



The World's Largest Open Access Agricultural & Applied Economics Digital Library

This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.

Help ensure our sustainability.

Give to AgEcon Search

AgEcon Search

<http://ageconsearch.umn.edu>

aesearch@umn.edu

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.

The Stata Journal (2016)
16, Number 3, pp. 805–812

Speaking Stata: Shading zones on time series and other plots

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

Abstract. Background shading of time series and other plots is a common need. For example, we often wish to identify particular periods of recession or war or other distinct conditions. The shading should be laid down before the data elements (for example, data points or lines). Several methods of shading can be effective. Tips are included for automating the production of a series of similar graphs.

Keywords: gr0067, shading, time series, line plots, scatterplots, twoway, graphics

1 Introduction

A common request on Statalist and in other forums featuring Stata questions is how to shade certain zones of a `twoway` plot distinctively. The runaway favorite example in my experience comes from economists or other social scientists who wish to shade periods of economic recession on time-series plots. More specifically, the U.S. National Bureau of Economic Research issues authoritative statements on peak and trough dates defining recessions. [Baum \(2006\)](#) published a helper Stata command to assist researchers using those definitions.

The same idea could be useful for other kinds of time series, and even more generally. We might wish to flag periods when certain parties or presidents were in power or office, or when certain policies or treatments were in effect. We might wish to emphasize periods of heavy rainfall or of intense seismic or volcanic activity in charting extreme environmental events and their aftermath. Without reference to time series at all, we might want to subdivide a quantitative range, say, on whether values are above or below a key threshold (for example, positive or negative; above or below subsistence level) or within certain intervals (for example, body mass index or systolic blood pressure).

Without solving the problem in its widest form (that way lies cartography: see, notably, [Pisati \[2004\]](#)), I gather some basic tips in this column. In essence, the graphic strategy is one of divide and conquer. You should use one subcommand of `twoway` to lay down shaded areas, and then plot the data on top. (Therein lies the first tip: some users have been puzzled when putting shading on top that some of their data points are no longer visible, so do the shading first!)

For related ideas you may find useful, see [Cox \(2009a,b\)](#).

2 The main idea

As a simple working example, we use data on crude birth and death rates for Scotland, part of the United Kingdom. The dataset illustrating this column extends from 1855 to 2014, but we focus on 1900–1960, which allows illustration of the world wars in 1914–1918 and 1939–1945 as periods to be shaded. The dataset (with the media for this column) also includes series for England and Wales and the United Kingdom as a whole.¹ For those unfamiliar with such data, the definition of (crude) birth or death rates is the number of births or deaths in a year per 1,000 population. Demographers or epidemiologists may want to make those definitions more precise.

As usual in the *Stata Journal*, the graph scheme used is `sj`, which uses no colors beyond grayscale varying from black to white. A little more will be said later about color.

The first step is to draw a graph with your basic data and think about how the shading is to be defined. Here, as small twists on the default, we insist on particular line patterns for the two series and suppress the legend in favor of text labels within the plot region.

```
. use uk_vital
. set scheme sj
. local linecall sc_births sc_deaths year if inrange(year, 1900, 1960),
> lpattern(dash solid) xscale(r(1900 1967)) legend(off)
> ytitle(birth and death rates/1000)
. local scattericall 19.6 1960 "births" 11.9 1960 "deaths", mlabsize(*1.2)
> msymbol(none)
. twoway line `linecall' || scatteri `scattericall'
```

Command chunks that will be reused several times were put into local macros. This is partly to save on typing and partly to make the overall structure of commands a little clearer. A pitfall for those unfamiliar with local macros is that they are indeed local. You can define local macros, for example, in a do-file or in a Do-file Editor window, or you can define them within your main session by typing commands into the Command window; but local macros defined in one place will not be visible in another. In practice, this limit typically gives the edge to developing a script in an editor window so that mistakes can be more quickly corrected, or the local macro definitions changed, as ideas or circumstances also change. If local macros are new to you, a good introduction can be found at [U] **18.3 Macros**.

Even with a fair amount of Stata experience, you should not expect to be able to write down commands for yourself without error and in exactly the right order, unless you happen to have just solved the same problem. This is a rational reconstruction of my code attempts in writing this column: details are mentioned in what now seems the logical order, and you do not see what lies strewn on the cutting-room floor.

1. The data were downloaded from <http://www.ons.gov.uk/peoplepopulationandcommunity/populationandmigration/populationestimates/datasets/vitalstatisticspopulationandhealthreferencetables> on 24 July 2016.

Figure 1 shows where we are as a first attempt. We already know that we want to shade 1914–1918 and 1939–1945, and we can see that the vertical limits will be, as indicated by axis labels, around 10 and 30. We will consider shortly what to do about the offsets where the axis goes a little below and a little above the labeled limits.

