance matrices, whereas `ereturn matrix` is used to store other matrices associated with the estimation command.

`ereturn display` displays or redisplays the coefficient table corresponding to results that have been previously posted using `ereturn post` or `repost`.

For a discussion of posting results with constraint matrices (*Cns* in the syntax diagram above), see [P] **makecns**, but only after reading this entry.

Syntax

Set macro returned by estimation command

```
ereturn local name ...                                   (see [P] macro)
```

Set scalar returned by estimation command

```
ereturn scalar name = exp
```

Set matrix returned by estimation command

```
ereturn matrix name [ = ] matname [ , copy ]
```

Clear e() stored results

```
ereturn clear
```

List e() stored results

```
ereturn list [ , all ]
```

Store coefficient vector and variance–covariance matrix into e()

```
ereturn post [ b [ V [ Cns ] ] ] [ weight ] [ , depname(string) obs(#) dof(#)
    esample(varname) properties(string) buildfvinfo findomitted ]
```

Change coefficient vector and variance–covariance matrix

```
ereturn repost [ b = b ] [ V = V ] [ Cns = Cns ] [ weight ] [ , esample(varname)
    properties(string) buildfvinfo findomitted rename resize ]
```

Display coefficient table

```
ereturn display [ , eform(string) first neq(#) plus level(#) display_options ]
```

where *name* is the name of the macro, scalar, or matrix that will be returned in `e(`*name*`)` by the estimation program; *matname* is the name of an existing matrix; **b** is a $1 \times p$ coefficient vector (matrix); **V** is a $p \times p$ covariance matrix; and **Cns** is a $c \times (p + 1)$ constraint matrix.

`fweights`, `aweights`, `iweights`, and `pweights` are allowed; see **[U] 11.1.6 weight**.

Options

copy specified with ereturn matrix indicates that the matrix is to be copied; that is, the original matrix should be left in place.

all specifies that hidden and historical stored results be listed along with the usual stored results. This option is seldom used. See *Using hidden and historical stored results* and *Programming hidden and historical stored results* under *Remarks and examples* of [P] **return** for more information. These sections are written in terms of return list, but everything said there applies equally to ereturn list.

depname(*string*) specified with ereturn post supplies a name that should be that of the dependent variable but can be anything; that name is stored and added to the appropriate place on the output whenever ereturn display is executed.

obs(#) specified with ereturn post supplies the number of observations on which the estimation was performed; that number is stored in e(N).

dof(#) specified with ereturn post supplies the number of (denominator) degrees of freedom that is to be used with t and F statistics and is stored in e(df_r). This number is used in calculating significance levels and confidence intervals by ereturn display and by subsequent test commands performed on the posted results. If the option is not specified, normal (Z) and χ^2 statistics are used.

esample(*varname*) specified with ereturn post or ereturn repost gives the name of the 0/1 variable indicating the observations involved in the estimation. The variable is removed from the dataset but is available for use as e(sample); see **[U] 20.7 Specifying the estimation subsample**. If the esample() option is not specified with ereturn post, it is set to all zeros (meaning no estimation sample). See [P] **mark** for details of the marksample command that can help create *varname*.

properties(*string*) specified with ereturn post or ereturn repost sets the e(properties) macro. By default, e(properties) is set to b V if properties() is not specified.

buildfvinfo specified with ereturn post or ereturn repost computes the H matrix that postestimation commands contrast, margins, and pwcompare use for determining estimable functions.

findomitted specified with ereturn post or ereturn repost adds the omit operator o. to variables in the column names corresponding to zero-valued diagonal elements of e(V). This option is generally unnecessary but is useful when _rmcoll is not used before estimation.

rename is allowed only with the b = **b** syntax of ereturn repost and tells Stata to use the names obtained from the specified **b** matrix as the labels for both the b and V estimation matrices. These labels are subsequently used in the output produced by ereturn display.

resize is allowed only with ereturn repost and tells Stata that the replacements b, V, and Cns have a different number of elements than the originals. This option implies rename.

eform(*string*) specified with ereturn display indicates that the exponentiated form of the coefficients is to be output. *string* is used to label the exponentiated coefficients; see [R] ***eform_option***.

first requests that Stata display only the first equation and make it appear as if only one equation were estimated.

neq(#) requests that Stata display only the first # equations and make it appear as if only # equations were estimated.

