The Stata Journal (2008)
8, Number 1, pp. 142–145

Stata tip 59: Plotting on any transformed scale

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

Using a transformed scale on one or the other axis of a plot is a standard graphical technique throughout science. The most common example is the use of a logarithmic scale. This possibility is wired into Stata through options `yscale(log)` and `xscale(log)`; see [G] ***axis_scale_options***. The only small difficulty is that Stata is not especially smart at reading your mind to discern what axis labels you want. When values range over several orders of magnitude, selected powers of 10 are likely to be convenient. When values range over a shorter interval, labels based on multiples of 1 2 5 10, 1 4 7 10, or 1 3 10 may all be good choices.

No other scale receives such special treatment in Stata. However, other transformations such as square roots (especially for counts) or reciprocals (e.g., in chemistry or biochemistry [Cornish-Bowden 2004]) are widely used in various kinds of plots. The aim of this tip is to show that plotting on *any* transformed scale is straightforward. As an example, we focus on logit scales for continuous proportions and percents.

Given proportions p, logit $p = \ln\{p/(1-p)\}$ is perhaps most familiar to many readers as a link function for binary response variables within logit modeling. Such logit modeling is now over 60 years old, but before that lies a century over which so-called logistic curves were used to model growth or decay in demography, ecology, physiology, chemistry, and other fields. Banks (1994), Kingsland (1995), and Cramer (2004) give historical details, many examples, and further references.

The growth of literacy and its complement—the decline of illiteracy—provide substantial examples. In a splendid monograph, Cipolla (1969) gives fascinating historical data but no graphs. Complete illiteracy and complete literacy provide asymptotes to any growth or decay curve, so even without any formal modeling we would broadly expect something like S-shaped or sigmoid curves. Logit scales in particular thus appear natural or at least convenient for plotting literacy data (Sopher 1974, 1979). More generally, plotting on logit scales goes back at least as far as Wilson (1925).

Figure 1 shows how many newly married people could not write their names in various countries during the late nineteenth century, as obtained with data from Cipolla (1969, 121–125) and the following commands:

```
. local yti "% newly married unable to write their names"
. line Italy_females Italy_males France_females France_males Scotland_females
> Scotland_males year, legend(pos(3) col(1) size(*0.8)) xla(1860(10)1900)
> xtitle("") yla(, ang(h)) ytitle(`yti´)
```

© 2008 StataCorp LP

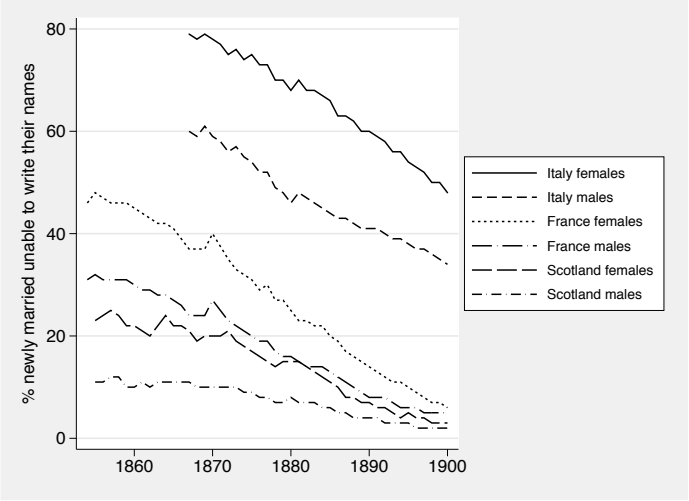

Figure 1. Line plot of illiteracy by sex for various countries in the nineteenth century.

How do we show such data on a logit scale? There are two steps. First, calculate the coordinates you want shown before you graph them. Here we loop over a bunch of variables and apply the transform `logit(`*percent*`/100)`:

```
. foreach v of var *males {
.       gen logit_`v´ = logit(`v´/100)
.       label var logit_`v´ "`: var label `v´´"
. }
```

For tutorials on `foreach` and the machinery used here in looping, see Cox (2002, 2003). Note that, at the same time, we copy variable labels across so that they will show up automatically on later graph legends.

Second, and just slightly more difficult, is to get axis labels as we want them (and axis ticks also, if needed). Even people who work with logits all the time usually do not want to decode that a logit of 0 means 50%, or a logit of 1 means 73.1%, and so forth, even if the `invlogit()` function makes the calculation easy. Logit scales stretch percents near 0 or 100 compared with those near 50. Inspection of figure 1 suggests that `2 5 10(10)80` would be good labels to show for percents within the range of the data. So we want text like `50` to be shown where the graph is showing `logit(50/100)`. The key trick is to pack all the text we want to show and where that text should go into a local macro.

```
. foreach n of num 2 5 10(10)80 {
.       local label `label´ `= logit(`n´/100)´ "`n´"
. }
```

To see what is happening, follow the loop: First time around, local macro `'n'` takes on the value 2. `logit(2/100)` is evaluated on the fly (the result is about −3.8918) and that is where on our y axis the text "2" should go. Second time around, the same is done for 5 and `logit(5/100)`. And so forth over the numlist `2 5 10(10)80`.

