The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*), *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$115	1-year subscription	$145
2-year subscription	$210	2-year subscription	$270
3-year subscription	$285	3-year subscription	$375
1-year student subscription	$ 85	1-year student subscription	$115
1-year institutional subscription	$345	1-year institutional subscription	$375
2-year institutional subscription	$625	2-year institutional subscription	$685
3-year institutional subscription	$875	3-year institutional subscription	$965
Electronic only		**Electronic only**	
1-year subscription	$ 85	1-year subscription	$ 85
2-year subscription	$155	2-year subscription	$155
3-year subscription	$215	3-year subscription	$215
1-year student subscription	$ 55	1-year student subscription	$ 55

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.


The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, STATA, Stata Press, Mata, MATA, and NetCourse are registered trademarks of StataCorp LP.

The Stata Journal (2015)
15, Number 2, pp. 597–598

Stata tip 124: Passing temporary variables to subprograms

Maarten L. Buis
Department of History and Sociology
University of Konstanz
Konstanz, Germany
maarten.buis@uni-konstanz.de

A useful tool when programming in Stata is the temporary variable, which can be created using the `tempvar` command (see [P] **macro**). When it is convenient to store intermediate steps in a temporary variable, `tempvar` reserves a variable name for that temporary variable that is guaranteed not to exist in your current dataset. This ensures that your program will not accidentally overwrite an already existing variable. `tempvar` also ensures that the temporary variable is removed once the program that created it is finished so that your program will not clutter the user's dataset with unwanted intermediate results. Similarly, one can create temporary scalars and matrices with the `tempname` command (see [P] **macro**). When one programs in Stata, it is also useful to break up larger programs into various smaller subroutines. This helps to keep longer programs organized and makes it easier to write, debug, certify, and maintain them. Sometimes, creating temporary results in a temporary variable is a good candidate for such a subroutine. If we use `tempvar` or `tempname` in that subroutine, the temporary variable, scalar, or matrix will be deleted as soon as the subroutine is finished. In this case, that is not what we want.

To use the temporary objects created or changed in subroutines in the main program, we need to use `tempvar` or `tempname` in the main program and pass that name to the subroutine. Consider the example below.

```
. set seed 1234567
. program mainprog
  1.         tempvar random
  2.         quietly generate `random´ = .
  3.         tempname mean
  4.         scalar `mean´ = 2
  5.         subprog, random(`random´) mean(`mean´)
  6.         summarize `random´
  7. end
. program subprog
  1.         syntax, random(name) mean(name)
  2.         quietly replace `random´ = rnormal(`mean´)
  3. end
. sysuse auto
(1978 Automobile Data)
. mainprog
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    __000000 |         74     1.87675    1.013603  -.4361137   4.350792
```


In line 1 of `mainprog`, a variable name is chosen that does not exist in the current data, and this variable name is stored in the local macro `‘random’`. In line 2, this name is used to create a variable. In lines 3 and 4, a temporary scalar `‘mean’` is created. In line 5, the names of the temporary variable and the temporary scalar are passed to `subprog` in the options `random()` and `mean()`. Notice that `subprog` runs when `mainprog` is not yet finished, so variables created with `tempvar` and matrices and scalars created with `tempname` still exist. Line 1 of `subprog` means that `subprog` expects two options containing a name, and that name will be stored in the local macros `‘random’` and `‘mean’`. Line 2 of `subprog` then changes the temporary variable by using the temporary scalar. Now, we go back to line 6 of `mainprog`, which uses that changed temporary variable. `mainprog` ends, and the temporary variable `‘random’` and temporary scalar `‘mean’` are deleted.

The same logic can also be used to pass temporary variables, matrices, and scalars to Mata functions; as long as the program that created them has not finished, the objects exist. To pass them on, you must pass their names to the Mata function. For example, the program below does the same thing as the example above, except that it uses Mata for the subroutine.

```
. clear all
. mata
------------------------------------------------- mata (type end to exit) -------
: void mata_subprog(
>     string scalar randomname,
>     string scalar meanname) {
>
>     st_view(random=., ., randomname)
>     mean = st_numscalar(meanname)
>
>     random[.,.] = rnormal(st_nobs(),1,mean,1)
> }
: end
--------------------------------------------------------------------------------

. program mainprog
  1.         tempvar random
  2.         quietly generate `random´ = .
  3.         tempname mean
  4.         scalar `mean´ = 2
  5.         mata: mata_subprog("`random´", "`mean´")
  6.         summarize `random´
  7. end
. sysuse auto
(1978 Automobile Data)
. mainprog
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
    __000000 |         74    1.904863     .9529203  -.5836316   3.828233
```