plus changes the bottom separation line produced by ereturn display to have a + symbol at the position of the dividing line between variable names and results. This is useful if you plan on adding more output to the table.

level(#), an option of ereturn display, specifies the confidence level, as a percentage, of confidence intervals for the estimated parameters; see **[U] 20.8 Specifying the width of confidence intervals**.

display_options: noci, nopvalues, noomitted, vsquish, noemptycells, baselevels, allbaselevels, nofvlabel, fvwrap(#), fvwrapon(*style*), cformat(%*fmt*), pformat(%*fmt*), sformat(%*fmt*), and nolstretch; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

Estimation-class programs
Setting individual estimation results
Posting estimation coefficient and variance–covariance matrices
Single-equation models
Multiple-equation models
Single-equation models masquerading as multiple-equation models
Setting the estimation sample
Setting estimation-result properties
Reposting results
Minor details: The depname() and dof() options

For a summary of the ereturn command, see [P] **return**.

Estimation-class programs

After any estimation command, you can obtain individual coefficients and standard errors by using _b[] and _se[] (see **[U] 13.5 Accessing coefficients and standard errors**); list the coefficients by using matrix list e(b); list the variance–covariance matrix of the estimators by using matrix list e(V) or in a table by using estat vce (see [R] **estat vce**); obtain the linear prediction and its standard error by using predict (see [R] **predict**); and test linear hypotheses about the coefficients by using test (see [R] **test**). Other important information from an estimation command can be obtained from the stored e() results. (For example, the estimation command name is stored in e(cmd). The dependent variable name is stored in e(depvar).) The e() results from an estimation command can be listed by using the ereturn list command. All of these features are summarized in **[U] 20 Estimation and postestimation commands**.

If you decide to write your own estimation command, your command can share all of these features as well. This is accomplished by posting the results you calculate to Stata. The basic outline of an estimation command is

```
program myest, eclass
        version 19.5      // (or version 19 if you do not have StataNow)
        if !replay() {
                syntax whatever [, whatever Level(cilevel)]
                marksample touse      // see [P] mark
                perform any other parsing of the user's estimation request;
                local depn "dependent variable name"
                local nobs = number of observations in estimation
                tempname b V
                produce coefficient vector `b' and variance–covariance matrix `V'
                ereturn post `b' `V', obs(`nobs') depname(`depn') esample(`touse')
                ereturn local depvar "`depn'"
                store whatever else you want in e()
                ereturn local cmd "myest"     // set e(cmd) last
        }
        else {    // replay
                if "`e(cmd)'"!="myest" error 301
                syntax [, Level(cilevel)]
        }
        output any header above the coefficient table;
        ereturn display, level(`level')
end
```

We will not discuss here how the estimates are formed; see [P] **matrix** for an example of programming linear regression, and see [R] **ml** for examples of programming maximum likelihood estimators. However the estimates are formed, our interest is in posting those results to Stata.

When programming estimation commands, remember to declare them as estimation commands by including the `eclass` option of `program`; see **[U] 18 Programming Stata**. If you do not declare your program to be `eclass`, Stata will produce an error if you use `ereturn local`, `ereturn scalar`, or `ereturn matrix` in your program. For more information about storing program results, see [P] **return**.

The estimation program definition statement—`program` *myest*`, eclass`—should also have included a `properties()` option, but we omitted it because 1) it is not necessary and 2) you might confuse it with `ereturn`'s `properties()` option.

There are two sets of properties associated with estimation commands: program properties and estimation-result properties. The first are set by the `properties()` option of the `program` definition statement. The second are set by `ereturn`'s `properties()` option. The first tell Stata's prefix commands, such as `stepwise` and `svy`, whether they should work with this new estimation command. The second tell Stata's postestimation commands, such as `predict` and `test`, whether they should work after this new estimation command.

The first is discussed in [P] **program properties**. The second will be discussed below.

❑ Technical note

Notice the use of the `replay()` function in our estimation program example. This function is not like other Stata functions; see [FN] **Programming functions**. `replay()` simply returns 1 if the command line is empty or begins with a comma, and 0 otherwise. More simply: `replay()` indicates whether the command is an initial call to the estimation program (`replay()` returns 0) or a call to redisplay past estimation results (`replay()` returns 1).

In fact,

```
if !replay() {
```

is equivalent to

```
if trim(`""0""') == "" | substr(trim(`""0""'),1,1) == "," {
```

but is easier to read.

