The Stata Journal

Editor
H. Joseph Newton
Department of Statistics
Texas A & M University
College Station, Texas 77843
979-845-3142; FAX 979-845-3144
jnewton@stata-journal.com

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

Associate Editors

Christopher Baum
Boston College

Rino Bellocco
Karolinska Institutet

David Clayton
Cambridge Inst. for Medical Research

Mario A. Cleves
Univ. of Arkansas for Medical Sciences

William D. Dupont
Vanderbilt University

Charles Franklin
University of Wisconsin, Madison

Joanne M. Garrett
University of North Carolina

Allan Gregory
Queen's University

James Hardin
University of South Carolina

Stephen Jenkins
University of Essex

Ulrich Kohler
WZB, Berlin

Jens Lauritsen
Odense University Hospital

Stanley Lemeshow
Ohio State University

J. Scott Long
Indiana University

Thomas Lumley
University of Washington, Seattle

Roger Newson
King's College, London

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
Cornell University

Jeffrey Wooldridge
Michigan State University

Stata Press Production Manager Lisa Gilmore

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 web sites, fileservers, 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*, electronic version (ISSN 1536-8734) is a publication of Stata Press, and Stata is a registered trademark of StataCorp LP.

The Stata Journal (2005)
5, Number 1, pp. 135–136

Stata tip 17: Filling in the gaps

Nicholas J. Cox
University of Durham, UK
n.j.cox@durham.ac.uk

The `fillin` command (see [R] **fillin**) does precisely one thing: it fills in the gaps in a rectangular data structure. That is very well explained in the manual entry, but people who do not yet know the command often miss it, so here is one more plug. Suppose that you have a dataset of people and the choices they make, something like this:

```
id   choice
1     1
2     3
3     1
4     2
```

Now suppose that you wish to run a nested logit model using `nlogit` (see [R] **nlogit**). This command requires all choices, those made and those not made, to be explicit. With even 4 values of `id` and 3 values of `choice`, we need 12 observations so that each combination of variables exists once in the dataset; hence, 8 more are needed in this case. The solution is just

```
. fillin id choice
```

and a new variable, `_fillin`, is added to the dataset with values 1 if the observation was "filled in" and 0 otherwise. Thus `count if _fillin` tells you how many observations were added. You will often want to `replace` or `rename _fillin` to something appropriate:

```
. rename _fillin chosen
. replace chosen = 1 - chosen
```

If you do not `rename` or `drop _fillin`, it will get in the way of a subsequent `fillin`. Usually, the decision is clear-cut: Either `_fillin` has a natural interpretation, so you want to keep it, or a relative, under a different name; or `_fillin` was just a by-product, and you can get rid of it without distress.

Another common variant is to show zero counts or amounts explicitly. With a dataset of political donations for several years, we might want an observation showing that `amount` is zero for each pair of `donor` and `year` not matched by a donation. This typically leads to neater tables and graphs and may be needed for modeling: in particular, for panel models, the zeros must be present as observations. The main idea is the same, but the aftermath is different:

```
. fillin donor year
. replace amount = 0 if _fillin
```

© 2005 StataCorp LP

Naturally if we have more than one donation from various donors in various years, we might also want to `collapse` (or just possibly `contract`) the data, but that is the opposite kind of problem.

Yet another common variant is the creation of a grid for some purpose, perhaps before data entry, or before you draw a graph. You can be very lazy by typing

```
. clear
. set obs 20
. gen y = _n
. gen x = y
. fillin y x
```

which creates a 20×20 grid. This is good, but sometimes you want something different; see functions `fill()` and `seq()` in [R] **egen**.

The messiest `fillin` problems are when some of the categories you want are not present in the dataset at all. If you know a person is not one of the values of `donor`, no amount of filling in will add a set of zeros for that person. One strategy here is to add pseudo-observations so that every category occurs at least once and then to `fillin` in terms of that larger dataset. This is just a variation on the technique for creating a grid out of nothing.

As far as you can see from what is here, `fillin` just does things in place, so you need not worry about file manipulation. This is an illusion, as underneath the surface, `fillin` is firing up `cross` (see [R] **cross**), which does the work using files. Thus `cross` is more fundamental. A forthcoming tip will say more.