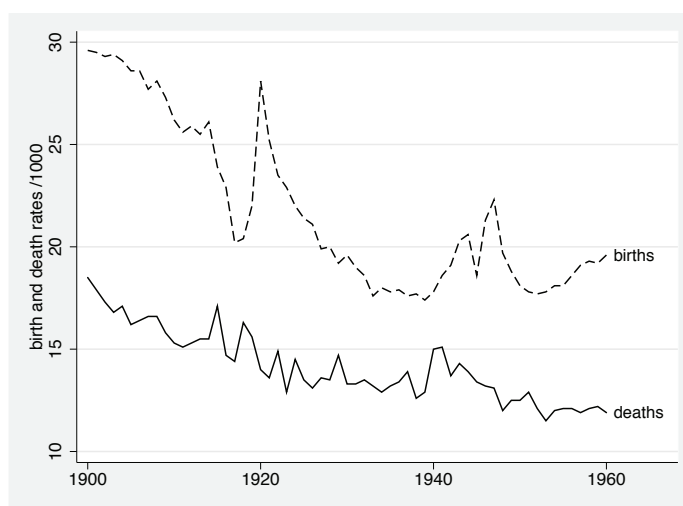


Figure 1. Crude birth and death rates for Scotland for 1900–1960

More readers will, I guess, be familiar with the `twoway line` command than with the `twoway scatteri` command. If the latter is new to you, consider checking out its help file or manual entry. `scatteri` can be useful for several different graphical tweaks, such as adding text (and not even showing the associated marker symbols), as done here. There was no reason for me not to use the `text()` option; it was just a matter of what I thought of first.

If you want to know more about `inrange()`, see [Cox \(2006\)](#).

The shading can be done in several ways. If there is a choice between them, it is largely one of whichever appears most convenient or congenial.

One way is to draw one or more bars as appropriate. For bars, we need to consider an upper limit, always; a lower limit, if it is not 0; and a width. The width is often easiest to determine. In `twoway bar`, the width defaults to 1 in whatever units are being used for the x -axis variable. Here, and often, that is exactly right, because the x variable is time in years, and the time step in the data is indeed 1 year. For other examples, choosing the bar width will need more care. If you had weekly data defined by days precisely 7 days apart, a width nearer 7 is likely to make more sense. (I say “nearer 7” rather than always 7 because it may seem a good idea to leave a small gap to flag the discreteness of time resolution.) If you had datetimes defined in milliseconds, a bar of width 1 millisecond might make no sense at all.

```
. generate upper = 30
. local barcall upper y if inrange(y, 1914, 1918) | inrange(y, 1939, 1945),
> bcolor(gs14) base(10)
. twoway bar `barcall' || line `linecall' || scatteri `scattericall'
```

For the bars, we put the upper limit into a variable and can specify the lower limit with the `base()` option. Figure 2 shows the result.

The key point here is that the shading is laid down first, then the data as shown by a line plot, and finally annotations, which are designed not to interfere with the data.

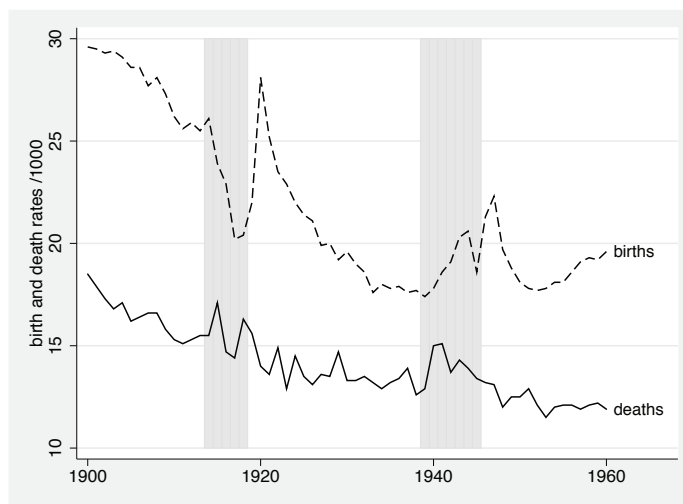


Figure 2. Crude birth and death rates for Scotland for 1900–1960; the periods of world wars 1914–1918 and 1939–1945 have been shaded distinctly

To remove the offset, we note that it is the default nonzero margin to the `plotregion`, which we can thus just set to 0. `help region options` would take you to documentation.

```
. twoway bar `barcall' || line `linecall' || scatteri `scattericall'
> plotregion(margin(zero))
```

Figure 3 shows the result.