❑

The ereturn local, ereturn scalar, ereturn matrix, ereturn clear, and ereturn list commands are discussed in *Setting individual estimation results*. The ereturn post, ereturn repost, and ereturn display commands are discussed in *Posting estimation coefficient and variance–covariance matrices*.

Setting individual estimation results

Stata's estimation commands store the command name in the returned macro e(cmd) and store the name of the dependent variable in e(depvar). Other macros and scalars are also stored. For example, the estimation sample size is stored in the returned scalar e(N). The model and residual degrees of freedom are stored in e(df_m) and e(df_r).

These e() macro and scalar results are stored using the ereturn local and ereturn scalar commands. Matrices may be stored using the ereturn matrix command. The coefficient vector e(b) and variance–covariance matrix e(V), however, are handled differently and are stored using only the ereturn post and ereturn repost commands, which are discussed in the next section.

▷ Example 1

Assume that we are programming an estimation command called xyz and that we have the dependent variable in `depname', the estimation sample size in `nobs', and other important information stored in other local macros and scalars. We also wish to store an auxiliary estimation matrix that our program has created called lam into the stored matrix e(lambda). We would store these results by using commands such as the following in our estimation program:

```
...
ereturn local depvar "`depname'"
ereturn scalar N = `nobs'
ereturn matrix lambda lam
...
ereturn local cmd "xyz"
```

◁

The matrix given to the ereturn matrix command is removed, and the new e() matrix is then made available. For instance, in this example, we have the line

```
ereturn matrix lambda lam
```

After this line has executed, the matrix lam is no longer available for use, but you can instead refer to the newly created e(lambda) matrix.

The e() results from an estimation command can be viewed using the ereturn list command.

▷ Example 2

We regress automobile weight on length and engine displacement by using the auto dataset.

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)
. regress weight length displ
```

Source	SS	df	MS	Number of obs	=	74
				F(2, 71)	=	480.99
Model	41063449.8	2	20531724.9	Prob > F	=	0.0000
Residual	3030728.55	71	42686.3176	R-squared	=	0.9313
				Adj R-squared	=	0.9293
Total	44094178.4	73	604029.841	Root MSE	=	206.61

weight	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
length	22.91788	1.974431	11.61	0.000	18.98097	26.85478
displacement	2.932772	.4787094	6.13	0.000	1.978252	3.887291
_cons	-1866.181	297.7349	-6.27	0.000	-2459.847	-1272.514

```
. ereturn list
scalars:
                  e(N) =  74
              e(sum_w) =  74
               e(df_m) =  2
               e(df_r) =  71
                  e(F) =  480.9907735088096
                 e(r2) =  .9312669232040125
               e(rmse) =  206.6066736285298
                e(mss) =  41063449.82964133
                e(rss) =  3030728.548737053
               e(r2_a) =  .9293307801956748
                 e(ll) =  -497.9506459758983
               e(ll_0) =  -597.0190609278627
               e(rank) =  3
macros:
            e(cmdline) : "regress weight length displ"
              e(title) : "Linear regression"
          e(marginsok) : "XB default"
                e(vce) : "ols"
            e(depvar) : "weight"
               e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
              e(model) : "ols"
         e(estat_cmd) : "regress_estat"
matrices:
                  e(b) :  1 x 3
                  e(V) :  3 x 3
               e(beta) :  1 x 2
functions:
             e(sample)
```

In addition to listing all the e() results after an estimation command, you can access individual e() results.

