The Stata Journal (2008)
8, Number 2, pp. 269–289

Speaking Stata: Between tables and graphs

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

Abstract. Table-like graphs can be interesting, useful, and even mildly innovative. This column outlines some Stata techniques for producing such graphs. `graph dot` is likely to be the most under-appreciated command among all existing commands. Using `by()` with various choices is a good way to mimic a categorical axis in many `graph` commands. When `graph bar` or `graph dot` is not flexible enough to do what you want, moving to the more flexible `twoway` is usually advisable. `labmask` and `seqvar` are introduced as new commands useful for preparing axis labels and axis positions for categorical variables. Applications of these ideas to, e.g., confidence interval plots lies ahead.

Keywords: gr0034, labmask, seqvar, tables, graphs, dot charts

1 Introduction

When reporting on data analyses, researchers may need to choose between tables and graphs. The choice may be severely constrained. An advisor, a supervisor, a reviewer, or an editor may lay down instructions about acceptable forms of reporting. Many disciplines have firm conventions on the format of tables of parameter estimates, t statistics, p-values, and the like, or on ways of graphing results. Yet often there is much room for discussion on whether tables or graphs may better serve a particular aim and a particular audience. Tufte (2001, 56) suggested that "Tables usually outperform graphics in reporting on small data sets of 20 numbers or less". Gelman, Pasarica, and Dodhia (2002) in contrast emphasized the scope for turning tables into graphs. This column focuses on the transition zone between tables and graphs by discussing some Stata techniques for graphs with table-like structure or content.

Graphs may resemble tables if axes show sets of categories rather than numeric scales. Stata's `graph` command has a formal idea of a categorical axis, which is embedded in the commands `graph dot`, `graph bar`, and `graph box`. These commands show values for a numeric variable according to the groups of one or more categorical variables. Their common structure is that a response variable is considered to be plotted on the y axis—even if that is horizontal, as with `graph hbar` or `graph hbox`—and the other axis is considered to be categorical. The idea of a categorical axis in this column is not so formal. Indeed, I will emphasize that it is easy to plot categories on a standard y or x axis using `graph twoway`. You will need to do that if the commands just mentioned do not do what you want. There are also some advantages in doing so.

Additionally, graphs may resemble tables if they contain much by way of numbers or text, rather than standard graphic ingredients such as point symbols, line segments, or shaded areas. The stem-and-leaf plots named and popularized by Tukey (1977) and recently discussed in this column by Cox (2007) are a good example. The displays used in meta-analytic studies are another; see Harris et al. (2008) for an excellent Stata-based discussion. Hybridizing graphs and tables in this way remains a fruitful source of innovations in reporting. It has often been suggested (e.g., Cox [2003]) that the distinction between tables and figures owes most to a division of labor imposed by the invention of printing. Different tasks—typesetting and preparing illustrations—became the responsibility of different professions. Modern computing makes that distinction obsolescent, if not obsolete, and creates scope for rethinking forms of reporting.

2 A first example

The Economist (2008) gave a table of data on uses of mobile phones (cell phones, if you prefer) and personal digital assistants (PDAs) by age group (see table 1).

Table 1. Use of mobile phone or PDA to do the following, percent in 2007

	18–29	30–49	50–64	65+
Send or receive text messages	85	65	38	11
Take a picture	82	64	42	22
Play a game	47	29	13	6
Play music	38	16	5	2
Record a video	34	19	8	3
Access the Internet	31	22	10	6
Send or receive email	28	21	12	6
Send or receive instant messages	26	18	11	7
Watch a video	19	11	4	2
At least one of these activities	96	85	63	36

Source: *The Economist* 2008. Nomads at last: A special report on mobile telecoms. April 12, 4.

Their source: Pew Research Center.

There is no indication in *The Economist* of sample size or method or of where interviews took place. Our focus is entirely on possible presentations of these data. You may consider that a table like this one is perfectly appropriate and adequate for such a small set of numbers with such simple structure: note in particular how uses of all kinds decrease with the age group. Despite that, we can use the data as an example to discuss alternatives.

3 Data entry

The first question is how best to get the data into Stata. The structure may seem obvious from a glance at the table: one string variable and four numeric variables. However, there are other possibilities.

Suppose you typed the various activities on the left of the table into a string variable. When asked for a display, whether table or graph, using that variable Stata would impose its own idea of tidiness and sort those values alphabetically—or, more precisely, alphanumerically. Here, as almost everywhere else, that would not help at all. Alphabetical order is only essential when the sole purpose of a table or graph is looking up individual values and it is irrelevant or confusing otherwise.

