The Stata Journal (2021)
21, Number 1, pp. 263–271 DOI: 10.1177/1536867X211000032

Stata tip 140: Shorter or fewer category labels with graph bar

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

1 The problem: Messy category labels with graph bar

Stata does not have the concept of a categorical variable, but statistical people do, as shown by many book titles alone (for example, Fienberg [1980]; Lloyd [1999]; Simonoff [2003]; Tutz [2012]; Agresti [2013]; Long and Freese [2014]). The categories of such variables are distinct and often, but not necessarily, few in number. Examples are differing disease condition, employment status, or land cover type. A categorical variable in Stata can be stored as string; as numeric with value labels; or sometimes as just numeric, as when the number of cars or cats or children in households is just a discrete count that you want to treat as categorical.

The main focus of `graph bar` and its siblings `graph hbar` and `graph dot` is showing data or results for one or more outcomes and for distinct values of categorical predictor variables. Typically, the categorical predictors are named in the `over()` or `by()` option, or both. That way of thinking does not rule out using any of these commands in other ways. In practice, the difficulties discussed in this tip arise most commonly with `graph bar`, but if the solutions help with the other commands, that is fine. Also, those difficulties arise often with predictor variables that a researcher would not think of as categorical at all. The leading example here has that flavor.

The general problem addressed in this tip is that you are using `graph bar` and your categorical axis labels are a mess. They overlap and you need shorter labels, or fewer labels, or perhaps both. Although it may seem puzzling or even perverse, `graph bar` does not have an x axis; the horizontal axis is thought of as a categorical axis. The y axis is as usual the vertical axis showing outcome values or means or counts or whatever else the graph shows. One good reason for that idiosyncrasy is that typing `graph hbar` or `graph dot` instead flips the axes and indeed often improves the graph. The y axis remains the axis that shows the outcome, even though it is now horizontal. You would not want to have to edit a series of options naming x-axis properties to the corresponding y-axis properties, and conversely, which is what `graph twoway` often requires if you change axes.

The example in this thread of a bar chart for time series raises just about all the generic issues that arise commonly. It also raises some specific issues for time series that are frequent.

The immediate stimulus for this tip was a thread on Statalist,

> https://www.statalist.org/forums/forum/general-stata-discussion/general/
> 1565227-graph-bar-over-year-how-to-shorten-displayed-year-labels

which started on 24 July 2020. R. Allan Reese in particular made one suggestion incorporated here.

Let's dive in and consider the use of a bar chart for two time series from the Grunfeld dataset bundled with your version of Stata. Bars side by side for two or more variables are often wanted and one of the attractions of `graph bar`.

We are setting on one side any discussion of whether some other kind of graph, such as a line chart, would be better; or of whether it is a good idea to juxtapose two series that may not have even the same units of measurement.

Time in the Grunfeld dataset is each year from 1935 to 1954, but `graph bar` does not know or care that one variable is time. The default sort order for time as a numeric categorical variable is almost always what you want anyway, but `graph bar` pays no attention to any gaps or unequal spacing of time values.

To set yet other details out of the way, let's say that we have decided in advance what bar colors we want and where the legend should go. In practice, a script for your own data may go through several iterations as you play with possibilities and discover small points to be resolved.

```
. webuse grunfeld
. set scheme sj
. local opts1 bar(1, fcolor(gs14) lcolor(black))
> bar(2, fcolor(gs3) lcolor(black))
. local opts2 legend(pos(11) ring(0) col(1))
. graph bar (asis) invest kstock if company == 1, over(year) `opts1´ `opts2´
```

