THE STATA JOURNAL

Editor
H. Joseph Newton
Department of Statistics
Texas A&M University
College Station, Texas 77843
979-845-8817; fax 979-845-6077
jnewton@stata-journal.com

Editor
Nicholas J. Cox
Department of Geography
Durham University
South Road
Durham DH1 3LE UK
n.j.cox@stata-journal.com

Associate Editors
Christopher F. Baum
Boston College

Nathaniel Beck
New York University

Rino Bellocchio
Karolinska Institutet, Sweden, and
University of Milano-Bicocca, Italy

Maarten L. Buis
Tübingen University, Germany

A. Colin Cameron
University of California–Davis

Mario A. Cleves
Univ. of Arkansas for Medical Sciences

William D. Dupont
Vanderbilt University

David Epstein
Columbia University

Allan Gregory
Queen’s University

James Hardin
University of South Carolina

Ben Jann
University of Bern, Switzerland

Stephen Jenkins
University of Essex

Ulrich Kohler
WZB, Berlin

Frauke Kreuter
University of Maryland–College Park

Peter A. Lachenbruch
Oregon State University

Jens Lauritsen
Odense University Hospital

Stanley Lemeshow
Ohio State University

J. Scott Long
Indiana University

Roger Newson
Imperial College, London

Austin Nichols
Urban Institute, Washington, DC

Marcello Pagano
Harvard School of Public Health

Sophia Rabe-Hesketh
University of California–Berkeley

J. Patrick Royston
MRC Clinical Trials Unit, London

Philip Ryan
University of Adelaide

Mark E. Schaffer
Heriot-Watt University, Edinburgh

Jeroen Weesie
Utrecht University

Nicholas J. G. Winter
University of Virginia

Jeffrey Wooldridge
Michigan State University

Stata Press Editorial Manager
Deirdre Patterson and Erin Roberson
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.

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

http://www.stata-journal.com

The *Stata Journal* is indexed and abstracted in the following:

- CompuMath Citation Index®
- Current Contents/Social and Behavioral Sciences®
- RePEc: Research Papers in Economics
- Science Citation Index Expanded (also known as SciSearch®)
- Scopus™
- Social Sciences Citation Index®

**Copyright Statement:** The *Stata Journal* and the contents of the supporting files (programs, datasets, and help files) are copyright © by StataCorp LP. The contents of the supporting files (programs, datasets, and help files) may be copied or reproduced by any means whatsoever, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the *Stata Journal*.

The articles appearing in the *Stata Journal* may be copied or reproduced as printed copies, in whole or in part, as long as any copy or reproduction includes attribution to both (1) the author and (2) the *Stata Journal*.

Written permission must be obtained from StataCorp if you wish to make electronic copies of the insertions. This precludes placing electronic copies of the *Stata Journal*, in whole or in part, on publicly accessible websites, file servers, or other locations where the copy may be accessed by anyone other than the subscriber.

Users of any of the software, ideas, data, or other materials published in the *Stata Journal* or the supporting files understand that such use is made without warranty of any kind, by either the *Stata Journal*, the author, or StataCorp. In particular, there is no warranty of fitness of purpose or merchantability, nor for special, incidental, or consequential damages such as loss of profits. The purpose of the *Stata Journal* is to promote free communication among Stata users.

The *Stata Journal* (ISSN 1536-867X) is a publication of Stata Press. Stata, Mata, NetCourse, and Stata Press are registered trademarks of StataCorp LP.
Stata tip 92: Manual implementation of permutations and bootstraps

Lars Ångquist
Institute of Preventive Medicine
Copenhagen University Hospitals
Copenhagen, Denmark
la@ipm.regionh.dk

In mathematics, a permutation might be seen as a reordering of an ordered set of abstract elements (see, for example, [Fraleigh 2002]), whereas in data analysis—when facing empirical data—this concept may correspond to a reordering of an ordered set of observations. Vaguely speaking, in statistics and significance testing, this might be an interesting concept when simulating under a null hypothesis corresponding to, in some sense, a null association or an effect (most often an outcome) of one specific variable with respect to another one. Here one basically keeps the dataset constant except for the values, which are instead randomly permuted, corresponding to the core variable. Because all permutations are generally equally likely (at least if properly dealing with potential confounding) under the null hypothesis of no association, this is a way, through such simulations, of estimating the corresponding null distribution underlying, for instance, related p-values.

