

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
jnnewton@stata-journal.com

Associate Editors

Christopher F. Baum
Boston College

Rino Bellocco
Karolinska Institutet, Sweden and
Univ. degli Studi di Milano-Bicocca, Italy

A. Colin Cameron
University of California–Davis

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

Allan Gregory
Queen's University

James Hardin
University of South Carolina

Ben Jann
ETH Zürich, Switzerland

Stephen Jenkins
University of Essex

Ulrich Kohler
WZB, Berlin

Jens Lauritsen
Odense University Hospital

Stata Press Production Manager

Stata Press Copy Editor

Editor

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

Stanley Lemeshow
Ohio State University

J. Scott Long
Indiana University

Thomas Lumley
University of Washington–Seattle

Roger Newson
Imperial 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
University of Virginia

Jeffrey Wooldridge
Michigan State University

Lisa Gilmore

Deirdre Patterson

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, 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*, electronic version (ISSN 1536-8734) is a publication of Stata Press. Stata and Mata are registered trademarks of StataCorp LP.

Stata tip 52: Generating composite categorical variables

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

If you have two or more categorical variables, you may want to create one composite categorical variable that can take on all the possible joint values. The canonical example for Stata users is given by cross-combinations of `foreign` and `rep78` in the `auto` data. Setting aside missings, `foreign` takes on values of 0 and 1, and `rep78` takes on values of 1, 2, 3, 4, and 5. Hence there are ten possible joint values, which could be 0 and 1, 0 and 2, and so forth. As it happens, only eight occur in the data. If we add the value labels attached to `foreign`, we have Domestic 1, Domestic 2, and so forth.

Writing the values like that raises the question of whether these cross-combinations will be better expressed as string variables or as numeric variables with value labels. On the whole, an integer-valued numeric variable with value labels defined and attached is the best arrangement for any categorical variable, but a string variable may also be convenient, especially if you are producing a kind of composite identifier.

A method often seen is to produce string variables with `tostring` (see [D] `destring`), for example,

```
. tostring foreign rep78, generate(Foreign Rep78)
. gen both = Foreign + Rep78
```

Naturally, there are endless minor variations on this method. A small but useful improvement is to insert a space or other punctuation:

```
. gen both = Foreign + " " + Rep78
```

However, this method is not especially good. `tostring` is really for correcting mistakes, whether attributable to human fault or to some software used before you entered Stata: some variable that should be string is in fact numeric. You need to correct that mistake. `tostring` is a safe way of doing that.

That intended purpose does not stop `tostring` being useful for things for which it was not intended, but there are two specific disadvantages to this method:

1. This method needs two lines, and you can do it in one. That is a little deal.
2. This method could lose information, especially for variables with value labels or with noninteger values. That is, potentially, a big deal.

The second point may suggest using `decode` instead, but my suggestions differ. A better method is to use `egen, group()`. See [D] `egen`.

```
. egen both = group(foreign rep78), label
```

This command produces a new numeric variable, with integer values 1 and above, and value labels defined and attached. Particularly, note the `label` option, which is frequently overlooked.

This method has several advantages:

1. One line.
2. No loss of information. Observations that are identical on the arguments are identical on the results. Value labels are used, not ignored. Distinct noninteger values will also remain distinct.
3. The label is useful—indeed essential—for tables and graphs to make sense.
4. Efficient storage.
5. Extends readily to three or more variables.

Another fairly good method is to use `egen, concat()`.

```
. egen both = concat(foreign rep78), decode p(" ")
```

This command creates a string variable, so it is less efficient for data storage and is less versatile for graphics or modeling. Compared with `tostring`, the advantages are

1. One line.
2. You can mix numeric and string arguments. `concat()` will calculate what is needed.
3. You can use the `decode` option to use value labels on the fly.
4. You can specify punctuation as separator, here a blank.
5. Extends to three or more variables.

Stata tip 55: Better axis labeling for time points and time intervals

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

Plots of time-series data show time on one axis, usually the horizontal or x axis. Unless the number of time points is small, axis labels are usually given only for selected times. Users quickly find that Stata's default time axis labels are often not suitable for use in public. In fact, the most suitable labels may not correspond to *any* of the data points. This will arise when it is better to label longer time intervals, rather than any individual times in the dataset.

For example,

```
. webuse turksales
```

reads in 40 quarterly observations for 1990q1 to 1999q4 with a response variable of turkey sales. The default time axis labels with both `line sales t` and `tsline sales` are 1990q1, 1992q3, 1995q1, 1997q3, and 2000q1. These are not good choices for any purpose, even exploration of the data in private.

Label choice is partly a matter of taste, but you might well agree with Stata that labeling every time point would be busy and the result difficult to read. With 40 quarterly values, possible choices include one point per year (10 labels) and one point every other year (5 labels). One possibility is to label every fourth quarter, as that is usually the quarter with highest turkey sales. `summarize` reveals that the times range from 120 to 159 quarters (0 means the first quarter of 1960), so we can type

```
. line sales t, xlabel(123(4)159)
```

Note how we use a *numlist*, `123(4)159`, to avoid spelling out every value. The step length is 4 for four quarters. See [U] **11.1.8 numlist** or `help numlist` for more details of *numlists*. This graph too would need more work before publication, as the labels are still crowded. The text of the labels (e.g., 1990q4) may or may not be judged suitable, depending partly on the readership for the graph.