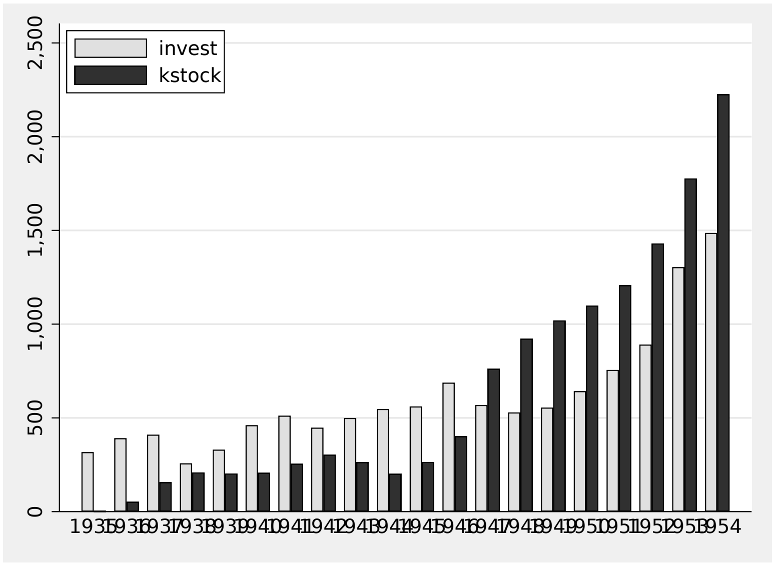

Figure 1. The time labels overlap. What to do?

Figure 1 is evidently poor, even though there are only 20 distinct years here. As you know, very many time series include far more times, but 20 is enough to cause a mess.

The thinking of `graph bar` is that you care about all your categories, so you want to know what each one is. But the default that every predictor value—every distinct category—is matched by an explicit text label is not always what you want for time series.

2 Possible solutions

2.1 Possible solution: Use graph hbar instead

Very often, the answer is simply to go horizontal by using `graph hbar`, not `graph bar`. Many of the vertical bar charts I see (some people call such charts "column charts") would be better off horizontal, giving space for longer, readable text labels and avoiding solutions that are awkward if not horrible, such as vertical labels, labels on a slant, overabbreviated labels, or labels in a tiny font size.

When this is the answer, good, and you bail out here.

For time series, this is not usually an acceptable answer. There is a strong convention across many fields that time belongs on the horizontal axis. Put your time variable on a vertical axis, and someone reviewing your work is likely to query that directly.

2.2 Possible solution: Use shorter text labels through value labels

You could assign value labels, such as `"35"` to 1935. `graph bar` respects value label definitions. You are in charge and are not obliged to use the same recipe throughout, so, for example, first and last value labels could be full length, say, `"1935"` or `"1954"`.

Typing out definitions for 20 value labels (to say nothing of many more, as would be needed for longer series) is less fun than writing a loop to write code for your later `label` command. If you are new to loops, or indeed to local macros, my recent column (Cox 2020) provides a tutorial introduction.

```
. local call
. forvalues j = 1/20 {
  2.       local show = `j´ + 34
  3.       local year = 19`show´
  4.       local call `call´ `year´ "`show´"
  5. }
. label define year `call´
. label value year year
. graph bar (asis) invest kstock if company == 1, over(year) `opts1´ `opts2´
```

The initial statement

```
. local call
```

deserves comment. Thereby, I blank out any contents of the local macro call that exist (equivalently, I delete the macro if it exists). That way, I will not get bitten because of whatever is left behind in the macro from any previous code. This device will be used repeatedly in what follows. Here the macro does not exist at the outset, but explicitly blanking it out is suggested as good style.

