The Stata Journal (2019)
19, Number 4, pp. 989–1008 DOI: 10.1177/1536867X19893641

Speaking Stata: Some simple devices to ease the spaghetti problem

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

Abstract. The spaghetti problem arises in graphics when multiple time series or other functional traces show mostly a tangled mess. Devices to improve on graphical defaults include transformed scales (especially logarithmic scales); trying to increase the graph area showing the data (especially by losing the legend whenever possible); different colors sometimes; subdividing data into a few groups; subtraction to focus on residuals or smoothing to reduce noise; selection or sampling of what is shown or emphasized; and stacking series vertically.

Keywords: gr0080, graphics, line plots, quantile plots, marker labels, panel data, longitudinal data

1 The spaghetti problem in statistical graphics

The spaghetti problem is easy to explain. Spaghetti plots are those showing many tangled lines—say, for multiple time series or other functional traces—that can be hard to distinguish and interpret. We may see broad collective patterns, but can we easily focus on individual series too or tell apart fine structure and mere noise?

My attempts to identify who first used *spaghetti* with this meaning have been frustrated twice over. Recent examples of use include Wong (2010), Knaflic (2015), and Camões (2016), but my guess is that the term has longer roots, which may prove difficult to identify. Most mentions of spaghetti really do concern pasta, not graphics. Most uses of the term in graphical contexts appear to be informal and do not carry over to literature. It is perhaps better to treat the word as one readily occurring to many people faced with a tangled mess when looking at graphs. Its meaning is more crucial than its history. Much usage is negative, so spaghetti is, as said, a problem. The problem is to give a clearer picture of the data, even if the conclusion might remain that many series are loosely similar.

A related problem might be called the paella problem. Paella plots show multiple point patterns for many groups, sufficiently mixed up such that comparisons are made difficult.

This column gives a first look at some simple devices for easing the spaghetti problem. I intend to return to the problem in later columns. In particular, a sequel to Cox (2010) is long overdue. Although the problem appears most common with multiple time series, many of the ideas here carry over to other kinds of data shown by lines or curves.

2 The problem portrayed

At its simplest, the spaghetti problem—in Stata or any other comparable software—is not getting a graph, but getting a graph that works well. It can seem that whatever you do is wrong in some way. Superimposing different curves gives the problem, a tangled mess, but separating curves by showing them in different panels may make comparison more difficult and at worst requires too many panels even to be readable or worth trying to read.

The Grunfeld dataset bundled with Stata serves as a convenient sandbox for play. The dataset can be read into Stata with

```
. webuse grunfeld
```

The dataset includes various measures for 10 companies, each measured for 20 years. There are no missing values. Incidentally, the dataset is named for Yehuda Grunfeld (1930–1960) and exists in several different versions. For more detail, see Kleiber and Zeileis (2010). In essence, the Grunfeld dataset is here a test bed for graphical ideas. If an idea does not work well with the Grunfeld data, it is unlikely to work well for larger or more complicated datasets.

Figures 1 and 2 are two graphs from a standard official helper command that show the main issue.

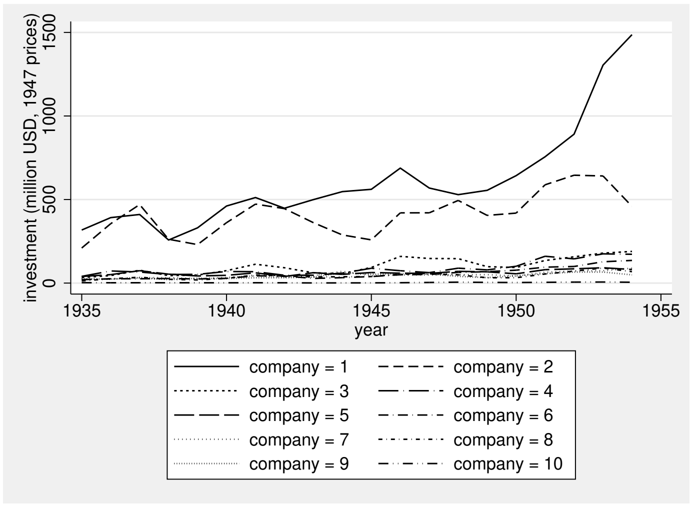

Figure 1. The spaghetti problem. Even 10 time series can display as a tangled mess, as is shown by investment in the Grunfeld dataset for 10 companies over 20 years.

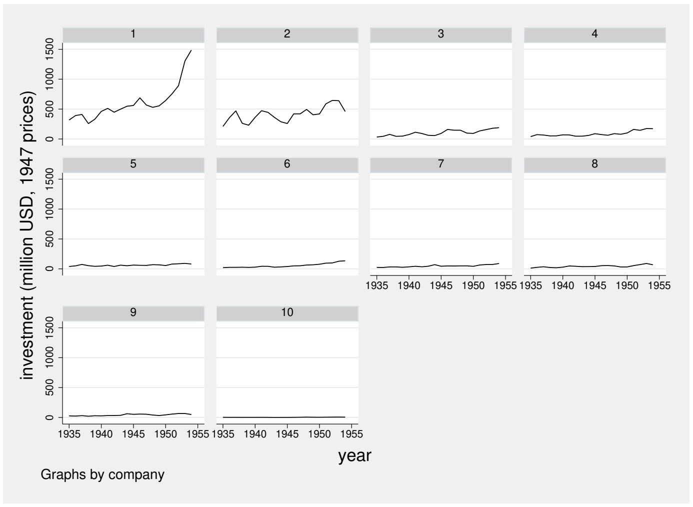

Figure 2. Separating companies, one in each panel, solves the problem only to create another. Effective comparison is just as hard.

