Title

stata.com

stored results — Stored results

Description Syntax Option Remarks and examples References Also see

Description

Results of calculations are stored by many Stata commands so that they can be easily accessed and substituted into later commands.

`return list` lists results stored in `r()`.

`ereturn list` lists results stored in `e()`.

`sreturn list` lists results stored in `s()`.

This entry discusses using stored results. Programmers wishing to store results should see [P] **return** and [P] **ereturn**.

Syntax

List results from general commands, stored in r()

`return list [, all]`

List results from estimation commands, stored in e()

`ereturn list [, all]`

List results from parsing commands, stored in s()

`sreturn list`

Option

`all` is for use with `return list` and `ereturn list`. `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`.

`all` is not allowed with `sreturn list` because `s()` does not allow hidden or historical results.

Remarks and examples

stata.com

Stata commands are classified as being

r-class	general commands that store results in `r()`
e-class	estimation commands that store results in `e()`
s-class	parsing commands that store results in `s()`
n-class	commands that do not store in `r()`, `e()`, or `s()`

There is also a c-class, `c()`, containing the values of system parameters and settings, along with certain constants, such as the value of pi; see [P] **creturn**. A program, however, cannot be c-class.

You can look at the *Stored results* section of the manual entry of a command to determine whether it is r-, e-, s-, or n-class, but it is easy enough to guess.

Commands producing statistical results are either r-class or e-class. They are e-class if they present estimation results and r-class otherwise. s-class is a class used by programmers and is primarily used in subprograms performing parsing. n-class commands explicitly state where the result is to go. For instance, `generate` and `replace` are n-class because their syntax is `generate` *varname* `=` ... and `replace` *varname* `=` ....

After executing a command, you can type `return list`, `ereturn list`, or `sreturn list` to see what has been stored.

▷ Example 1

```
. use http://www.stata-press.com/data/r15/auto4
(1978 Automobile Data)
. describe
Contains data from http://www.stata-press.com/data/r15/auto4.dta
  obs:            74                          1978 Automobile Data
 vars:             6                          6 Apr 2016 00:20
 size:         2,072
-------------------------------------------------------------------------------
              storage   display    value
variable name   type    format     label      variable label
-------------------------------------------------------------------------------
price           int     %8.0gc                Price
weight          int     %8.0gc                Weight (lbs.)
mpg             int     %8.0g                 Mileage (mpg)
make            str18   %-18s                 Make and Model
length          int     %8.0g                 Length (in.)
rep78           int     %8.0g                 Repair Record 1978
-------------------------------------------------------------------------------
Sorted by:
. return list
scalars:
            r(changed) =  0
              r(width) =  28
                  r(k) =  6
                  r(N) =  74
```

To view all stored results, including those that are historical or hidden, specify the `all` option.

```
. return list, all
scalars:
            r(changed) =  0
              r(width) =  28
                  r(k) =  6
                  r(N) =  74
Historical; used before Stata 12, may exist only under version control
scalars:
           r(widthmax) =  1048576
              r(k_max) =  2048
              r(N_max) =  2147483619
```

`r(widthmax)`, `r(k_max)`, and `r(N_max)` are historical stored results. They are no longer relevant because Stata dynamically adjusts memory beginning with Stata 12.

◁

❑ Technical note

In the above example, we stated that `r(widthmax)` and `r(N_max)` are no longer relevant. In fact, they are not useful. Stata no longer has a fixed memory size, so the methods used to calculate `r(widthmax)` and `r(N_max)` are no longer appropriate.

❑

▷ Example 2

You can use stored results in expressions.

```
. summarize mpg
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         mpg |         74     21.2973    5.785503        12         41
. return list
scalars:
                  r(N) =  74
              r(sum_w) =  74
               r(mean) =  21.2972972972973
                r(Var) =  33.47204738985561
                 r(sd) =  5.785503209735141
               r(min) =  12
               r(max) =  41
               r(sum) =  1576
. generate double mpgstd = (mpg-r(mean))/r(sd)
. summarize mpgstd
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      mpgstd |         74   -1.64e-16           1  -1.606999    3.40553
```

Be careful to use results stored in `r()` soon because they will be replaced the next time you execute another r-class command. For instance, although `r(mean)` was 21.3 (approximately) after `summarize mpg`, it is −1.64e–16 now because you just ran `summarize` with `mpgstd`.