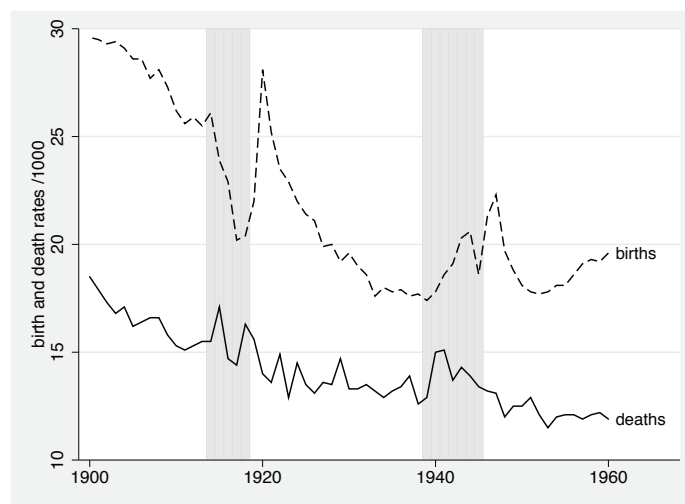


Figure 3. Crude birth and death rates for Scotland for 1900–1960; the periods of world wars 1914–1918 and 1939–1945 have been shaded distinctly. Compare carefully with figure 2 to see that the marginal offsets have all been shaved off.

Sometimes we would like to keep the offsets on left or right. Here is sample syntax for that, although no graph is shown here.

```
twoway bar `barcall' || line `linecall' || scatteri `scattericall' ///
plotregion(margin(b=0 t=0))
```

The syntax is mnemonic: **b** means bottom, **t** means top, and if you need them, **l** and **r** mean left and right, respectively.

What we want is areal shading, so calling up **twoway area** should seem as natural as calling up **twoway bar**. To ensure two separate shaded zones, we need to call **twoway area** twice.

```
twoway area upper y if inrange(y, 1914, 1918), bcolor(gs14) base(10) || ///
area upper y if inrange(y, 1939, 1945), bcolor(gs14) base(10) || ///
line `linecall' || scatteri `scattericall' plotregion(margin(zero))
```

However, the need for repetition would become tedious (and more error-prone) if several zones, not just two, need to be identified in this way. Because the graph looks very similar to figure 3, it is not shown here.

A little more of a mental stretch, but as effective, is to specify **twoway line** with the **recast(area)** option. **recast()** is billed as an “advanced” option, which might be needlessly scary. A line plot recast as an area plot implies areal shading below the line, except that the **base()** option gives a lower bound to the shading.

```

twoway line upper y if inrange(y, 1914, 1918), recast(area) color(gs14)
      base(10) || ///
line upper y if inrange(y, 1939, 1945), recast(area) color(gs14)
      base(10) || ///
line `linecall' || scatteri `scattericall' plotregion(margin(zero))

```

The final possibility I will mention (there are yet others) is to specify the vertex coordinates for each zone directly, that is, each coordinate pair defines a corner of a rectangle. As before, we also include `recast(area)`.

```

twoway scatteri 30 1914 30 1918 10 1918 10 1914, recast(area) color(gs14)
      base(10) || ///
scatteri 30 1939 30 1945 10 1945 10 1939, recast(area) color(gs14)
      base(10) || ///
line `linecall' || scatteri `scattericall' plotregion(margin(zero))

```

3 Extra twists and tweaks

I bundle together here various smaller or subsidiary points.

The aesthetic choices of shading, including using colors if available, do raise matters of taste, but advice is still possible. Shading is most effective when the area shaded is just a small fraction of the total graph area. (If everything is emphasized, then nothing is!) If a graph showed mostly periods of recessions, flipping the comparison so that times of growth were emphasized would be a good idea. Clearly, the shading should be light enough not to be obtrusive but strong enough to be noticed easily; ideally, the reader can tune the shading in and out of conscious concern. The contrast between `gs14` and white is, arguably, a case in point. Above all else, the data must still be readable, but plotting them on top of the shading usually is sufficient to ensure that.

Using other colors will often be possible, especially for a presentation or for the Internet. But use subdued, subtle colors and consider the difficulties that some readers will have distinguishing various colors. See <http://colorbrewer2.org/> or [Brewer \(2015\)](#) for more advice.

Added reference lines pose a little challenge. If you used (for example) `yline()` to add a reference line (say, for a key threshold, or to show a summary statistic such as the series mean), you would find that the line disappears under the shading when the two meet. Some users find that disconcerting, although Stata's philosophy is that added lines always yield to data elements. Of various solutions, perhaps the simplest is to use `twoway function` as the last `twoway` call. Here note that `twoway function` can show functions that are constants; conversely, its use for showing variable functions can be helpful, too.