The previous two graphs came from easy commands. But first, we right two small wrongs, an overabbreviated variable name and an absent variable label:

```
. label variable invest "investment (million USD, 1947 prices)"
. xtline invest, overlay
. xtline invest
```

Unsurprisingly, the results of default choices are often poor. A command like `xtline` does not analyze the data to guess at what choices will work well. Still less can it sense your precise goals. That is true even with a few panels and a few observations in each. At best, `xtline` and its sibling `tsline` are starting points.

3 Try a transformation

The first suggestion is not specific to spaghetti, let alone time series, but often it is a good idea anyway. Often with students or colleagues, or on Statalist or other web forums, it is my immediate advice: Have you thought of working on a transformed scale?

In turn, the most commonly considered transformed scale is logarithmic. That fits panel or longitudinal frameworks quite naturally: response variables are often both positive always and positively skewed, and patterns of growth or decline over time suggest thinking on a logarithmic scale. Exponential growth or decline is in many fields a more natural first approximation or null model than linear growth or decline. Exponential change is linear with response on a logarithmic scale.

For time series, the time variable is often best left as it arrives. Exceptions do exist but are not discussed here; see Cox (2012). It is more common to use a logarithmic scale for the response variable, which is easy through `yscale(log)`. Getting nice axis labels can be harder work, but again a fuller story is given separately; see Cox (2018).

For the Grunfeld data, we merely specify `yscale(log)` and choose sensible labels ourselves. The result is figure 3.

```
. xtline invest, overlay yscale(log) ylabel(1 10 100 1000, angle(h))
```

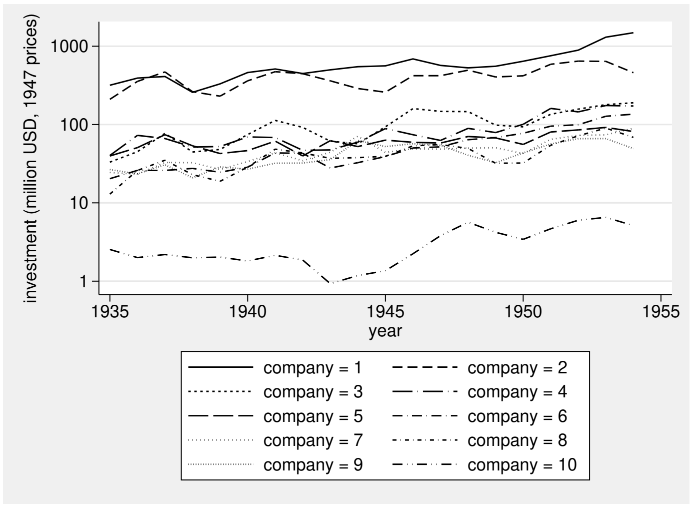

Figure 3. Try a transformation, and in particular, know that a logarithmic scale is often helpful

Other transformations can be helpful too. Logarithms belong to a family whose other members are all powers. Such a family can be parameterized in many ways, but for the moment the family resemblance can be underlined by noting that a family of transformations $T(y,p)$ for a response y and a parameter p comes from

$$T(y,p) = \int y^{p-1} dy$$

so that, in particular, $p = 0$ yields $\ln y$, just as, say, $p = 2$ yields squares y^2 and $p = -1$ yields reciprocals $y^{-1} = 1/y$. Compare Mosteller and Tukey (1977, 80) for this formulation. Here I am neglecting constants, whether additive or multiplicative, that appear on integration. Accounts of transformations that are cogent, comprehensive, and concise are thin on the ground, but Atkinson (1985) and various chapters in Hoaglin, Mosteller, and Tukey (1983, 1991) remain helpful.

A complication often met in practice but rarely discussed in texts or courses is that the response variable can be large negative or large positive. Often, we want to respect sign—and to respect zeros too—as indicating genuine characteristics of a variable. Responses do not have to be exotic or extraordinary for this to happen because it is common with any kind of response that is a change or difference, such as profit (which could be loss), net income, change in systolic blood pressure, or change in glacier terminus position (the last a personal example in Miles et al. [2013]). A little thought or experiment shows that

```
cond(y > 0, log(y), -log(y))
```

is a bad idea even if we do not have any exact zeros and a worse idea if we do. But a smarter twist such as

```
sign(y) * log(1 + abs(y))
```

behaves like $\log(y)$ for $y \gg 0$ and like $-\log(-y)$ for $y \ll 0$, is exactly 0 for $y = 0$, and changes smoothly throughout. Note that in Stata 15.1, updated 7 August 2018, and in Stata 16 onward, that calculation is a little easier using

```
sign(y) * log1p(abs(y))
```

Other transformations not yet mentioned that frequently arise in my experience are square roots and cube roots, inverse hyperbolic sines (in `asinh()`), and logits and folded roots for data that are proportions or percents.

4 Reclaim lost real estate

Figure 3 is, I suggest, an improvement for researchers on figure 1. A lay audience might need some help with logarithmic scales, on which Cox (2018) says more. But so much space in figures 1 and 3 is given up to explaining what is shown that we should try to reclaim some of this lost real estate.

4.1 Clear the axes of unnecessary detail

With time series, we can usually delete the default `xtitle()`, here an echo of the name of the time variable `year`. Because the years 1935(5)1955 are clearly shown, we surely do not need that. My Stata advice often includes a suggestion to remove such a title, let alone variable names like `date` or `time`. `xtitle("")` suffices to excise.

