The Stata Journal (2007)
7, Number 4, pp. 587–589

Stata tip 54: Post your results

Philippe Van Kerm
CEPS/INSTEAD
Differdange, Luxembourg
philippe.vankerm@ceps.lu

The command `post` and its companion commands `postfile` and `postclose` are described in [P] **postfile** as "utilities to assist Stata programmers in performing Monte Carlo type experiments". That description understates their usefulness, as `post` is one of the most flexible ways to accumulate results and save them for later use in an external file.

Stata output is displayed in the Results window and can be stored in log files. However, browsing log files and selecting particular results can be tedious and inefficient. Fortunately, there are several alternatives, including the use of `file` (see [P] **file**) or the `estimates` suite of commands (see [R] **estimates**), and `post`, the focus here.

Use of `post` is fully described in [P] **postfile**. The steps are in essence:

1. Call `postfile` to initialize the results file: identify the filename, name its variables, and determine their types.
2. Run the analysis and accumulate the results by repeatedly calling `post`. Each call to `post` adds one observation (record or line) to the results file.
3. Close the results file with `postclose`.

`post` is flexible in what it records: e-class, r-class, or s-class results, string or numeric values, locals, constants, etc. Posted results are recorded without disturbing the data in memory. This is particularly neat: it keeps datasets tidy and allows calling multiple files without interfering with the accumulation of results.

This first example uses the `auto` data. We loop over all possible combinations of `foreign` and `rep78` and save average `price` within each group. Estimates are recorded in a new file named `autoinfo.dta`, which is later opened for displaying results with `tabdisp`.

```
. tempname hdle
. postfile `hdle´ foreign rep78 mean using autoinfo
. sysuse auto
(1978 Automobile Data)
. forvalues f=0/1 {
  2.         forvalues r=1/5 {
  3.                 summarize price if foreign==`f´ & rep78==`r´, meanonly
  4.                 post `hdle´ (`f´) (`r´) (r(mean))
  5.         }
  6. }
. postclose `hdle´
```



```
. use autoinfo, clear
. label define lf 0 "Domestic car" 1 "Foreign car"
. label values foreign lf
. label variable foreign "Origin of car"
. label variable rep78 "1978 repair record"
. tabdisp rep78 foreign, cell(mean)
```

1978 repair record	Origin of car: Domestic car	Origin of car: Foreign car
1	4564.5	
2	5967.625	
3	6607.074	4828.667
4	5881.556	6261.444
5	4204.5	6292.667

This example just shows the technique. In fact, for similar problems, the same effect can be produced easily with `statsby` (see [D] **statsby**):

```
. sysuse auto
(1978 Automobile Data)
. statsby mean=r(mean), by(foreign rep78) saving(autoinfo2): summarize price
(running summarize on estimation sample)
  (output omitted)
. use autoinfo2
(statsby: summarize)
. tabdisp rep78 foreign, cell(mean)
  (output omitted)
```

However, `statsby` is too restricted for more elaborate problems. A second example shows computations that store results for each of a series of files, here the numbers of observations and variables. It also demonstrates that graph commands are easily used for displaying results.

```
. tempname hdle
. postfile `hdle´ str20 name str100 label nobs nvar using sysfilesinfo
. sysuse dir
  (output omitted)
. local allfiles "`r(files)´"
. foreach dtafile of local allfiles {
  2.         sysuse `dtafile´, clear
  3.         describe, short
  4.         post `hdle´ ("`dtafile´") (`"`: data label´"´) (r(N)) (r(k))
  5. }
  (output omitted)
. postclose `hdle´
. use sysfilesinfo
. keep if label!=""
(18 observations deleted)
```

```
. replace name = subinstr(name,".dta","",.)
(15 real changes made)
. label variable nobs "Number of observations in dataset"
. label variable nvar "Number of variables in dataset"
. scatter nvar nobs if nobs<250, mlabel(name) mlabpostion(12)
```

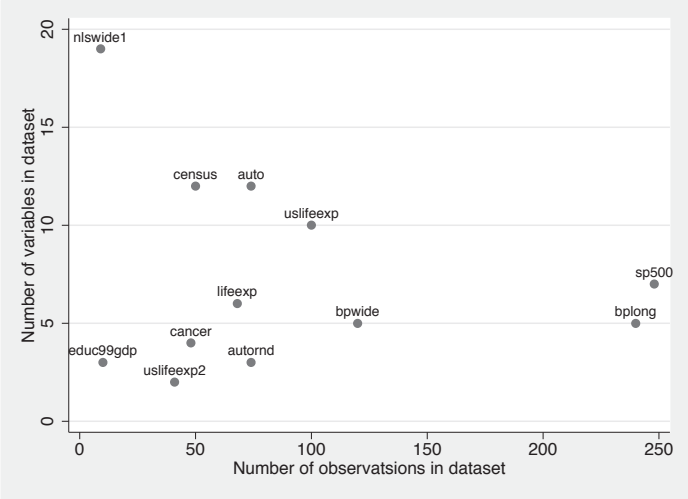

Figure 1: Exploiting posted results

The flexibility of `post` for both collection, when relevant results are posted, and processing, when collected results are analyzed, makes it useful in a broad range of settings, which is different from Monte Carlo simulations.