Inspection of the table shows a sensible ordering from more popular activities to less popular activities, except for a kind of total at the end. Such ordering by magnitude and appending a total or a ragbag category are common and commendable table habits (see Ehrenberg [1975] for more good advice on tables).

A good Stata answer to this difficulty is to let the activities be value labels to a numeric variable that has the order you desire. (Start with the online help for `label` if value labels are new territory for you.) I am going to add a small twist to this standard advice. You can enter the activities as a string variable and the numeric order as a numeric variable. In the next section, we will see a new command to link the two.

The numeric data also pose a choice of data structure. The table may suggest using four numeric variables, but there can be advantages in having one numeric variable and indicating the age groups by another variable, a long rather than a wide data structure. Whichever way you do it, there is a good chance that some later purpose will make a `reshape` or a `separate` a good idea; see [D] **reshape** or [D] **separate**.

Here it is fortuitous, but fortunate, that the strings `"18-29"`, `"30-49"`, `"50-64"`, and `"65+"` would always sort the way we want. That would not be true if we had an extra category such as `"3-17"`. A more general way to do it, as before, is to define a numeric variable specifying desired display order and to put the category labels in a string variable.

Typing even a small dataset like this into Stata is a little tedious and error-prone. For other problems, you may be able to read a table in directly to Stata. Or the dataset for a display may result from a reduction command such as `collapse`, `contract`, or `statsby`. For more information, see the corresponding entries in [D]. Otherwise, for data entry, it is worth mastering three Stata tricks that can cut down on the busy work.

First, recall that a variable defining integers 1 and up is obtained by using the observation numbers:

```
. generate seq = _n
```

Second, a more general tool is `egen, seq()` ([D] **egen**). If you followed the advice above of a long data structure you need either 10 blocks of 4 observations or 4 blocks

of 10 observations. Which you choose is immaterial. We will use the first. If you have not typed anything else, you would need to declare the size of the dataset:

```
. set obs 40
```

Then you could follow with

```
. egen ageorder = seq(), to(4)
. egen activityorder = seq(), block(10)
```

The `egen` function `seq()` is a little unusual. It takes no arguments, but we still need to spell out that it is a function by giving parentheses `()`. `seq()` creates integer sequences that by default start at 1 and increase. With this default the first value will be 1, the second 2, and so on. If an upper limit is specified in the `to()` option and the end of the dataset has not been reached, the sequence will restart. If a block size is specified, then values will be repeated in blocks. Thus `ageorder` will start 1 2 3 4 1 2 3 4, and `activityorder` will start 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2.

Third, if a string or numeric value is repeated in blocks, you need not type in the same values repeatedly. Enter just the first value in each block and leave subsequent entries blank, so that Stata leaves missing values in between, either numeric missing `.` or the empty string `""`. Then fill in the blanks with a single command

```
. replace whatever = whatever[_n - 1] if missing(whatever)
```

The test `if missing(`*whatever*`)` can apply to numeric or string variables. The idea is to copy the previous value if a value is missing. As `replace`, and for that matter `generate`, use the current sort order—a point made explicit by Newson (2004)—this works with blocks of missing values too. The first missing is replaced by the previous value, which then becomes nonmissing, so that it can be used in turn to overwrite the next missing value, and so forth.

A few more general words on data and display: If a novel table or graph form is being produced, life is often much simpler if the data are cut down first to just those observations and variables needed. Having to worry about data reduction or manipulation at the same time as working out the syntax may add up to too much like hard work. In some cases, it can even be easier to type text and numbers directly into a completely new dataset than to work out how to get them into new variables within the existing dataset. Naturally, if you find yourself doing such typing repeatedly, there is then much more incentive to work out how to automate the tabulation or graphics.

4 Introducing labmask

You can always define value labels explicitly and then link them to a numeric variable, so long as that numeric variable is integer-valued. However, if you already have a numeric variable *numvar* and a string variable `strvar` and you want the values of `strvar` to be the value labels of `numvar`, then another approach is necessary, or at least convenient.

To see this, note that `encode` ([D] **encode**) is of no use here unless you have already defined the labels the way you want. Conversely, if you have defined the labels, you do not need `encode`. The `encode` command is of no use here because its own defensible idea of tidiness is to sort the distinct string values that occur alphanumerically and then link them to integers 1 and up. So you might end up with value labels 1 `"aardvark"`, 2 `"bison"`, 3 `"colugo"`, or whatever. For tables and graphs, that is not usually ideal, as already emphasized.