Naturally, it is good graphical practice to explain what is shown on each axis, as (I guess) your high school and even your college or university teachers were at pains to emphasize. Nevertheless, rules here are guidance for the wise, and there is a broader imperative to simplify by removing unneeded detail. Conversely, if your time variable is unusual, that might be better explained in the caption written in your word or text processor.

A similar detail often arises. Look back to figure 2 and find the text “Graphs by company”. A graph that uses the `by()` option, which in this case underlies the `xtline` code, allows a `note()` suboption naming the grouping variable. If such a note appears, often it is best deleted too, if only because the grouping variable is evident from context or explained in your text caption. The result is not shown here, but you can check that the extra option `byopts(note(""))` works to remove it. In any context in which a `by()` option is explicit, add to the recipe `by(, note(""))`.

4.2 Lose the legend

Bigger real estate gains can be achieved if you can lose the legend completely or (if you prefer) kill the key. A legend is at best a necessary evil. At worst, it takes up too much space and obliges mental "back and forth" by the reader. Most readers are unwilling to memorize even a short legend and so resort to thinking along these lines: Which company or place is this line? Which one is Apple or Amazon or Albania or Alabama or Aberdeen? Or, to be frank, the reader (me and you too?) knows that in principle the graph could be studied in detail going back and forth between figure and legend but in practice moves on to something else that is not too much like hard work.

With this dataset, repeating the text "company =" 10 times over is not essential. So, working away at the `legend()` option is possible. With other datasets, editing value labels or string values for the grouping variable (in this example, panel identifier) to something shorter is an equivalent way forward. Broad hint: Don't overwrite the existing labels or string values; create more concise alternatives. That longer text could be precisely what you need for another purpose, even if it is only to be metadata that explain the grouping variable to you or others.

Direct labeling is widely recognized and recommended for statistical graphics (see, for example, Kosslyn [2006], Few [2012], and Wilke [2019]). In Stata, that can mean adding text directly within the plot region with a dedicated option, but using marker labels is usually as effective and much easier. This really is not a new idea: William Playfair was generally at pains to explain each series on a graph and close to it (see, for example, Playfair [1801]).

Let us see some sample extra code for adding text labels at the end of each series. We put the code in a local macro mostly so that commands to come do not get excessively long. If you are familiar with local macros, you can skip the next two paragraphs and focus on the next line of code.

If you are new to local macros, think that we are putting the text in a kind of bag, which we name. Later, we will tell Stata to take the text out of the bag and use it immediately.

The name "local" is not arbitrary or meant as mischievous or even mystifying, as jargon sometimes is. It flags that the macro is visible only locally, meaning within the same program space. Here we might imagine an interactive session in which commands are typed one by one. Local macros work fine in that context. Whatever was defined earlier in a session remains visible unless it was overwritten. In many ways, it would be better practice to put all the code in a do-file, and that will work fine too. Being local bites in this way: local macros in one do-file cannot automatically see local macros in another do-file or in the main interactive session, and conversely. That is a strong feature, not a limitation. It means that we can define local macros without worrying about messing up those somewhere else or being messed up by those somewhere else. Much more could be said, but I stop the story there. If this is interesting or seems useful, then there is no better place to start careful reading than chapter 18 on "Programming Stata" in the *User's Guide*.

```
. local endlabels addplot(scatter invest year if year == 1954, msymbol(none)
> mlabel(company))
```

That syntax may seem puzzling. There are three parts: the command name `local`; the name of the macro—here `endlabels`—that the command is defining; and whatever follows, which is the text copied into the local macro. We could use an equals sign and put the text—the macro contents—in double quotes, but we do not have to do that.

The `addplot()` option allowed by `xtline` lets us specify a plot to be superimposed in the same graphical space. We plot investment against year, but for the last year only, 1954 in this dataset. We suppress the marker symbol but add as a marker label the company identifier, which as seen already is an integer 1 to 10. The default marker label position using a clock position syntax is 3. Think of a conventional clock or watch with hours 1 to 12, so 12 at the top, 3 on the right, and so on. Stata adds a rule that 0 indicates the center of the display, where a data point would be. Then a position of 3 means plotting on the right of the data point. This default is exactly correct for marker labels to be added at the end of a series. Remembering to remove the `xtitle()`, we can now try this. The result is figure 4.

```
. xtline invest, overlay yscale(log) ylabel(1 10 100 1000, angle(h)) xtitle("")
> `endlabels´ legend(off)
```

Note the syntax to instruct Stata to take the code out of the local macro. Stata will use it immediately, replacing the macro name by its contents. The result of that substitution is what is passed to `xtline`. If you have not used local macros before, be careful to use the correct quotation marks, ‘ and ’.

We also need to spell out `legend(off)`, which is much of the point.

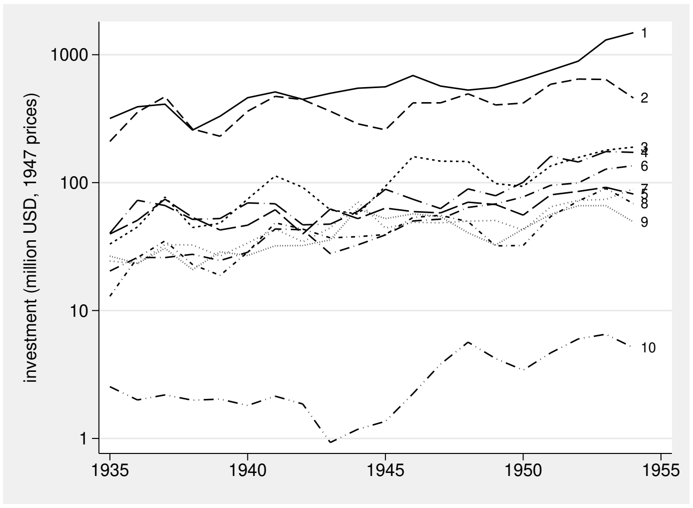

Figure 4. Lose the legend if you can. Here the integer identifiers are used as labels at the end of each series.