```
. display "The command is: `e(cmd)'"
The command is: regress
. display "The adjusted R-squared is: `e(r2_a)'"
The adjusted R-squared is: .9293307801956748
. display "The residual sums-of-squares is: `e(rss)'"
The residual sums-of-squares is: 3030728.548737053
. matrix list e(V)
symmetric e(V)[3,3]
                  length  displacement         _cons
      length     3.8983761
displacement    -.78935643     .22916272
       _cons    -576.89342     103.13249     88646.064
. matrix list e(b)
e(b)[1,3]
          length  displacement         _cons
y1     22.917876     2.9327718    -1866.1807
```

For more information on referring to e() results, see [P] **return**.

◁

The reference manuals' entries for Stata's estimation commands have a *Stored results* section describing the e() results that are returned by the command. If you are writing an estimation command, we recommend that you store the same kind of estimation results by using the same naming convention as Stata's estimation commands. This is important if you want postestimation commands to work after your estimation command. See **[U] 20 Estimation and postestimation commands** and [P] **return** for details.

When programming your estimation command, you will want to issue either an ereturn clear command or an ereturn post command before you store any estimation results. The ereturn clear command clears all e() results. The ereturn post command, which is discussed in the next section, first clears all previous e() results and then performs the post.

We recommend that you postpone clearing past estimation results and setting new e() results until late in your program. If an error occurs early in your program, the last successful estimation results will remain intact. The best place in your estimation program to set the e() results is after all other calculations have been completed and before estimation results are displayed.

We also recommend that you store the command name in e(cmd) as your last act of storing results. This ensures that if e(cmd) is present, then all the other estimation results were successfully stored. Postestimation commands assume that if e(cmd) is present, then the estimation command completed successfully and all expected results were stored. If you stored e(cmd) early in your estimation command and the user pressed *Break* before the remaining e() results were stored, postestimation commands operating on the partial results will probably produce an error.

Posting estimation coefficient and variance–covariance matrices

The most important estimation results are the coefficient vector b and the variance–covariance matrix V. Because these two matrices are at the heart of most estimation commands, for increased command execution speed, Stata handles these matrices in a special way. The ereturn post, ereturn repost, and ereturn display commands work on these matrices. The ereturn matrix command discussed in the last section cannot be used to store or to post the b and V matrices.

Single-equation models

Before posting, the coefficient vector is stored as a $1 \times p$ matrix and the corresponding variance–covariance matrix as a $p \times p$ matrix. The names bordering the coefficient matrix and those bordering the variance–covariance matrix play an important role. First, they must be the same. Second, it is these names that tell Stata how the results link to Stata's other features.

Estimation results come in two forms: those for single-equation models and those for multiple-equation models. The absence or presence of equation names in the names bordering the matrix (see [P] **matrix rownames**) tells Stata which form it is.

▷ Example 3

For instance, consider

```
. use https://www.stata-press.com/data/r19/auto
(1978 automobile data)
. regress price weight mpg
  (output omitted)
. matrix b = e(b)
. matrix V = e(V)
. matrix list b
b[1,3]
        weight         mpg       _cons
y1   1.7465592  -49.512221   1946.0687
. matrix list V
symmetric V[3,3]
            weight         mpg       _cons
weight   .41133468
   mpg   44.601659    7422.863
 _cons  -2191.9032  -292759.82    12938766
```

If these were our estimation results, they would correspond to a single-equation model because the names bordering the matrices have no equation names. Here we post these results:

```
. ereturn post b V
. ereturn display
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
weight	1.746559	.6413538	2.72	0.006	.4895288	3.003589
mpg	-49.51222	86.15604	-0.57	0.566	-218.375	119.3505
_cons	1946.069	3597.05	0.54	0.588	-5104.019	8996.156

Once the results have been posted, anytime the `ereturn display` command is executed, Stata will redisplay the coefficient table. Moreover, all of Stata's other postestimation features work. For instance,

```
. test weight
 ( 1)  weight = 0
           chi2(  1) =    7.42
         Prob > chi2 =    0.0065
. test weight=mpg/50
 ( 1)  weight - .02*mpg = 0
           chi2(  1) =    4.69
         Prob > chi2 =    0.0303
```

If the user were to type `predict pred`, then `predict` would create a new variable based on

$$1.746559\,\texttt{weight} - 49.51222\,\texttt{mpg} + 1946.069$$

except that it would carry out the calculation by using the full, double-precision values of the coefficients. All determinations are made by Stata on the basis of the names bordering the posted matrices.

◁

Multiple-equation models

If the matrices posted using the `ereturn post` or `ereturn repost` commands have more than one equation name, the estimation command is treated as a multiple-equation model.

▷ Example 4

Consider the following two matrices before posting:

```
. matrix list b
b[1,6]
           price:        price:        price:  displacem~t:  displacem~t:
          weight           mpg         _cons        weight       foreign
y1     1.7417059     -50.31993     1977.9249     .09341608    -35.124241

     displacem~t:
           _cons
y1    -74.326413
. matrix list V
symmetric V[6,6]
                             price:        price:        price:  displacem~t:
                            weight           mpg         _cons        weight
        price:weight     .38775906
           price:mpg     41.645165     6930.8263
         price:_cons    -2057.7522    -273353.75      12116943
 displacement:weight     .00030351    -.01074361    -.68762197     .00005432
displacement:foreign    -.18390487      -30.6065      1207.129     .05342871
  displacement:_cons    -.86175743     41.539129     1936.6875     -.1798972

                        displacem~t:  displacem~t:
                           foreign         _cons
displacement:foreign     152.20821
  displacement:_cons    -206.57691     625.79842
```