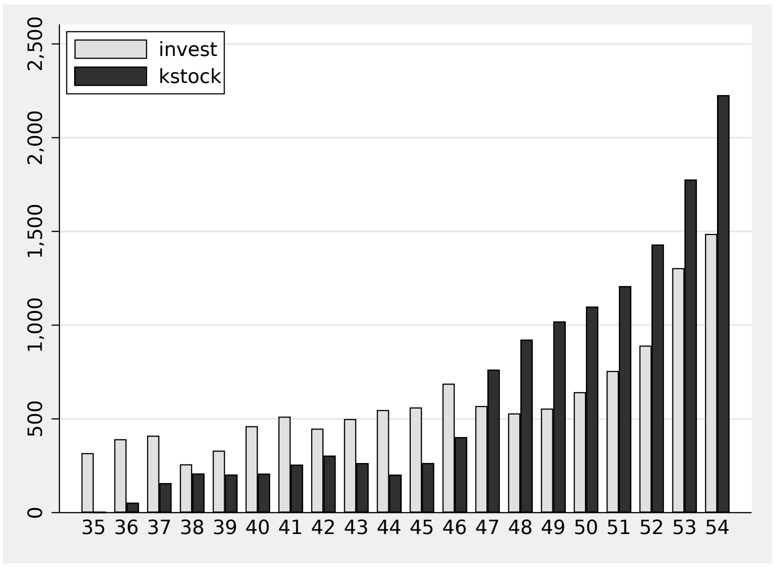

Figure 2. Insisting on shorter text labels (through value labels) removes the overlap. There are still questions: Do you want all those labels? Would this work with more distinct times?

Figure 2 is much better. You might want to stop here. But to spell out what is key: What we did was set up value labels such as `"35"` for 1935. Then, `graph bar` did not need to be told to look for and use value labels when they exist. Its expectation, as said, is that the variable named to `over()` is categorical. If that variable is numeric, then it is very likely to have value labels attached.

So far so good, but even for this example you might think "too many labels!". Your real data might be a time series with many more values, and if so, you really would think that.

Value labels that are just `"35"` to `"54"` for 1935 to 1954 are easy enough, but what if your years were 2000 to 2019 and you wanted the graph to show `"00"` to `"19"`? See a previous tip (Cox 2010) for how to get leading zeros when wanted.

Note that although you can set some of your value labels to spaces or even more exotic characters such as `char(160)` or `uchar(160)`, such value labels are not honored by `graph bar`. So we need some other device.

2.3 Possible solution: Use relabel()

The help for `graph bar` tells you about this suboption, enabling you to spell out whatever text you want. That does allow blanking out in a strong enough sense. In our running example, `graph bar` numbers the bars 1 to 20 (and so not 1935 to 1954). Here I write code using a loop for the suboption `relabel()` such that odd-numbered bars have as text a two-digit year (omitting century information) and even-numbered bars have as text a space (which you should not notice, except as resembling no text). Detail: an empty string will not work, because the command will not believe that you do not want a label at all.

```
. local call
. forvalues j = 1/20 {
  2.       local show = `j´ + 34
  3.       if mod(`j´, 2) local call `call´ `j´ "`show´"
  4.       else local call `call´ `j´ " "
  5. }
. graph bar (asis) invest kstock if company == 1, over(year, relabel(`call´))
> `opts1´ `opts2´
```

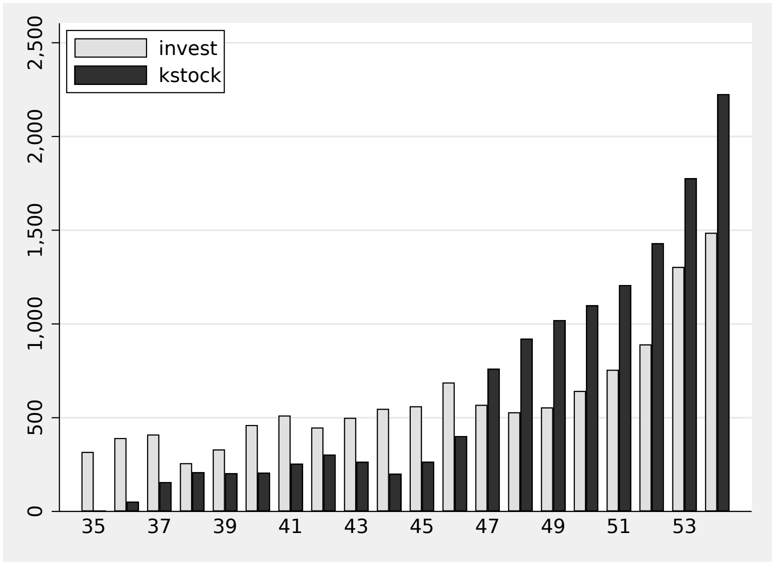

Figure 3. Every other year label is blanked out. Strictly, you are seeing spaces as value labels for even years, but the effect is identical.