The result clearly saves on space, which was the main goal. But problems may still persist. If two or more marker labels are in (almost) the same position, then they will be mushed together. In practice, this is not usually a disaster. If two series are similar, we do not need to worry much about which is which. If extreme series (highest or lowest) are of particular interest, they will usually be easy to discern.

Here the grouping variable is an integer identifier with values 1 to 10. Other identifiers with small integers can work well too. If you do not have such a variable, then the `group()` function of `egen` (see [D] **egen**) offers an easy way to produce one.

Short but evocative string identifiers can work fine too, indeed usually better. If your identifier is numeric with a value label and you want to see the value label, you need to `encode` the variable and then pass the resulting string variable to `mlabel()`. You are likely to be familiar with ISO country codes (say, DE or FR), abbreviations for states in the United States (say, TX or MA) or for subdivisions of your own country, and especially classifications with short codes in your own field (industries, diagnoses, etc.).

Worth mentioning even though obvious on reflection: if the identifiers have no meaning to you, then you will lose nothing by omitting the legend.

With the Grunfeld data, there is a happy small accident. The last year in the dataset is 1954, but `xtline` is already minded to show axis labels up to 1955, so there is enough space for the extra labels to fit anyway. This is not guaranteed, but at worst you may need to spell out to Stata that the x axis must be extended.

Suppose you wanted labels at the start of each series too. To my taste, and in my experience, this is neither attractive nor needed, but your problem could differ. For this, you could modify the extra code to

```
. local endlabels2 addplot(scatter invest year if year == 1954,
> msymbol(none) mlabel(company) || scatter invest year if year == 1935,
> msymbol(none) mlabel(company) mlabposition(9))
```

So we have an extra graph plotted in the same space, putting text labels by the values for 1935, but this time plotted on the left using position 9. If you try this, you will find that it does not work well unless you add a further option such as `xscale(range(1934 .))` to extend the range of the x axis.

Before I leave this topic—which is really just one easy idea, but the devil is in the details—another design should be mentioned. I do not consider the result especially appealing, either here or even more widely, but the idea is simple to understand and to implement, and it may be a trick worth knowing.

It is easiest here to back out of `xtline` and go for a conventional scatterplot. Code first and result in figure 5:

```
. scatter invest year, yscale(log) ylabel(1 10 100 1000, angle(h))
> msymbol(none) mlabel(company) mlabposition(0) xtitle("")
```

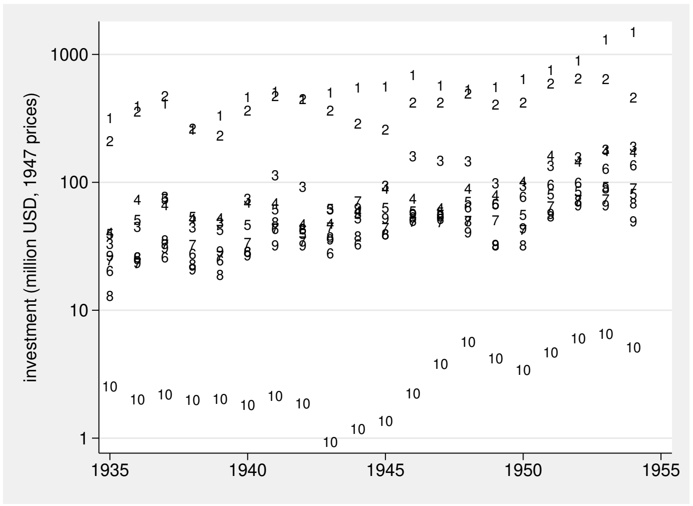

Figure 5. Every marker label can serve as a self-explanatory marker (so long as you can read them all)

So, simply, every marker label becomes its own self-explanatory marker. Note the detail of `mlabposition(0)`. Because no marker is shown, the marker label can and should be placed where the marker would have been.

How well this works is sensitive to dataset size. If you have not 20 observations in each series, but 200, 2,000, etc., the marker labels will just mush together, and the design will fail. More positively, different colors for different series can still help. A special bonus: so long as the marker labels are legible, the colors do not need to be explained in a legend. If series 7 with marker label 7 is always in red, you do not need a legend entry to tell you that red means 7. Quite how to do this we will set on one side beyond hinting that `separate` (see [D] **separate**) can be mighty helpful. See also Cox (2005b).

5 Comments on colors

In this column, we are using the *Stata Journal* scheme, which you can play with too using

```
    . set scheme sj
```

Denying ourselves color may also match your situation. You may be planning to submit an article to a journal that has limitations on use of color. Even in that situation, you could use a different scheme, say, `s1mono`. Conversely, for a seminar or conference presentation, exploiting color is likely to be not just possible, but expected.

Leaving aside a full discussion of color, the subject of many books and articles, I confine myself here to some concise comments. Wilke (2019) is sensible and sensitive on use of color.

Using 2 or 3 colors can work well, but not usually 10 or 12, let alone 20 or 30. Remember that using a color typically creates an expectation that you will explain it, which raises the problem once more of a legend taking up much of the available space.

