

Description

These commands are utilities to assist Stata programmers in performing Monte Carlo–type experiments. They post results to a file on disk. To post results to a frame in memory, see [\[P\] frame post](#).

`postfile` declares the variable names and the filename of a (new) Stata dataset where results will be saved.

`post` adds a new observation to the declared dataset.

`postclose` declares an end to the posting of observations. After `postclose`, the new dataset contains the posted results and may be loaded using `use`; see [\[D\] use](#).

`postutil dir` lists all open postfiles. `postutil clear` closes all open postfiles.

All five commands manipulate the new dataset without disturbing the data in memory.

If *filename* is specified without an extension, `.dta` is assumed.

Syntax

Declare variable names and filename of dataset where results will be saved

```
postfile postname newvarlist using filename [ , every(#) replace ]
```

Add new observation to declared dataset

```
post postname (exp) (exp) ... (exp)
```

Declare end to posting of observations

```
postclose postname
```

List all open postfiles

```
postutil dir
```

Close all open postfiles

```
postutil clear
```

Options

`every(#)` specifies that results be written to disk every `#`th call to `post`. `post` temporarily holds results in memory and periodically opens the Stata dataset being built to append the saved results. `every()` should typically not be specified, because you are unlikely to choose a value for `#` that is as efficient as the number `post` chooses on its own, which is a function of the number of results being written and their storage type.

`replace` indicates that the file specified may already exist, and if it does, that `postfile` may erase the file and create a new one.

Remarks and examples

The typical use of the `post` commands is

```
tempname memhold
tempfile results
...
postfile 'memhold' ... using "results"
...
while ... {
    ...
    post 'memhold' ...
    ...
}
postclose 'memhold'
...
use "results", clear
...
```

Two names are specified with `postfile`: *postname* is a name assigned to internal memory buffers, and *filename* is the name of the file to be created. Subsequent `posts` and the `postclose` are followed by *postname* so that Stata will know to what file they refer.

In our sample, we obtain both names from Stata's temporary name facility (see [P] [macro](#)), although, in some programming situations, you may wish to substitute a hard-coded *filename*. We recommend that *postname* always be obtained from `tempname`. This ensures that your program can be nested within any other program and ensures that the memory used by `post` is freed if anything goes wrong. Using a temporary filename, too, ensures that the file will be erased if the user presses *Break*. Sometimes, however, you may wish to leave the file of incomplete results behind. That is allowed, but remember that the file is not fully up to date if `postclose` has not been executed. `post` buffers results in memory and only periodically updates the file.

Because `postfile` accepts a *newvarlist*, storage types may be interspersed, so you could have

```
postfile 'memhold' a b str20 c double(d e f) using "results"
```

Note that *newvarlist* does not allow `strL` as the variable storage type. A similar utility that allows `strL` as a variable storage type is [frame post](#).

► Example 1

We wish to write a program to collect means and variances from 10,000 randomly constructed 100-observation samples of lognormal data and save the results in `results.dta`. Suppose that we are evaluating the coverage of the 95%, t -based confidence interval when applied to lognormal data. As background, we can obtain a 100-observation lognormal sample by typing

```
drop _all
set obs 100
generate z = exp(rnormal())
```

We can obtain the mean and standard deviation by typing

```
summarize z
```

Moreover, `summarize` stores the sample mean in `r(mean)` and variance in `r(Var)`. It is those two values we wish to collect. Our program is

```
program lnsim
version 19.5      // (or version 19 if you do not have StataNow)
tempname sim
postfile 'sim' mean var using results, replace
quietly {
    forvalues i = 1/10000 {
        drop _all
        set obs 100
        generate z = exp(rnormal())
        summarize z
        post 'sim' (r(mean)) (r(Var))
    }
}
postclose 'sim'
end
```

The `postfile` command begins the accumulation of results. `'sim'` is the name assigned to the internal memory buffers where results will be held; `mean` and `var` are the names to be given to the two variables that will contain the information we collect; and variables will be saved in the file named `results.dta`. Because two variable names were specified on the `postfile` line, two expressions must be specified following `post`. Here the expressions are simply `r(mean)` and `r(Var)`. If we had wanted, however, to store the mean divided by the standard deviation and the standard deviation, we could have typed

```
post 'sim' (r(mean)/r(sd)) (r(sd))
```

Finally, postclose ‘sim’ concluded the simulation. The dataset results.dta is now complete.

```
. set seed 12345
. lnsim
(file results.dta not found)
. use results, clear
. describe
```

Contains data from results.dta

Observations:	10,000			
Variables:	2			27 Mar 2025 18:53

Variable name	Storage type	Display format	Value label	Variable label
mean	float	%9.0g		
var	float	%9.0g		

Sorted by:

```
. summarize
```

Variable	Obs	Mean	Std. dev.	Min	Max
mean	10,000	1.649184	.217526	.9933856	3.087867
var	10,000	4.624283	4.144868	.6665277	97.41853

We set the random-number seed to an arbitrary value, 12345, so that this example would be reproducible.

◀

References

Drukker, D. M. 2015a. Efficiency comparisons by Monte Carlo simulation. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2015/10/13/efficiency-comparisons-by-monte-carlo-simulation/>.

———. 2015b. Monte Carlo simulations using Stata. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2015/10/06/monte-carlo-simulations-using-stata/>.

———. 2015c. Understanding the generalized method of moments (GMM): A simple example. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2015/12/03/understanding-the-generalized-method-of-moments-gmm-a-simple-example/>.

———. 2016. A simulation-based explanation of consistency and asymptotic normality. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2016/04/18/a-simulation-based-explanation-of-consistency-and-asymptotic-normality/>.

Pinzon, E. 2016a. probit or logit: Ladies and gentlemen, pick your weapon. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2016/01/07/probit-or-logit-ladies-and-gentlemen-pick-your-weapon/>.

———. 2016b. regress, probit, or logit? *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2016/01/14/regress-probit-or-logit/>.

Rajbhandari, A. 2016. ARMA processes with nonnormal disturbances. *The Stata Blog: Not Elsewhere Classified*. <https://blog.stata.com/2016/05/04/arma-processes-with-nonnormal-disturbances/>.

Van Kerm, P. 2007. Stata tip 54: Post your results. *Stata Journal* 7: 587–589.

Also see

- [P] [frame post](#) — Post results to dataset in another frame
- [R] [simulate](#) — Monte Carlo simulations

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](#).