If you want one graph like this, you might want several, and so the question arises of how to automate the choice of zone limits. In our example, it was easy enough to see that `twoway` had chosen 30 and 10 as the extreme axis labels. If we were drawing many such graphs, having to make individual choices for each graph would be less attractive.

Start with the rule that the axis range must include the minimum and maximum values in the data. Then consider that we usually want “nice” rounded numbers to be shown. Nice is easy to recognize—2 and 50 are nice but 1.9786 and 49.635 are not—but harder to define. Heckbert (1990) gave an excellent first try at simple rules. Hardin (1995) used these ideas in a Stata context, and they underlie Stata’s default choices. We will usually round up a bit from the maximum and down a bit from the minimum. Here is some technique, illustrated first on the births data.

```
. summarize sc_births if inrange(y, 1900, 1960), meanonly
. local bmax = 5 * ceil(r(max)/5)
. local bmin = 5 * floor(r(min)/5)
. display "`bmax' `bmin'"
30 15
```

The call to `ceil()` (think “ceiling”) rounds to the nearest multiple of 5, but always to the same value or upward. Conversely, the call to `floor()` rounds to the nearest multiple of 5, but always to the same value or downward. See Cox (2003) for more on building with floors and ceilings. There is a choice in there of 5 as a suitable step size; subject-matter knowledge will often make that choice easy, or ambitious readers may wish to think how to automate a choice.

We need to think of the other y -axis variable too, deaths as well as births, but the technique is more of the same. The one extra point is simple: in combining results for two or more variables, we need the largest maximum and the smallest minimum.

```
. summarize sc_deaths if inrange(y, 1900, 1960), meanonly
. local dmax = 5 * ceil(r(max)/5)
. local dmin = 5 * floor(r(min)/5)
. display "`dmax' `dmin'"
20 10
. local max = max(`bmax', `dmax')
. local min = min(`bmin', `dmin')
. display "`max' `min'"
30 10
```

Transferring such ideas to your own do-files will cut down on the need for human intervention.

Finally, I note that all the examples here are of zones defined by intervals on the x axis. How to select zones defined by intervals on the y axis is set as an exercise for the industrious, with the hint that various `twoway` commands offer a `horizontal` option.

4 Conclusions

Shading zones on time series and other plots yields to simple tricks, even though they may require parts of the `graph` syntax you do not yet know. The main idea is to think of your graph in stages. It may be easiest to start thinking about the data you want to

show, say, as points or lines, but graphically the first step is to lay down the shading as a backdrop so that the data can be placed on top.

Sometimes, you just want a single graph of this kind. If you want a series of such graphs, automating axis extremes and other choices pose small programming problems, soluble in their turn.

5 References

- Baum, C. F. 2006. nbercycles: Stata module to generate graph command (and optionally graph) timeseries vs. NBER recession dating. Statistical Software Components S456746, Department of Economics, Boston College.
<https://ideas.repec.org/c/boc/bocode/s456746.html>.
- Brewer, C. A. 2015. *Designing Better Maps: A Guide for GIS Users*. 2nd ed. Redlands, CA: ESRI Press.
- Cox, N. J. 2003. Stata tip 2: Building with floors and ceilings. *Stata Journal* 3: 446–447.
- . 2006. Stata tip 39: In a list or out? In a range or out? *Stata Journal* 6: 593–595.
- . 2009a. Stata tip 78: Going gray gracefully: Highlighting subsets and downplaying substrates. *Stata Journal* 9: 499–503.
- . 2009b. Stata tip 82: Grounds for grids on graphs. *Stata Journal* 9: 648–651.
- Hardin, J. W. 1995. dm28: Calculate nice numbers for labeling or drawing grid lines. *Stata Technical Bulletin* 25: 2–3. Reprinted in *Stata Technical Bulletin Reprints*, vol. 5, pp. 19–20. College Station, TX: Stata Press.
- Heckbert, P. S. 1990. Nice numbers for graph labels. In *Graphics Gems*, ed. A. S. Glassner, 61–63. Cambridge, MA: Academic Press.
- Pisati, M. 2004. Simple thematic mapping. *Stata Journal* 4: 361–378.

About the author

Nicholas Cox is a statistically minded geographer at Durham University. He contributes talks, postings, FAQs, and programs to the Stata user community. He has also coauthored 15 commands in official Stata. He was an author of several inserts in the *Stata Technical Bulletin* and is an editor of the *Stata Journal*. His “Speaking Stata” articles on graphics from 2004 to 2013 have been collected as *Speaking Stata Graphics* (2014, College Station, TX: Stata Press).