Now we can get our graph with logit scale:

```
. line logit_Italy_females logit_Italy_males logit_France_females
> logit_France_males logit_Scotland_females logit_Scotland_males year,
> legend(pos(3) col(1) size(*0.8)) xla(1860(10)1900) xtitle("")
> yla(`label´, ang(h)) ytitle(`yti´)
```

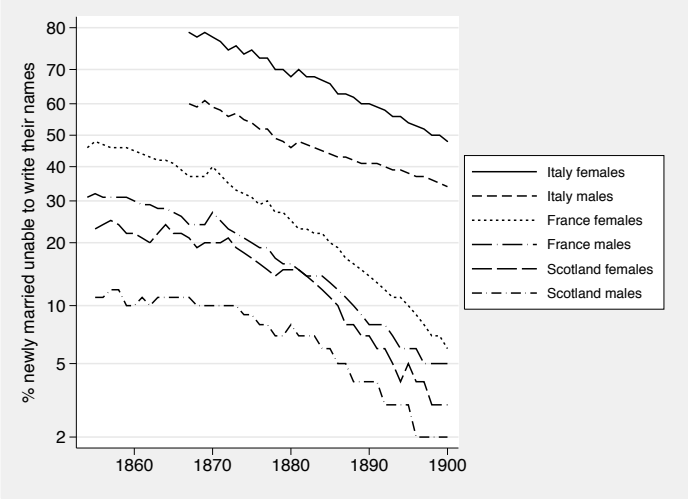

Figure 2. Line plot of illiteracy by sex for various countries in the nineteenth century. Note the logit scale for the response.

Specifically in this example, we now can see data on a more natural scale, complementing the original raw scale. The granularity of the data (rounded to integer percents) is also evident.

Generally, some small details of macro handling deserve flagging. You may be accustomed to a tidy form of local macro definition:

```
. local macname "contents"
```

But the delimiters `" "` used here would, for this problem, complicate processing given that we do want double quotes inside the macro. Note from the previous `foreach` loop that they can be left off, to advantage.

In practice, you might need to iterate over several possible sets of labels before you get the graph you most like. Repeating the whole of the `foreach` loop would mean that the local macro would continue to accumulate material. Blanking the macro out with

```
. local label
```

will let you start from scratch.

The recipe for ticks is even easier. Suppose we want ticks at `15(10)75`, that is, at `15 25 35 45 55 65 75`. We just need to be able to tell Stata exactly where to put them:

```
. foreach n of num 15(10)75 {
.       local ticks `ticks´ `= logit(`n´/100)´
}
```

Then specify an option such as `yticks(‘ticks’)` in the graph command.

Finally, note that the local macros you define must be visible to the graph command you issue, namely within the same interactive session, do-file, or program. That is what local means, after all.

In a nutshell: Showing data on any transformed scale is a matter of doing the transformation in advance, after which you need only to fix axis labels or ticks. The latter is best achieved by building graph option arguments in a local macro.

References

Banks, R. B. 1994. *Growth and Diffusion Phenomena: Mathematical Frameworks and Applications*. Berlin: Springer.

Cipolla, C. M. 1969. *Literacy and Development in the West*. Harmondsworth: Penguin.

Cornish-Bowden, A. 2004. *Fundamentals of Enzyme Kinetics*. 3rd ed. London: Portland Press.

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

———. 2003. Speaking Stata: Problems with lists. *Stata Journal* 3: 185–202.

Cramer, J. S. 2004. The early origins of the logit model. *Studies in History and Philosophy of Biological and Biomedical Sciences* 35: 613–626.

Kingsland, S. E. 1995. *Modeling Nature: Episodes in the History of Population Ecology*. 2nd ed. Chicago: University of Chicago Press.

Sopher, D. E. 1974. A measure of disparity. *Professional Geographer* 26: 389–392.

———. 1979. Temporal disparity as a measure of change. *Professional Geographer* 31: 377–381.

Wilson, E. B. 1925. The logistic or autocatalytic grid. *Proceedings of the National Academy of Sciences* 11: 451–456.

The Stata Journal (2008)
8, Number 1, p. 146

Software Updates

st0126_1: QIC program and model selection in GEE analyses. J. Cui. *Stata Journal* 7: 209–220.

This program has been updated to include the general negative binomial distribution while only a special case of the negative binomial distribution was previously considered. The help file has also been updated to reflect this extension in example 4.

sxd1_3: Random allocation of treatments in blocks. P. Ryan. *Stata Technical Bulletin* 54: 49–53; 50: 36–37; 49: 43–46. Reprinted in *Stata Technical Bulletin Reprints*, vol. 9, pp. 353–358; vol. 9, pp. 352–353; vol. 7, pp. 297–300.

`ralloc` has been enhanced to support up to 10 (previously only 5) treatments in a one-way design; 4×4 two-way factorial designs; and $2 \times 2 \times 2$, $2 \times 2 \times 3$, $2 \times 3 \times 3$, and $3 \times 3 \times 3$ three-way factorial designs. The help file has three new examples showing how to implement three-way factorial designs and how to specify more complex treatment allocation ratios than is apparent from the simple `ratio()` option.

© 2008 StataCorp LP