◁

▷ Example 3

e-class is really no different from r-class, except for where results are stored and that, when an estimation command stores results, it tends to store a lot of them:

```
. regress mpg weight length
  (output omitted)
. ereturn list
scalars:
                  e(N) =  74
               e(df_m) =  2
               e(df_r) =  71
                  e(F) =  69.34050004300227
                e(r2) =  .6613903979336323
              e(rmse) =  3.413681741382589
               e(mss) =  1616.08062422659
               e(rss) =  827.3788352328695
              e(r2_a) =  .6518520992838754
                e(ll) =  -194.3267619410807
              e(ll_0) =  -234.3943376482347
              e(rank) =  3
```

```
macros:
            e(cmdline) : "regress mpg weight length"
              e(title) : "Linear regression"
          e(marginsok) : "XB default"
                e(vce) : "ols"
            e(depvar) : "mpg"
                e(cmd) : "regress"
         e(properties) : "b V"
            e(predict) : "regres_p"
         e(estat_cmd) : "regress_estat"
matrices:
                  e(b) :  1 x 3
                  e(V) :  3 x 3
functions:
             e(sample)
```

These e-class results will stick around until you run another estimation command. Typing `return list` and `ereturn list` is the easy way to find out what a command stores.

◁

Both r- and e-class results come in four flavors: scalars, macros, matrices, and functions. (s-class results come in only one flavor—macros—and as earlier noted, s-class is used solely by programmers, so ignore it.)

Scalars are just that—numbers by any other name. You can subsequently refer to `r(mean)` or `e(rmse)` in numeric expressions and obtain the result to full precision.

Macros are strings. For instance, `e(depvar)` contains "`mpg`". You can refer to it, too, in subsequent expressions, but really that would be of most use to programmers, who will refer to it using constructs like `"‘e(depvar)’"`. In any case, macros are macros, and you obtain their contents just as you would a local macro, by enclosing their name in single quotes. The name here is the full name, so `‘e(depvar)’` is `mpg`.

Matrices are matrices, and all estimation commands store `e(b)` and `e(V)` containing the coefficient vector and variance–covariance matrix of the estimates (VCE).

Functions are stored by e-class commands only, and the only function existing is `e(sample)`. `e(sample)` evaluates to 1 (meaning true) if the observation was used in the previous estimation and to 0 (meaning false) otherwise.

❑ Technical note

Say that some command set `r(scalar)` and `r(macro)`, the first being stored as a scalar and the second as a macro. In theory, in subsequent use you are supposed to refer to `r(scalar)` and `‘r(macro)’`. In fact, however, you can refer to either one with or without quotes, so you could refer to `‘r(scalar)’` and `r(macro)`. Programmers sometimes do this.

When you refer to `r(scalar)`, you are referring to the full double-precision stored result. Think of `r(scalar)` without quotes as a function returning the value of the stored result `scalar`. When you refer to `r(scalar)` in quotes, Stata understands `‘r(scalar)’` to mean "substitute the printed result of evaluating `r(scalar)`". Pretend that `r(scalar)` equals the number 23. Then `‘r(scalar)’` is 23, the character 2 followed by 3.

Referring to `r(scalar)` in quotes is sometimes useful. Say that you want to use the immediate command `cii` with `r(scalar)`. The immediate command `cii` requires its arguments to be numbers—numeric literals in programmer's jargon—and it will not take an expression. Thus you could not type ‘`cii r(scalar) ...`’. You could, however, type ‘`cii ‘r(scalar)’ ...`’ because `‘r(scalar)’` is just a numeric literal.

For `r(macro)`, you are supposed to refer to it in quotes: `‘r(macro)’`. If, however, you omit the quotes in an expression context, Stata evaluates the macro and then pretends that it is the result of function-returning-string. There are side effects of this, the most important being that the result is trimmed to 80 characters.

Referring to `r(macro)` without quotes is never a good idea; the feature was included merely for completeness.

You can even refer to `r(matrix)` in quotes (assume that `r(matrix)` is a matrix). `‘r(matrix)’` does not result in the matrix being substituted; it returns the word `matrix`. Programmers sometimes find that useful.

❑

References

Jann, B. 2005. Making regression tables from stored estimates. *Stata Journal* 5: 288–308.

——. 2007. Making regression tables simplified. *Stata Journal* 7: 227–244.

Also see

[P] **ereturn** — Post the estimation results

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

[U] 18.8 Accessing results calculated by other programs

[U] 18.9 Accessing results calculated by estimation commands