Figure 3 shows the effect of showing fewer text labels.

If the `mod()` function is new to you, here is the small story. `mod()` returns the remainder on dividing its first argument by its second argument. (This is a long-standing abuse of terminology, because in mathematics the modulus is the divisor, not the remainder.) Division by 2 yields remainder 1 from odd integers and remainder 0 from even integers. A merit of this function is that it can be applied quite generally. A previous tip (Cox 2007) if anything understated that case. Indeed, functions in Stata are often undervalued by users (Cox 2011). So, in this example, if we wanted to show 1935 1940 1945 1950, these years have remainder 0 on dividing by 5. As a detail relevant to that example, note that `graph bar` will not let you show a label of 1955, because there is no bar for 1955 to label.

A guess suggests, and experiment confirms, that for these data we have enough space to show 1935 1937 to 1953. Let's clear the value labels out of the way and just show spaces instead of the even-numbered years.

```
. label val year

. local call

. forvalues j = 2(2)20 {
  2.       local call `call´ `j´ " "
. }

. graph bar (asis) invest kstock if company == 1, over(year, relabel(`call´))
> `opts1´ `opts2´
```

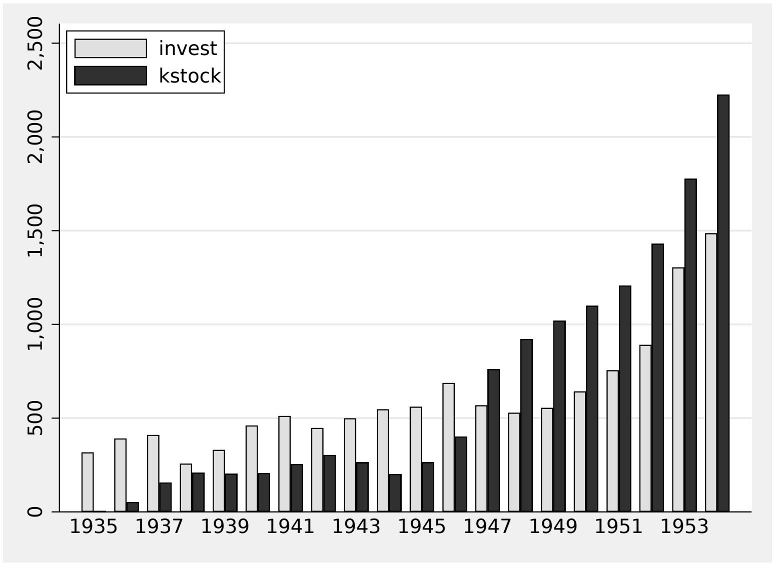

Figure 4. Showing every other year works well for these data. Strictly, the text for omitted years is a space, but the effect is as desired.

Figure 4 is in my view better than any previous figure. The same amount of text is shown on the x axis, but few readers should have difficulty imagining the omitted labels. There is not the puzzle of decoding 35 to 53 as years in their century.

2.4 Possible solution: Use twoway bar instead

You may be thinking that all of this is more messing around than you want for a very mundane problem. Is not there a simpler solution? For time series, there is, and it hinges on switching to `twoway bar`.

For bars side by side, you need to work a little at defining offset variables (Cui 2007). For two bars, we move one bar to the left and one bar to the right. We do need some small arithmetic to determine not just bar offsets but also good bar widths (which default to 1). Although we do not spell out any details, you could set up three or more bars, with the usual trade-off between the information gain in encoding several variables and the difficulty of decoding the several bars easily and effectively.