Colorful may sound good, but often colors are not so crucial. If we explain each series otherwise, say, with text labels, as discussed at length in the previous section, we may manage well without colors. Multiple colors can even seem garish or confusing, and there can be “fruit salad” or “technicolor dreamcoat” effects. Too many distinctions made on the graph can make it hard to distinguish in one’s mind.

Never use red and green together: use red or orange and blue. Difficulty in distinguishing red and green is a common visual limitation. People who struggle with red and green together are often comfortable with red and blue, which may well have connotations for your data anyway: red is Republican in United States politics but left wing in many other countries, red is negative in finance, and so on. Orange and blue is a modish combination in many circles, and with good reason. There are many examples in Hastie, Tibshirani, and Friedman (2009) and Wilke (2019).

Going grey or gray is good (Cox 2009b)! (The spelling of this colo[u]r can change mid-Atlantic.) A successful design can be to single out just a few series (even one) of special interest or importance, show those with strong colors, and show the rest as context in gray and possibly also with thin lines. See also Cox (2010) for a related idea, and recall from the beginning of this column that a sequel is promised, which may even have appeared by the time you read this.

6 Subdivide

If showing everything in one graph panel is too messy and showing each series separately is not a good solution either, perhaps there is a compromise to be found in some other way. Find a subdivision to subsets, each including several series, so that the spaghetti problem is at least reduced, if not removed.

Thus, consider a common example: wanting to show series for each of the 50 states of the United States (possibly even the District of Columbia and Puerto Rico too). A separate graph panel for each state would not, I imagine, be universally regarded as ideal. I have often seen it done, but I suspect that the reader reaction is more often "Wow! I didn't know you could do that" than it is "Aha! That really helps understanding of the data".

Typically, the best subdivision is substantive and depends on the data and the aims of the research. The states might be grouped by region, population density (giving a rural–urban contrast), or whatever else makes sense. The goal is not necessarily to avoid crowding, if what is wanted includes showing that a group of states has similar series.

Here is a small trick that can be used quite generally. In the Grunfeld data, the identifiers run from 1 (overall, the largest company) to 10 (overall, the smallest company), so dividing identifiers into odd and even would reduce the overlap between series. So, the odd identifiers are companies 1, 3, 5, 7, and 9 and the even identifiers are 2, 4, 6, 8, and 10.

Odd and even are indicated by the remainder on dividing the identifier by 2. Testing whether that is 0 gives 0 (false) if the identifier is odd and 1 (true) if the identifier is even. Such an indicator variable is 0 for odd (which will plot on the left with the command below) and 1 for even (plot on the right).

```
. generate which = mod(company, 2) == 0
. local endlabels scatter invest year if year == 1954, msymbol(none)
> mlabel(company)
. line invest year, connect(L) by(which, note("") legend(off))
> yscale(log) xtitle("") subtitle("") ylabel(1 10 100 1000, angle(h))
> || `endlabels´
```

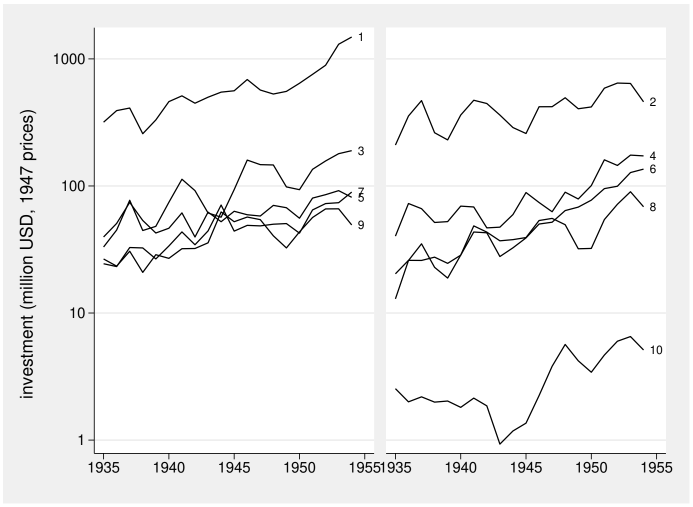

Figure 6. Relieve the strain of many series in one graph panel by grouping into two or occasionally more panels

Figure 6 is the result. Some details are new with this example. We need to redefine the code for the end labels because we are no longer using `addplot()`. We need to retreat to use `line`, the price being that we have to redo some work that `xtline` would carry out unbidden. We need to use `connect(L)` to inhibit spurious connections between the series for different companies. (Inhibit does not mean prohibit: there could still be problems, but the device mentioned next solves them anyway.) We can and should suppress the uninformative subtitles for each graph panel (which would be just 0 and 1).

A weakness of this design is using the same line pattern (by default solid) for all the companies. That is easy enough to fix. Separate variables for each company will automatically be plotted differently (see also Cox [2005b]). With several variables on the y axis, `graph` will give up on `ytitle()` unless you supply one. We could type in the variable-label text once again but prefer instead to automate choice, using syntax described at `help macro` to call up that variable label. Figure 7 is the result.

```
. separate invest, by(company) veryshortlabel
  (output omitted)
. line invest1-invest10 year, by(which, note("") legend(off))
> yscale(log) ylabel(1 10 100 1000, angle(h)) xtitle("") subtitle("")
> ytitle("`: variable label invest´") || `endlabels´
```