The row and column names of the matrices include equation names. Here we post these matrices to Stata and then use the posted results:

```
. ereturn post b V
. ereturn display
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
price						
weight	1.741706	.622703	2.80	0.005	.5212304	2.962181
mpg	-50.31993	83.25158	-0.60	0.546	-213.49	112.8502
_cons	1977.925	3480.94	0.57	0.570	-4844.592	8800.442
displacement						
weight	.0934161	.0073701	12.67	0.000	.0789709	.1078612
foreign	-35.12424	12.33727	-2.85	0.004	-59.30484	-10.94364
_cons	-74.32641	25.01596	-2.97	0.003	-123.3568	-25.29603

```
. test [price]weight
 ( 1)  [price]weight = 0
           chi2(  1) =    7.82
         Prob > chi2 =    0.0052
. test weight
 ( 1)  [price]weight = 0
 ( 2)  [displacement]weight = 0
           chi2(  2) =  164.51
         Prob > chi2 =    0.0000
```

Stata determined that this was a multiple-equation model because equation names were present. All of Stata's equation-name features (such as those available with the `test` command) are then made available. The user could type `predict pred` to obtain linear predictions of the `[price]` equation (because `predict` defaults to the first equation) or type `predict pred, equation(displ)` to obtain predictions of the `[displ]` equation:

$$0.0934161\,\texttt{weight} - 35.12424\,\texttt{foreign} - 74.32641$$

◁

Single-equation models masquerading as multiple-equation models

▷ Example 5

Sometimes, it may be convenient to program a single-equation model as if it were a multiple-equation model. This occurs when there are ancillary parameters. Think of linear regression: in addition to the parameter estimates, there is s, which is an estimate of σ, the standard error of the residual. This can be calculated on the side in that you can calculate $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$ independently of s and then calculate s given $\mathbf{b}$. Pretend that were not the case—think of a straightforward maximum likelihood calculation where s is just one more parameter (in most models, ancillary parameters and the coefficients must be solved for jointly). The right thing to do would be to give s its own equation:

```
. matrix list b
b[1,4]
         price:      price:      price:       _anc:
        weight         mpg       _cons       sigma
y1   1.7465592  -49.512221   1946.0687        2514
. matrix list V
symmetric V[4,4]
                   price:      price:      price:       _anc:
                  weight         mpg       _cons       sigma
price:weight   .41133468
   price:mpg   44.601659    7422.863
 price:_cons  -2191.9032  -292759.82    12938766
  _anc:sigma           0           0           0      810000
. ereturn post b V
```

```
. ereturn display
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
price						
weight	1.746559	.6413538	2.72	0.006	.4895288	3.003589
mpg	-49.51222	86.15604	-0.57	0.566	-218.375	119.3505
_cons	1946.069	3597.05	0.54	0.588	-5104.019	8996.156
_anc						
sigma	2514	900	2.79	0.005	750.0324	4277.968

Now consider the alternative, which would be simply to add s to the estimated parameters without equation names:

```
. matrix list b
b[1,4]
        weight         mpg       _cons       sigma
y1   1.7465592  -49.512221   1946.0687        2514
. matrix list V
symmetric V[4,4]
             weight         mpg       _cons       sigma
weight   .41133468
   mpg   44.601659    7422.863
 _cons  -2191.9032  -292759.82    12938766
 sigma           0           0           0      810000
. ereturn post b V
. ereturn display
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
weight	1.746559	.6413538	2.72	0.006	.4895288	3.003589
mpg	-49.51222	86.15604	-0.57	0.566	-218.375	119.3505
_cons	1946.069	3597.05	0.54	0.588	-5104.019	8996.156
sigma	2514	900	2.79	0.005	750.0324	4277.968

This second solution is inferior because, if the user typed `predict pred`, then `predict` would attempt to form the linear combination:

$$1.746559\,\texttt{weight} - 49.51222\,\texttt{mpg} + 1946.069 + 2514\,\texttt{sigma}$$

There are only two possibilities, and neither is good: either `sigma` does not exist in the dataset—which is to be hoped—and `predict` produces the error message "variable sigma not found", or something called `sigma` does exist, and `predict` goes on to form this meaningless combination.

◁

On the other hand, if the parameter estimates are separated from the ancillary parameter (which could be parameters) by the equation names, the user can type `predict pred, equation(price)` to obtain a meaningful result. Moreover, the user can omit `equation(price)` partly because `predict` (and Stata's other postestimation commands) defaults to the first equation.

We recommend that ancillary parameters be collected together and given their own equation and that the equation be called `_anc`.

Setting the estimation sample

In our previous examples, we did not indicate the estimation sample as specified with the `esample(`*varname*`)` option. In general, you provide this either with your initial `ereturn post` command or with a subsequent `ereturn repost` command. Some postestimation commands automatically restrict themselves to the estimation sample, and if you do not provide this information, they will complain that there are no observations; see **[U] 20.7 Specifying the estimation subsample**. Also, users of your estimation command expect to use `if e(sample)` successfully in commands that they execute after your estimation command.

Example 6

Returning to our first example:

```
. ereturn post b V
. ereturn display
  (output omitted)
. summarize price if e(sample)
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       price |          0
```

does not produce what the user expects. Specifying the estimation sample with the `esample()` option of `ereturn post` produces the expected result:

```
. ereturn post b V, esample(estsamp)
. ereturn display
  (output omitted)
. summarize price if e(sample)
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496       3291      15906
```

The `marksample` command (see **[P] mark**) is a useful programming command that aids in creating and setting up an estimation sample indicator variable, such as `estsamp`.

Setting estimation-result properties

The `properties()` option of `ereturn post` and `repost` allows you to set `e(properties)`. By default, `ereturn post` sets `e(properties)` to `b V` when you supply a **b** and **V** argument. If you supply the **b**, but not the **V**, it defaults to `b`. If you do not supply the **b** and **V**, it defaults to being empty. Using the `properties()` option, you can augment or override the default setting. You are also free to use `ereturn local` to set `e(properties)`.

`e(properties)` is used as a signal to postestimation commands. A `b` in `e(properties)` is a signal that the `e(b)` returned matrix can be interpreted as a coefficient vector. A `V` in `e(properties)` indicates that `e(V)` can be interpreted as a VCE matrix. An `e(properties)` containing `eigen` indicates that the estimation command has placed eigenvalues in `e(Ev)` and eigenvectors in `e(L)`. A command, such as `screeplot` (see **[MV] screeplot**), that plots the eigenvalues and can be used as a postestimation command looks to see if `eigen` is found in `e(properties)`. If so, it then looks for `e(Ev)` to contain the eigenvalues.

▷ Example 7

We demonstrate by interactively posting a **b** vector without posting a **V** matrix. Even without a **V** matrix, the available information provided by **b** is used appropriately.

```
. use https://www.stata-press.com/data/r19/auto, clear
(1978 automobile data)
. matrix b=(2,-1)
. matrix colnames b = turn trunk
. ereturn post b
. ereturn display
```

	Coefficient
turn	2
trunk	-1

```
. predict myxb, xb
. list turn trunk myxb in 1/4
```

	turn	trunk	myxb
1.	40	11	69
2.	40	11	69
3.	35	12	58
4.	40	16	64

The estimation table produced by `ereturn display` omits the standard errors, tests, and confidence intervals because they rely on having a VCE matrix. `predict` with the `xb` option produces the linear predictions. If you tried to use the `stdp` option of `predict`, you would get an error message indicating that the requested action was not valid.

◁

The `has_eprop()` programmer's function is useful for determining if `e(properties)` contains a particular property; see [FN] **Programming functions**.

❑ Technical note

Do not confuse the properties set with the `properties()` option of `ereturn post` and `ereturn repost`, which are placed in `e(properties)` and used by postestimation commands, with the `properties()` option of the `program` command; see [P] **program**. The properties set by `program` indicate to other programs before the command is executed that certain features have been implemented, for example, the `svyr` property indicates to the `svy` prefix command that the requirements to use the `vce(linearized)` variance estimation method have been satisfied. On the other hand, the properties set by `ereturn` are for use after the program has run and may depend on the data and options of the program.

❑

Reposting results

In certain programming situations, only a small part of a previous estimation result needs to be altered. `ereturn repost` allows us to change five parts of an estimation result that was previously posted with `ereturn post`. We can change the coefficient vector, the variance–covariance matrix, and the declared estimation sample by using the `esample()` option; we can change the declared properties by using the

properties() option; and we can change the variable names for the coefficients by using the rename option. A programmer might, for instance, simply replace the variance–covariance matrix provided by a previous ereturn post with a robust covariance matrix to create a new estimation result.

Sometimes a programmer might preserve the data, make major alterations to the data (using drop, reshape, etc.) to perform needed computations, post the estimation results, and then finally restore the data. Here, when ereturn post is called, the correct estimation sample indicator variable is unavailable. ereturn repost with the esample() option allows us to set the estimation sample without changing the rest of our posted estimation results.

▷ Example 8

For example, inside an estimation command program, we might have