`labmask` has been invented for this purpose and is published formally with this column. It is more general than the previous problem outline implies, as the values, or if desired the value labels, of one variable can be assigned as the value labels of another. Thus existing string values, value labels, and numeric values can all become the value labels of a numeric variable. The perhaps whimsical name `labmask` arises from the idea that the variable being assigned these value labels acquires a mask, which is what will be seen henceforth. The way `labmask` does that is at root no different from any other way of assigning value labels, but every program author needs to find a program name.

A more formal description of `labmask` is given at the end of this column. For now, just one restriction and one freedom need emphasis. The essential syntax is `labmask` *varname*`, values(`*valuesname*`)`. *valuesname* must not vary within groups defined by the distinct values of *varname* for the observations selected. That should not seem surprising, as it follows from the general idea of a value label. However, there is no rule that the same label may not be assigned to different values of *varname*. In other contexts, that would often seem an odd or even incorrect thing to do, as values and value labels usually correspond one to one. But for graphs, and even for tables, it can be useful. Thus, for example, a year variable might be coded by the party winning at election and those value labels then shown as labels on a graph axis. It is then quite natural that different years may be assigned the same text as a value label.

Assume that somehow we have string variables `activity` and `age` and numeric variables `activityorder`, `ageorder`, and `percent`. In any case, `mobile_uses.dta` released with this column gives the data in this form. Then the string values can be mapped to value labels by

```
. labmask ageorder, values(age)
. labmask activityorder, values(activity)
```

5 Dot charts

For a dataset with similar structure, researchers might consider bar or dot representations. We will focus on the latter, partly because bar graphs do not need much extra publicity. `graph dot` implements the dot charts introduced and discussed by Cleveland (1984, 1993, 1994) and recommended in several guidebooks (e.g., Helsel and Hirsch [1992]; Robbins [2005]; Lang and Secic [2006]). Dot charts should not be confused with the dot plots used to show univariate distributions, as implemented in Stata's `dotplot` command. However, both grow out of the idea of showing numeric values by point sym-

bols on one or more parallel axes. The difference is that programs like `dotplot` allow, or even enforce, binning, and stacking so that distributions can be visualized more easily.

Figure 1 is a graph reached after some trial and error. There remains plenty of scope for small and large changes.

```
. graph dot (asis) percent, over(ageorder) asyvars
> over(activityorder) legend(order(4 3 2 1) row(1))
> marker(1, ms(O) mfcolor(gs1) mlcolor(black))
> marker(2, ms(O) mfcolor(gs5) mlcolor(black))
> marker(3, ms(O) mfcolor(gs9) mlcolor(black))
> marker(4, ms(O) mfcolor(gs13) mlcolor(black))
```

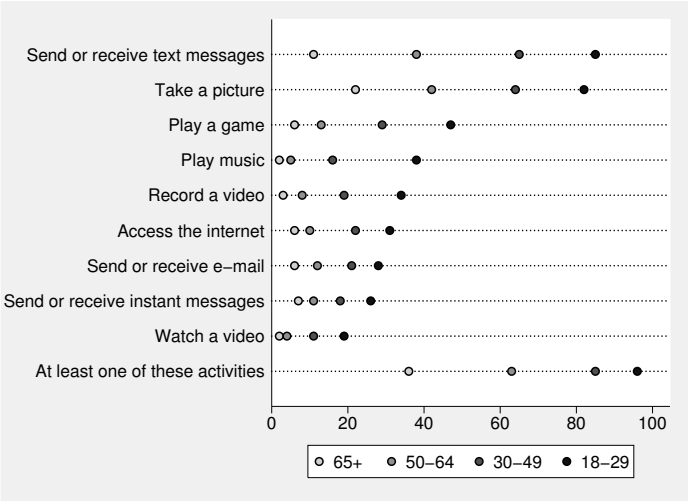

Figure 1. Dot chart of mobile phone and PDA uses, percent 2007. The `asyvars` option forces data for all age groups on to single lines.

Looking at each of the elements of this command in turn:

The data need no further reduction, so we ask for `graph dot (asis) percent`.