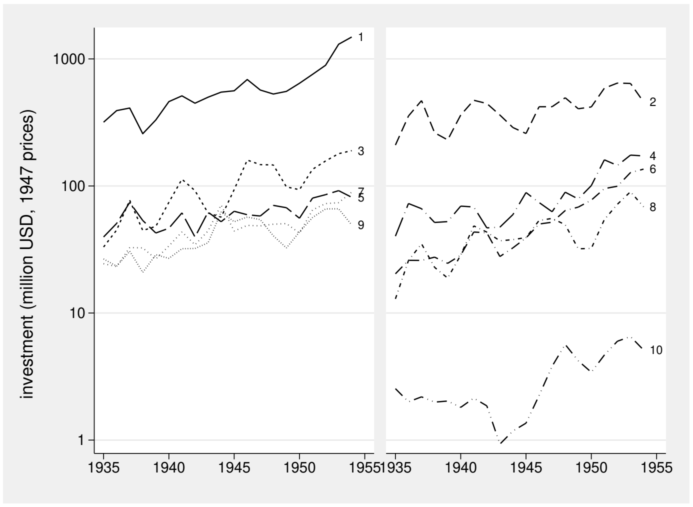

Figure 7. Each series can be shown with different line patterns

Rarely is a graph the last possible version. If colors were allowed, that possibility should be considered. You might want to change the line patterns in any case. Dotted patterns can be recessive and not show well if the graph is ported to other software.

There is a tradeoff: increase the number of graph panels for the sake of fewer series in each, and you can lose more clarity than you gain. Three panels side by side could just about work if the series are not long. Four panels, presumably in two rows and two columns, loses some comparability because some panels are on different rows. And so on.

See Cox (2009a) for a detailed example in which changes over time were plotted for different groups of patients. With only two time points, three groups could be plotted in one row.

7 Subtract or smooth

The focus so far has been on plotting the data as they arrive, modulo a possible decision to work on a transformed scale, which would not in any case change the data strongly. But data analysis clearly does not stop at exploration of data. For some of its practitioners, that is not even part of their practice. Looking at the data to see what they might say is for them at best redundant and at worst a source of prejudice or presumption. That aside, it can be agreed that what might be better plotted are the results of data analysis.

The range of strategies that may be considered is just about the range of strategies in statistical science, but mentioning two extremes should be enough to convey some flavor.

Statistics can be thought of as answering the elementary but fundamental concern of whether things are similar despite differences or different despite similarities, a banality until we develop quantitative handles for comparison. At one extreme, we cannot easily see differences in behavior unless we calculate some kind of summary (which could be as simple as a mean or median) or fit some kind of model. Then, we can subtract to get residuals (observed − summarized or fitted) to see what is idiosyncratic.

The opposite situation is also frequent. Sometimes, all we need is some modest smoothing to remove minor fluctuations that are likely to be noise or at most fine structure that does not hold interest or importance.

Now for something quite different—using quantile plots to compare height distributions of members of the New York Choral Society in 1979. The data come from Chambers et al. (1983, 350) and were previously analyzed in Cox (2007). The data are a small convenience sample and are best not taken too seriously but serve to provide an illustration of technique. A quantile plot using the community-contributed command `qplot` (Cox 1999, 2005a, 2019) shows ordered values versus their so-called plotting positions, here (unique rank − 0.5)/sample size by default. You must install `qplot` from the *Stata Journal* website if you wish to use it yourself. `qplot` is a considerable generalization of the official command `quantile`.

If plotting positions are new to you, check this recipe out with a simple example. With a sample, say, of size 7, the median would be at rank 4, which is a plotting position of $(4 - 0.5)/7 = 0.5$, so the plotting position is an approximation to cumulative probability. `quantile` and also `qplot` talk of "fraction of the data" by default. Several other plotting-position recipes would check out for the median in the same way: they differ slightly in what they return as cumulative probabilities on either side of the median. The art is to avoid plotting positions of exactly 0 or 1, if only because under various transforms, notably feeding plotting positions to quantile functions, such points cannot be plotted.

The dataset file for this example is bundled with the media for this column.

```
. use singer_height, clear
(source: http://www.stat.purdue.edu/~wsc/visualizing.tables.txt)

. qplot height, by(spart, row(1) note("")) ylabel(60(3)75, angle(h))
> ytitle(height (in))
```

Here the four groups distinguished (soprano, alto, tenor, bass) are first plotted side by side for clarity (figure 8).

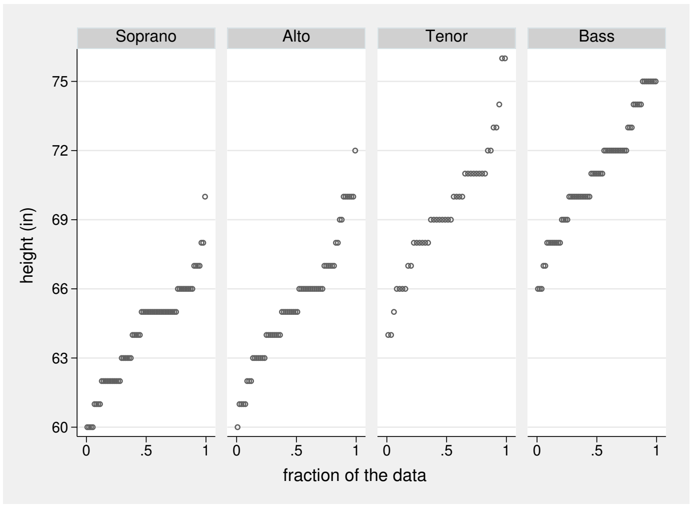

Figure 8. Singers' heights, New York Choral Society, 1979. Quantile plots for sopranos, altos, tenors, and basses.