```
...
ereturn post b V
...
ereturn repost, esample(estsamp)
...
```

◁

❑ Technical note

ereturn repost may be called only from within a program that has been declared an estimation class program by using the eclass option of the program statement. The same is not true of ereturn post. We believe that the only legitimate uses of ereturn repost are in a programming context. ereturn post, on the other hand, may be important for some non–e-class programming situations.

❑

Minor details: The depname() and dof() options

Single-equation models may have one dependent variable; in those that do, you should specify the identity of this one dependent variable in the depname() option with ereturn post. The result is simply to add a little more labeling to the output.

If you do not specify the dof(#) option at the time of posting or set e(df_r) equal to the degrees of freedom, normal (Z) statistics will be used to calculate significance levels and confidence intervals on subsequent ereturn display output. If you do specify dof(#) or set e(df_r) equal to #, t statistics with # degrees of freedom will be used. Similarly, if you did not specify dof(#) or set e(df_r), any subsequent test commands will present a χ^2 statistic; if you specify dof(#) or set e(df_r), subsequent test commands will use the F statistic with # denominator degrees of freedom.

▷ Example 9

Let's add the dependent variable name and degrees of freedom to example 3.

```
. ereturn post b V, depname(price) dof(71)
. ereturn display
```

price	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
weight	1.746559	.6413538	2.72	0.008	.467736	3.025382
mpg	-49.51222	86.15604	-0.57	0.567	-221.3025	122.278
_cons	1946.069	3597.05	0.54	0.590	-5226.245	9118.382

Note the addition of the word price at the top of the table. This was produced because of the depname(price) option specification. Also t statistics were used instead of normal (Z) statistics because the dof(71) option was specified.

◁

Stored results

ereturn post stores the following in e():

Scalars
- e(N) number of observations
- e(df_r) degrees of freedom, if specified

Macros
- e(wtype) weight type
- e(wexp) weight expression
- e(properties) estimation properties; typically b V

Matrices
- e(b) coefficient vector
- e(Cns) constraints matrix
- e(V) variance–covariance matrix of the estimators

Functions
- e(sample) marks estimation sample

ereturn repost stores the following in e():

Macros
- e(wtype) weight type
- e(wexp) weight expression
- e(properties) estimation properties; typically b V

Matrices
- e(b) coefficient vector
- e(Cns) constraints matrix
- e(V) variance–covariance matrix of the estimators

Functions
- e(sample) marks estimation sample

With ereturn post, all previously stored estimation results—e() items—are removed. ereturn repost, however, does not remove previously stored estimation results. ereturn clear removes the current e() results.

ereturn display stores the following in r():

Scalars
- r(level) confidence level of confidence intervals

Macros
- r(label#) label on the # coefficient, such as (base), (omitted), (empty), or (constrained)
- r(table) information from the coefficient table (see below)

`r(table)` contains the following information for each coefficient:

`b`	coefficient value
`se`	standard error
`t/z`	test statistic for coefficient
`pvalue`	observed significance level for `t/z`
`ll`	lower limit of confidence interval
`ul`	upper limit of confidence interval
`df`	degrees of freedom associated with coefficient
`crit`	critical value associated with `t/z`
`eform`	indicator for exponentiated coefficients

Also see

[P] **_estimates** — Manage estimation results

[P] **return** — Return stored results

[R] **estimates** — Save and manipulate estimation results

[U] 18 Programming Stata

[U] 18.9 Accessing results calculated by estimation commands

[U] 18.10.2 Storing results in e()

[U] 20 Estimation and postestimation commands

Stata, Stata Press, Mata, NetCourse, and NetCourseNow are registered trademarks of StataCorp LLC. Stata and Stata Press are registered trademarks with the World Intellectual Property Organization of the United Nations. StataNow is a trademark 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.