`over(ageorder) asyvars` gives us distinctions by `age`, except that we use `ageorder`, which now contains integers 1 and up with values of `age` as value labels, to ensure a logical order. `asyvars` gives different marker symbols: data for the categories of age are treated as if they were separate y variables, meaning responses. (With the extra option `vertical`, the y axis would be truly vertical, but the result is a mess and that path is not pursued here.)

`over(activityorder)` gives us distinctions by `activity`, except that we use `activityorder`, which contains integers 1 and up with values of `activity` as value labels, again to ensure a good order. This option defines a categorical axis, vertical in our case.

The order of options is crucial here. Options `over(activityorder) asyvars over(ageorder)` would define a different graph, although one that seems less helpful, even if tidied up.

`legend(order(4 3 2 1) row(1))` tweaks the legend away from the default. As use declines with age, reversing the order of the labels in the legend seems natural. There is enough space for making the legend a single row, or conversely no need to make it two rows, the default in this case.

The marker options are partly a matter of taste or fine judgment. We are plotting in black and white, which imposes limits that may not constrain your choices for your presentations, especially your talks. As age is an ordered (ordinal) variable, I like to use the same symbol and convey gradations by a choice of gray scales (Cox 2005). Gray scales also have, accidentally but helpfully here, associations with human aging. A black outline for marker symbols ensures that nearly white symbols remain obvious even with a white or nearly white background. Others might find the different colors here insufficient to discriminate between the different ages. Alternatives include varying the marker symbols (e.g., using squares or triangles too) or making the symbols a little larger.

A quite different possibility is to plot on a transformed scale. The most obvious example is a logit scale. For why and how, see Cox (2008).

Let us focus on one detail, the choice of `asyvars`, which forces data points for all age groups onto single lines. If we dispense with that, the syntax is now simpler. With just one kind of marker symbol, there is no need for a legend or specification of different markers. However, the number of data lines increases fourfold, so extra jiggery-pokery is needed to keep the graph legible. One choice is to switch the second `over()` to a `by()` and move to a two column display. See figure 2.

```
. graph dot (asis) percent, over(ageorder)
> by(activityorder, note("") compact cols(2))
```

(*Continued on next page*)

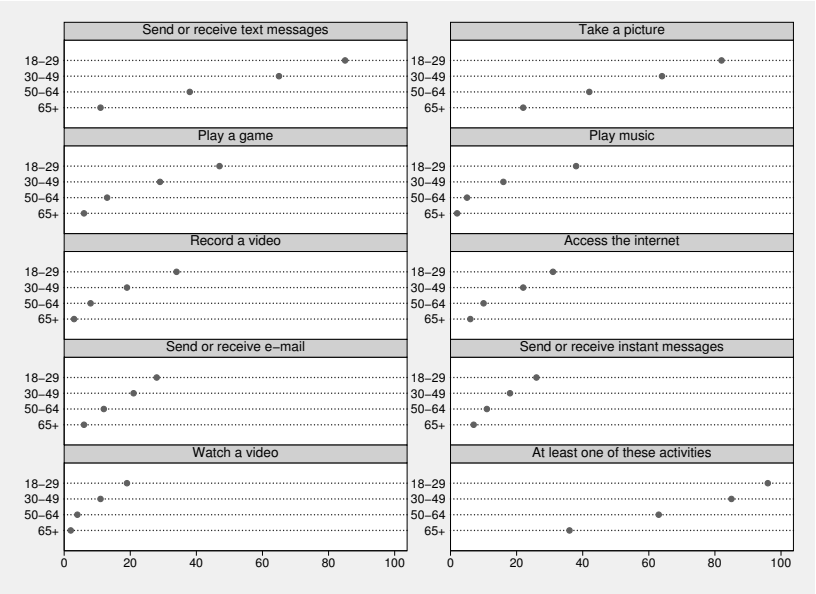

Figure 2. Dot chart of mobile phone and PDA uses, percent 2007. Separate displays for each age group and a switch to `by()` illustrate some of the trade-offs in graph design.