Readers in many countries might appreciate a little help with the units of measurement. Heights are given in inches, and indeed are often converted to feet, where 1 foot is 12 inches ("feet" is the plural of "foot"). So 60 inches would often be reported as 5 feet; 72 inches as 6 feet; 66 inches as 5 feet 6 inches. So now the choice of axis labels should make more sense. More help still: 1 inch is exactly 25.4 millimeters, so, for example, 60 inches is 1.524 meters and 75 inches is 1.905 meters.

That explained, the unusual stepped nature of the quantile plots is easy to understand. Singers' heights are reported to the nearest inch. Once we know this, we should no longer care about it: it is no more than a measurement convention. But the eye and brain are still distracted by that, a secondary feature of the data, which is grounds enough for smoothing.

Quantiles can be smoothed, and easily. A good method was given by Harrell and Davis (1982) and is implemented in `hdquantile` (Statistical Software Components), which again you must install before you can use it, say, by typing

```
. ssc install hdquantile
```

A full discussion of this interesting and useful method would take us too far from the main path. Pertinent references are given in the help for `hdquantile`.

Smoothing and trying again, a first stab at a graph shows that the smoothed or estimated quantiles—in this use of the method, one for each data point—are close enough that a line representation of the quantiles is less busy, which is shown in figure 9.

```
. hdquantile height, by(spart) generate(heightHD)
. qplot heightHD, by(spart, row(1) note("")) ylabel(60(3)75, angle(h))
> ytitle(height (in)) recast(line)
```

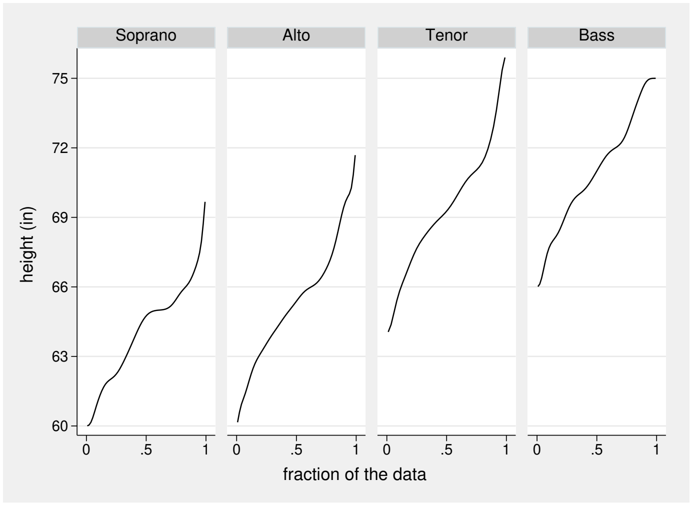

Figure 9. Quantile smoothing removes much of the distracting detail

Note that using plotting position as the horizontal coordinate is equivalent to using a uniform distribution with the same range as the data as the reference distribution. This default for `qplot` resembles the only allowed behavior for the official command `quantile`.

Advice on smoothing is often that feeling you may not have smoothed quite enough is better than worrying that you have smoothed too much. Fluctuations that look trivial can be smoothed out mentally, but it is harder to restore genuine detail in the same fashion. In this respect, figure 9 qualifies as helpful. What may well be merely quirks in the data are not all smoothed out, but it is axiomatic that chance is lumpy (Abelson 1995).

Enough noise, or fine structure, has been smoothed away now that we can try superimposing the quantile traces (figure 10). The numbers 0.25, 0.5, and 0.75 are plotting positions corresponding to lower quartile, median, and upper quartile.

```
. local labs xlabel(0 "0" 0.25 "0.25" 0.5 "0.5" 0.75 "0.75" 1 "1")
. qplot heightHD, over(spart) ylabel(60(3)75, angle(h))
> ytitle(height(in)) recast(line) aspect(1) legend(order(4 3 2 1) cols(1)
> position(3)) `labs´
```

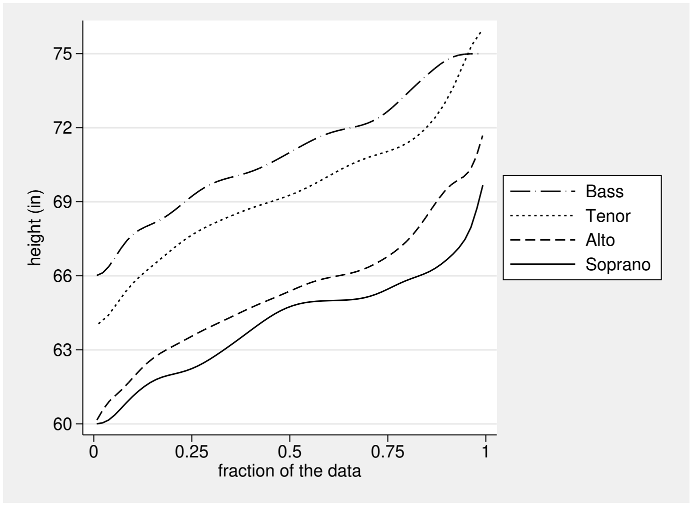

Figure 10. Smoothed quantile traces can now be seen in one graph panel

We will stop there without claiming to say everything that could be said even about this small dataset. The approximate ranking of sopranos, altos, tenors, and basses is vivid enough. For adults, sopranos and altos are traditionally female parts and tenors and basses male parts, so it is no surprise to see a contrast between the first two and the last two groups: females are not quite so tall as males. But it is a surprise to me to see systematic differences of the magnitude shown between sopranos and altos and between tenors and basses.

8 Select or sample

I am not out of suggestions yet but will close with some brief general strategic remarks.