However, there is another choice: label time intervals (years) and mark the boundaries between those time intervals by ticks. Consider 1990. The four quarters in Stata's units are 120, 121, 122, and 123. Thus we could put text showing the year at a midpoint of 121.5 and ticks showing year boundaries at 119.5 and 123.5. For all years, we should use the *numlist* idea again with the following command to produce figure 1.

```
. line sales t, xtick(119.5(4)159.5, tlength(*1.5))  
> xlabel(121.5(4)157.5, noticks format(%tqCY)) xtitle("")
```

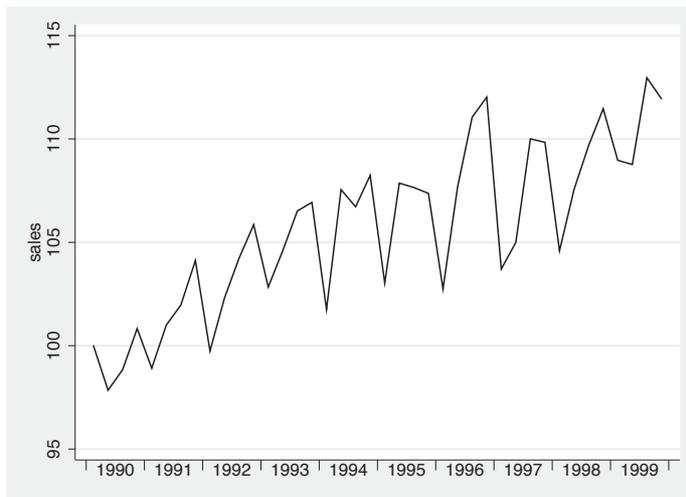


Figure 1: Turkey sales in each quarter. Time axis labels show years (with ticks suppressed) and time axis ticks show year ends.

The most important details here are suppressing the ticks for the axis labels and specifying a format for them. Cosmetic additions include lengthening the ticks compared with the default and suppressing the axis title, which would otherwise be the variable name `t` (or a variable label if it existed). It is usually clear from the labels what is being shown. Other possibilities include changing the text size for the axis label, changing the angle at which the axis label is shown, and suppressing the century by using a format like `%tqY`. Those may not be especially attractive, but nevertheless might be forced upon you by practicalities.

The main idea is clearly more general. The axis labels and the axis ticks need not correspond to each other, and it might be good to have fewer labels than ticks for longer series. Monthly and half-yearly data naturally yield to the same method, but use 12 or 2 and not 4 as the step length. Weekly and daily data are more awkward but still manageable.

If you were producing many similar graphs, you might want to automate this process to some degree. The mental arithmetic might easily be more challenging than in the turkey example. Let us imagine daily data for several years. Thus we could put ticks every January 1 and year labels every July 1. That will be adequate precision in practice. Find the first and last years in your data, if necessary by a command like `gen year = year(date)` followed by `summarize`. Suppose again that the years are 1990–1999. We can put the needed dates in local macros with a loop:

```
. forvalues y = 1990/1999 {
    local jan `jan' `=mdy(1,1,`y')'
    local jul `jul' `=mdy(7,1,`y')'
}
```

Each time around the loop the daily dates for January 1 and July 1 in each year are calculated on the fly with a call to the `mdy()` function and added to a macro. For more details, see [P] **forvalues** and [P] **macro**, the corresponding help files, or Cox (2002). Once done, the graph command is something like

```
. line whatever date, xlabel(`jul`, format(%tdCY) noticks)
> xtick(`jan`, tlength(*1.5))
```

A key requirement is that the local macros used in the graph command must be visible, by virtue of being in the same interactive session, do-file, or program. That is in essence what `local` means.

Calendar years, meaning here Western calendar years, are clearly not the only possibilities. You could use other boundaries and midpoints for years or other periods defined by other criteria (e.g., academic, financial, fiscal, hydrological, political, religious).

Reference

Cox, N. J. 2002. Speaking Stata: How to face lists with fortitude. *Stata Journal* 2: 202–222.

Software Updates

gr0012.1: Density probability plots. N. J. Cox. *Stata Journal* 5: 259–273.

The program has been updated so that users of Stata 9 and later can use an `addplot()` option.

st0133.1: Fitting mixed logit models by using maximum simulated likelihood. A. R. Hole. *Stata Journal* 7: 388–401.

New features include options for specifying weights (including sampling weights) and for obtaining robust and cluster-robust standard errors. The estimation speed has also been improved by using analytical instead of numerical derivatives when maximizing the simulated log-likelihood function. This change has the side effect of producing somewhat different estimation results compared with the previous version for some datasets and model specifications. The new `numerical` option may be used to replicate estimation results produced with the old version, but it should only be used for that purpose, as it causes the command to run slowly.