Now, your horizontal axis labels really are controllable simply and directly by the option `xlabel()`. Partly to show that we can, this final example uses axis labels every 5 years for simplicity, including the previously out-of-reach 1955:

```
. generate yearL = year - 0.15
. generate yearR = year + 0.15
. twoway bar invest yearL if company == 1, barw(0.3) fcolor(gs14) lcolor(black)
> || bar kstock yearR, barw(0.3) fcolor(gs3) lcolor(black) xla(1935(5)1955)
> `opts2´
```

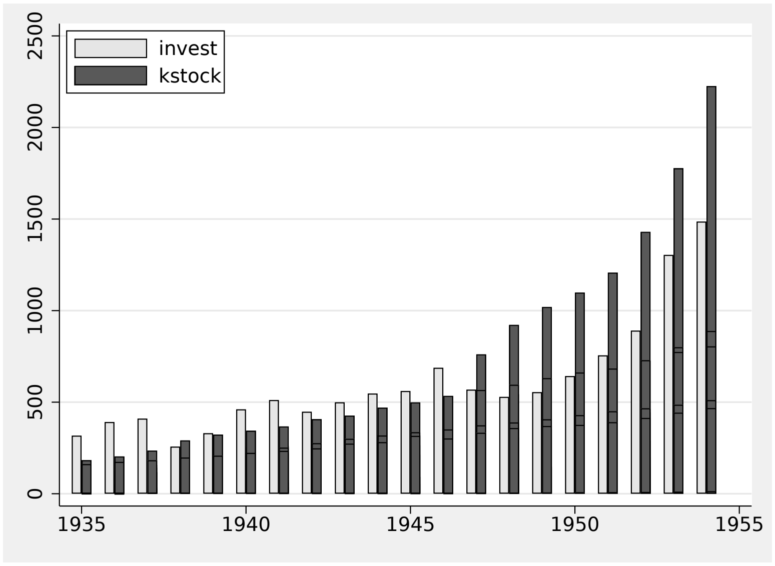

Figure 5. Just use `twoway bar` instead. You need to work at defining offset variables, but once that is done, it is downhill all the way.

Figure 5 plots one series of bars against one year variable offset left and the other series against one year variable offset right. The upshot is that `twoway` does not know what I want as x-axis title. That is fine by me, because I do not want to display an axis title like `"year"` at all. If I want or need to explain that series are plotted against year, I can do that in the text option I write in my text editor or word processor. Often enough, it will be clear from context. Your teachers were more right than wrong in urging you to explain every graph axis, but calendar year can be an honorable exception.

Many users start with `twoway` in any case for time series. If the time series has gaps or missing values, that is a really good idea. If you want line charts, unconnected or connected, it is essential.

2.5 Edit the value labels or string values

It is not relevant here, but often the solution is to slim down the existing value labels or string values. It can be prudent to save the original versions in case you want to go back for some later purpose.

References

Agresti, A. 2013. *Categorical Data Analysis*. 3rd ed. Hoboken, NJ: Wiley.

Cox, N. J. 2007. Stata tip 43: Remainders, selections, sequences, extractions: Uses of the modulus. *Stata Journal* 7: 143–145. https://doi.org/10.1177/1536867X0700700113.

———. 2010. Stata tip 85: Looping over nonintegers. *Stata Journal* 10: 160–163. https://doi.org/10.1177/1536867X1001000115.

———. 2011. Speaking Stata: Fun and fluency with functions. *Stata Journal* 11: 460–471. https://doi.org/10.1177/1536867X1101100308.

———. 2020. Speaking Stata: Loops, again and again. *Stata Journal* 20: 999–1015. https://doi.org/10.1177/1536867X20976340.

Cui, J. 2007. Stata tip 42: The overlay problem: Offset for clarity. *Stata Journal* 7: 141–142. https://doi.org/10.1177/1536867X0700700112.

Fienberg, S. E. 1980. *The Analysis of Cross-Classified Categorical Data*. 2nd ed. Cambridge, MA: MIT Press.

Lloyd, C. J. 1999. *Statistical Analysis of Categorical Data*. New York: Wiley.

Long, J. S., and J. Freese. 2014. *Regression Models for Categorical Dependent Variables Using Stata*. 3rd ed. College Station, TX: Stata Press.

Simonoff, J. S. 2003. *Analyzing Categorical Data*. New York: Springer.

Tutz, G. 2012. *Regression for Categorical Data*. Cambridge: Cambridge University Press.