Select. Do not feel obliged to try to show everything. Focus on what is of greatest interest or importance. In a previous section, the strategy of emphasizing a few series and showing the rest as context was flagged.

Sample. If you have thousands of series, it is possible that plotting a random sample of tens or hundreds may be as or more helpful in giving insight as plotting them all. Alternatively, Bowley (1910, 62; 1952, 73) advised use of minimum and maximum and 10%, 25%, 50%, 75%, and 90% points as a basis for graphical summary. You may be reminded somehow of box plots. A century and more on, the advice remains good. We might select series according to whether they fall at those levels on some variable or criterion based on two or more variables. Diggle et al. (2002) gave further advice and examples in similar spirit.

9 Stacking

Stacking series vertically within a single graph panel is another old idea. Implicitly or explicitly, each series is plotted distinctly within its own bounds. Tufte (2006) added impetus, and even chic, to the idea by talking of *sparklines* and giving several intriguing examples. In practice, much depends on what you are willing to give up and what you want to be a focus. Stacking can be helpful if you want to compare times at which series were at their maximum or minimum, as one goal among several. As with other strategies barely mentioned here, this approach deserves more detailed discussion.

10 Conclusion

Spaghetti arises frequently when plotting multiple series as a challenge to do better graphically. Sometimes the aim is just too ambitious, but there are many small and large devices that you can try. This column has not been comprehensive, and it has been capricious in which devices were discussed in detail, leaving much scope for further treatments.

11 References

Abelson, R. P. 1995. *Statistics as Principled Argument*. Hillsdale, NJ: Lawrence Erlbaum.

Atkinson, A. C. 1985. *Plots, Transformations, and Regression: An Introduction to Graphical Methods of Diagnostic Regression Analysis*. Oxford: Oxford University Press.

Bowley, A. L. 1910. *An Elementary Manual of Statistics*. London: Macdonald and Evans.

———. 1952. *An Elementary Manual of Statistics*. 7th ed. London: Macdonald and Evans.

Camões, J. 2016. *Data at Work: Best Practices for Creating Effective Charts and Information Graphics in Microsoft Excel*. San Francisco, CA: New Riders.

Chambers, J. M., W. S. Cleveland, B. Kleiner, and P. A. Tukey. 1983. *Graphical Methods for Data Analysis*. Belmont, CA: Wadsworth.

Cox, N. J. 1999. gr42: Quantile plots, generalized. *Stata Technical Bulletin* 51: 16–18. Reprinted in *Stata Technical Bulletin Reprints*. Vol. 9, pp. 113–116. College Station, TX: Stata Press.

———. 2005a. Speaking Stata: The protean quantile plot. *Stata Journal* 5: 442–460.

———. 2005b. Stata tip 27: Classifying data points on scatter plots. *Stata Journal* 5: 604–606.

———. 2007. Speaking Stata: Turning over a new leaf. *Stata Journal* 7: 413–433.

———. 2009a. Speaking Stata: Paired, parallel, or profile plots for changes, correlations, and other comparisons. *Stata Journal* 9: 621–639.

———. 2009b. Stata tip 78: Going gray gracefully: Highlighting subsets and downplaying substrates. *Stata Journal* 9: 499–503.

———. 2010. Speaking Stata: Graphing subsets. *Stata Journal* 10: 670–681.

———. 2012. Speaking Stata: Transforming the time axis. *Stata Journal* 12: 332–341.

———. 2018. Speaking Stata: Logarithmic binning and labeling. *Stata Journal* 18: 262–286.

———. 2019. Software Updates: gr42_8: Quantile plots, generalized. *Stata Journal* 19: 748–751.

Diggle, P. J., P. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.

Few, S. 2012. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. 2nd ed. Burlingame, CA: Analytics Press.

Harrell, F. E., and C. E. Davis. 1982. A new distribution-free quantile estimator. *Biometrika* 69: 635–640.

Hastie, T., R. Tibshirani, and J. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. New York: Springer.

Hoaglin, D. C., F. Mosteller, and J. W. Tukey, eds. 1983. *Understanding Robust and Exploratory Data Analysis*. New York: Wiley.

———. 1991. *Fundamentals of Exploratory Analysis of Variance*. New York: Wiley.

Kleiber, C., and A. Zeileis. 2010. The Grunfeld data at 50. *German Economic Review* 11: 404–417.

Knaflic, C. N. 2015. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Hoboken, NJ: Wiley.

Kosslyn, S. M. 2006. *Graph Design for the Eye and Mind*. New York: Oxford University Press.

Miles, B. W. J., C. R. Stokes, A. Vieli, and N. J. Cox. 2013. Rapid, climate-driven changes in outlet glaciers on the Pacific coast of East Antarctica. *Nature* 500: 563–566.

Mosteller, F., and J. W. Tukey. 1977. *Data Analysis and Regression: A Second Course in Statistics*. Reading, MA: Addison–Wesley.

Playfair, W. H. 1801. *Commercial and Political Atlas: Representing, by Means of Stained Copper-Plate Charts, the Progress of the Commerce, Revenues, Expenditure, and Debts of England during the Whole of the Eighteenth Century*. 3rd ed. London: Wallis.

Tufte, E. R. 2006. *Beautiful Evidence*. Cheshire, CT: Graphics Press.

Wilke, C. O. 2019. *Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures*. Sebastopol, CA: O'Reilly.

Wong, D. M. 2010. *The Wall Street Journal Guide to Information Graphics: The Dos and Don'ts of Presenting Data, Facts, and Figures*. New York: W. W. Norton.

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 16 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).