The Stata Journal (2024)
24, Number 3, pp. 546–550 DOI: 10.1177/1536867X241276116

Stata tip 157: Adding extra lines to graphs

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

1 The goal

This tip is a miniature review of how you can add extra lines to graphs in Stata. Examples of why you might want to do so include showing or emphasizing key reference levels, such as 0 for many variables or freezing point 32°F for Fahrenheit temperatures; showing means, medians, geometric means, or any other summary; or flagging key dates such as 1812 or 1939. Those examples would all imply adding either vertical or horizontal lines, depending on which variable is on which axis. Some of the techniques to be discussed here allow addition of sloping or even curved lines.

If your interest is in adding shaded zones to plots, see Cox (2016) or Schenck (2020). For more on grid lines, see Cox (2009).

2 Added line options

2.1 Options in twoway

Typing

```
    . help added line options
```

tells you about the `xline()`, `yline()`, and `tline()` options for `graph twoway` and suboptions controlling them. In particular, `xline()` will draw vertical lines for which the x-axis variable plotted horizontally is constant, while `yline()` will draw horizontal lines for which the y-axis variable plotted vertically is constant. `tline()` is essentially a twist on `xline()` with some extra understanding of dates or date times. In each case, you specify the constant or constants you want to be shown, as in `yline(42)` or `xline(1939 1945)`.

Such lines always extend over the entire vertical or horizontal range of the plot region, with scope to control whether that does or does not include being shown in the margin of the plot region.

Some further key points are not obvious until you experiment or look at other documentation.

Stata will always draw these lines before it lays down points, lines, or areas representing variables that are your data. The motive here is that such added lines are usually references or context showing interesting or informative detail. You can tweak the appearance of such lines through suboptions, but that may not be enough to stop such lines being occluded or obscured by overplotting of data elements. That is especially common if you are using bars or more generally areas to show data. Stata takes showing your data directly to be more important than showing added lines. If you want your added lines to be more prominent than these suboptions allow, you need to use a command (not an option) to specify added lines, as discussed in section 3.

Often, indeed perhaps usually, you will want to add axis labels to any such line. Typing

```
. help axis label options
```

tells you about the possibilities. Indeed, you could use such options to control added lines directly. Even if a grid line does not appear by default with whatever graph scheme is in effect, you can insist on having a grid line. Once a grid line exists, you can, as with added line options, control its appearance.

The provision of minor label options implies, apart from the documented consequence of a default smaller label size, scope for added lines with different colors, patterns, and so forth.

See also Cox and Wiggins (2019): despite its overt focus on axis ticks, it touches on various tricks for axis labels.

2.2 Options with graph bar, graph hbar, and graph dot

As documented under the commands concerned, you can also specify `yline()` with `graph bar`, `graph hbar`, or `graph dot` but not `xline()` or `tline()`.

Here is a twist that is well documented but often overlooked: The y axis in this case is always taken to be the axis showing the outcome or summary being plotted. That axis is the vertical axis with `graph bar` or `graph dot, vertical`, and it is the horizontal axis with `graph hbar` or `graph dot`. Note that the `vertical` option of `graph dot` is undocumented. It is often helpful.

With the other commands just mentioned, the other axis is regarded as a categorical axis, not an x axis. Perhaps that is to be regarded as a Stata idiosyncrasy, but it should help to explain why `xline()` and `tline()` are not allowed. Similarly, `xlabel()`, `xmlabel()`, and their `t` cousins are not allowed with `graph bar`, `graph hbar`, or `graph dot`.

This Stata convention flouts long-standing mathematical practices but was introduced for your convenience. Suppose you try out `graph bar` and then realize that it will not work well, usually because the text labels for categories are just mushed together and unreadable. There is not enough space available at the foot of each bar for text you want to show to be shown well. You could try work-arounds, say, reducing the text size or placing the text at an angle. Either or both might help, but often the best solution is to try `graph hbar` instead. The point then is that all you need to do to experiment is to change `bar` to `hbar`. Option calls such as `ytitle()`, `ylabel()`, or `ymlabel()` will be applied to the new y axis, which is now horizontal.

3 Commands for adding lines

`help twoway` documents all the subcommands of `twoway`.

If the straight (or other) line you want is a regression or smooth of some kind, go straight to the subcommand concerned, whether the line is to be produced by (say) plain regression fits, fractional polynomial fits, or lowess or local polynomial smoothing.

Many of the other commands can be used to add lines. Let us start with commands for which you do not need extra variables.

3.1 twoway function and twoway scatteri

`twoway function` here has some small but commendable virtues and, in particular, can plot both horizontal and sloping lines and naturally curves too, which, in a strong sense, is its main role. Constants defining horizontal lines need not be values of some variable in your dataset but can be specified directly. You can also control the horizontal extent of any added line. That horizontal extent defaults to $[0, 1]$: watch out because that interval may easily be a minute part of your data range or even outside it.

As a simple example, suppose first that we want to add a line showing the mean of the y variable to a scatterplot. `summarize` will show us the mean we want. `twoway function` really does not care that this constant as a function is indeed a constant.

```
. sysuse auto
(1978 automobile data)
. summarize mpg
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
         mpg |         74     21.2973    5.785503         12         41
. scatter mpg weight || function 21.2973, ra(weight) ytitle(Miles per gallon)
> ytitle(Weight (pounds)) legend(off)
  (output omitted)
```

In the `stsj` scheme used for the *Stata Journal*, the function is drawn as a dashed line.