For similar reasons, one may apply the bootstrap simulation procedure. Here one does not reorder observations (or in general elements) but rather simulates from the empirical distribution based on this very set. In simulation terminology, the bootstrap and permutation procedures in this sense correspond to a uniformly random selection of values from the empirical distribution with and without replacement, respectively. For more information, see, for example, [Manly 2007] for permutations, [Davison and Hinkley 1997] for bootstraps, and [Robert and Casella 2004] for stochastic simulation, in general.

In Stata, one may—given some assumed framework—use the commands `permute` and `bootstrap` to perform tasks related to permutation-based and bootstrap-based significance tests, respectively. Sometimes however, whether it arises as a need to be more specific or because one simply wants to keep more detailed control over the actual data manipulations, it might be favorable to perform some related manual labor at your computer keyboard. This tip is about the general structure of a solution for such a task.

**Permuting:** Assume that you have a variable of interest, `permvar`, that you want to permute in the sense noted above. Typing

1. The set of all possible reorderings (permutations) includes the permutation that actually leaves the order intact. This is called the identity permutation.
in Stata will give you an additional column (upermvar) of permuted values. In the first command, a new variable, id, that corresponds to the current sort order is created. In the second command, a column is generated with values uniformly distributed between 0 and 1. Because the values of u were randomly generated, sorting on u puts the observations in a random order. The next command saves the variable type of permvar in the local macro type so that the type can be applied to the new variable in the last command. The last command stores the permutation in the new variable upermvar: each new value is a value of permvar from a randomly selected observation. (The random selection is controlled by the id variable, which was put in a random order by the sort command.)

To reduce the risk of tied values with respect to the (inherently discrete) random draws, and moreover to further increase the, so to speak, randomness of the derived values, one might replace the code lines 2–3 with the following:

```
... generate double u1=runiform() generate double u2=runiform() sort u1 u2 ...
```

The randomness reference corresponds to the fact that computer-generated random numbers are random only to the extent permitted by the implementation of what is termed pseudorandom numbers (see, for instance, Knuth [1998]). To achieve reproducible results, one might take advantage of this pseudorandomness by explicitly stating a starting point, that is, a seed value, for the deterministic algorithm:

```
set seed 760130
```

The number must be a positive integer. For instance, this command might be used when assuring that different methods give equivalent results or, for example, with respect to estimated variances of certain derived estimates of interest, when comparing methods with respect to efficiency performance.

---

2. In other words, this construction is based on the observation number indicator n, which equals 1, 2, . . . , N through the present observations (rows), where N is the number of observations in the dataset (generally reachable in a similar fashion through N in Stata). Moreover, one approach to retaining a sort order, irrespective of the content of an executed program, is by taking advantage of the sortpreserve option (see help program or Newson [2002]).

3. You might use your personal birthdate as an easily remembered seed value. This is in fact used in the above case, though I am not revealing which date format I used; see help dates and times. I thank Claus Holst for this tip!
Bootstrapping: A related but slightly different variant of the above schedule might be used to derive a bootstrapped variable called `ubootsvar`. It is based on the empirical distribution formed or constituted by the present observations of the original variable `bootsvar`.

```stata
generate u=ceil(runiform()*_N)
generate ubootsvar=bootsvar[u]
```

Here the uniformly distributed values are not used to decide on a sort order (the underlying index values), but rather to directly constitute index values by making them be part of a uniformly distributed simulation of values on the integers 1, 2, ..., N. To achieve this, the so-called ceiling function, `ceil()`, is used. For more information on `runiform()`, see help runiform or Buis (2007) (with respect to its use for simulations); further, `ceil()` and the related `floor()` function are described in Cox (2003).

Moreover, one might implement the above code structures into loops based on, for instance, `foreach` or `forvalues`. Under such circumstances, one might also take advantage of both usage of temporary variables (see help tempvar) and the specific matrix-oriented environment of Mata (see help mata) though the general structure described here might to some extent serve as a guideline or a template for such cases, as well. Once ready, strap your boots and let the permutation begin!

References


---

4. The `uniform()` function was improved in Stata 11 and was renamed `runiform()`. 