Now we have started, so let's continue down the same road. You could try to get vertical lines by trying to draw an absurdly steep line over a very narrow horizontal range, but that is not recommended. The next trick is much more flexible. `twoway scatteri` lets us plot specified (x, y) coordinate pairs. From elementary geometry, two distinct points define a line uniquely, so we can plot two points defining the ends of our desired line. The invaluable `recast(line)` option converts our pair of points to a straight line joining those points.

Let's add a vertical line showing the mean of the predictor variable.

```
. summarize weight
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
      weight |         74    3019.459    777.1936       1760       4840
. scatter mpg weight || function 21.2973, ra(weight)
> ytitle(Miles per gallon) xtitle(Weight (pounds)) legend(off)
> || scatteri 12 3019 41 3019, recast(line) lp(dash)
> note(added lines show {it:y} and {it:x} means)
```

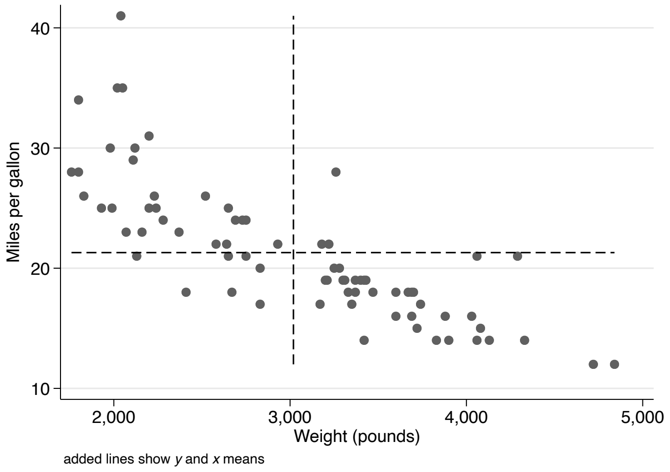

Figure 1. Scatterplot of miles per gallon versus weight for cars from the auto data. Added lines show the means for each variable.

Figure 1 shows the result. Graphs like this are often used to explain Pearson correlation or regression. Readers are invited to consider incidence of data points in each of the quadrants defined: the top right quadrant shows data points greater than both means and so on.

For this example, we could have done that just about as well with `xline()` and `yline()`. But often, we do need this extra technique, particularly if there is much shading of bars or other areas on our graphs. The simple rule is that whatever is specified later in a `twoway` call is plotted on top of whatever was specified earlier.

You will now see that we could have used a call to `scatteri` to define the horizontal line too. Naturally, you can tweak the appearance of each line in all sorts of ways. Further, you could use both subcommands to define sloping lines.

The example was made entirely concrete. We used `summarize` to get the constants we needed, not just the means but also the range on `mpg`. However, it is by no means essential to type particular constants into commands interactively or even in do-files. You could, and often should, use saved results to automate the process, particularly if you wanted a long series of similar plots. So, immediately after `summarize`—or indeed after `summarize, meanonly` (Cox 2007)—you can pick up results such as the mean, the minimum, and the maximum from `r(mean)`, `r(min)`, and `r(max)`. You may need to copy those to scalars or local macros to stop them from being overwritten, but the entire process is programmable.

3.2 Other twoway commands for adding lines

Some other `twoway` commands offer scope to specify specific data points that define lines, including arrows. Note particularly `pci` and `pcarrowi`. For more details and examples on arrows, see Cox (2005).

Alternatively, you could always create a variable holding a constant to be used to add an extra line. Or you could create a variable such that distinct y or x values define lines, whether horizontal, vertical, or sloping. That device may offend a programmer's sense of style or efficiency, which is largely why we started with other ways to do it. However, such a device is easy to implement and to understand from reading code and seeing its effects.

The existence of these commands, and the scope for repeating them in a `twoway` call, makes it easy to add lines with differing colors, patterns, and so forth as taste and circumstance suggest.

References

Cox, N. J. 2005. Stata tip 21: The arrows of outrageous fortune. *Stata Journal* 5: 282–284. https://doi.org/10.1177/1536867X0500500214.

———. 2007. Stata tip 50: Efficient use of summarize. *Stata Journal* 7: 438–439. https://doi.org/10.1177/1536867X0700700311.

———. 2009. Stata tip 82: Grounds for grids on graphs. *Stata Journal* 9: 648–651. https://doi.org/10.1177/1536867X0900900411.

———. 2016. Speaking Stata: Shading zones on time series and other plots. *Stata Journal* 16: 805–812. https://doi.org/10.1177/1536867X1601600315.

Cox, N. J., and V. Wiggins. 2019. Stata tip 132: Tiny tricks and tips on ticks. *Stata Journal* 19: 741–747. https://doi.org/10.1177/1536867X19874264.

Schenck, D. 2020. Adding recession shading to time-series graphs. *The Stata Blog: Not Elsewhere Classified*. https://blog.stata.com/2020/02/13/adding-recession-shading-to-time-series-graphs/.