Another choice is a little more radical. If we force the titles of the `by()` graphs one ring further out, then a different use of space is shown. The incantations in `subtitle()` are all important: leaving any out has consequences ranging from the awkward to the grotesque. See figure 3.

```
. graph dot (asis) percent, over(ageorder)
> by(activityorder, col(2) note("") compact)
> subtitle(, pos(9) ring(1) nobexpand bcolor(none) placement(e))
```

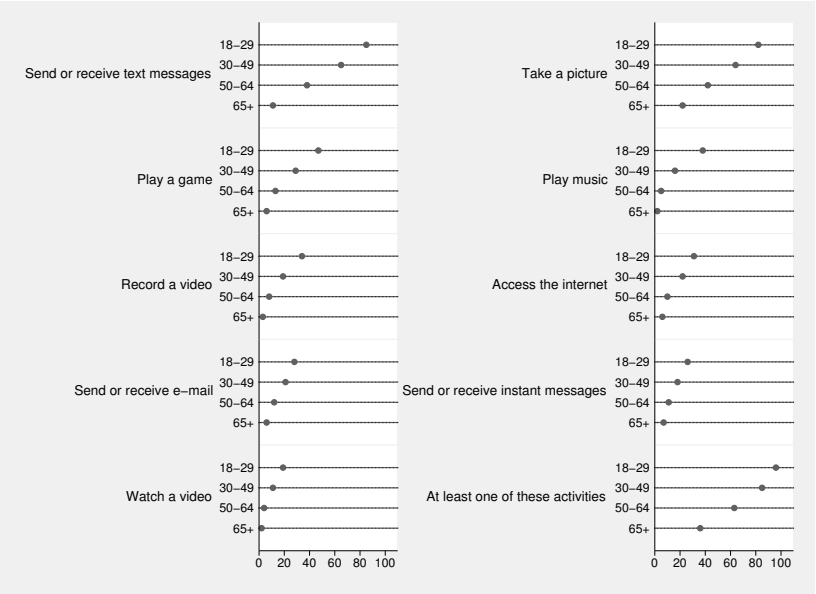

Figure 3. Dot chart of mobile phone and PDA uses, percent 2007. Separate displays for each age group and a switch to `by()`, but now putting graph titles one ring further out.

Yet further tuning clearly remains possible. Some of the activity names could be shortened without much loss, yielding more space for the data region. You may want to experiment. A more crucial detail to carry forward, however, is a realization that you can mimic a categorical axis with suitable use of `by()`. This will be especially useful whenever you need to turn to `twoway` graphs to achieve effects impossible with `graph dot` or its relatives.

6 Why dot charts?

Despite a history about a quarter century long—and no doubt a diligent search would turn up precursors—dot charts have yet to make much of a dent on the popularity of bar charts. I have asked groups of students to choose between bar and dot charts of the same data, and bar charts always win overwhelmingly, but no reason is ever given that does not boil down in one way or another to greater familiarity. My students, like so many others, have been exposed to bar charts from an early age and feel comfortable with them. But dot charts nevertheless have several distinct advantages over bar charts.

Dot charts typically take up less space than bar charts, or at least less ink. The use of space is especially important if you need to show several bars that logically cannot be stacked, as they are not distinct fractions of the same total. Such bars cannot usually be superimposed without some occluding others, so that they must be juxtaposed, taking up more space. In contrast, it is easy to plot several point symbols on the same axis with good use of space and less risk of occlusion.

A bar must have a base, usually zero: clearly, what bar charts show are values encoded by the heights or lengths of bars measured from their bases. A problem arises when values do not vary much, as differences which may be interesting or important are hard to discern. The difference between 102% and 104% may often be a big deal, scientifically or practically. Sometimes a simple change of base is the solution, but otherwise, the need to show a base must be traded off against a temptation to truncate the bars. At worst, readers may struggle to interpret the resulting bar charts correctly, or designers are suspected or accused of misleading graphics. The whole issue is avoided completely by plotting with point symbols. The base of the scale may then even be omitted.

The issue is well illustrated by temperature scales. Weather and climate are often discussed in terms of either Fahrenheit temperatures (in the United States and a few other countries) or Celsius temperatures (everywhere else). Both are interval scales, not ratio scales, meaning that their zero points are arbitrary and ratios make no sense. Even with Celsius, in which zero degrees at least has a physical meaning as the freezing point of water, bar charts are a dubious choice, whether the observed temperatures include zero or negative values, as any choice of base arbitrarily privileges a point on the scale. Point symbols on a numeric scale are not subject to such an objection.

Some readers may be wondering about pie charts. Pie charts are not an option for our example data, as the categories are not mutually exclusive—unless the prospect of 40 pies is appealing. I do not want to add to the criticisms of pie charts by (for example) Tufte (2001) and Cleveland (1994), except to refer to a very well-rounded history and some spirited defense of the form by Spence (2005).

7 Moving to twoway

Despite rafts of options, `graph dot` and its relatives, like any other graph commands, boast only a limited repertoire. There will come a time when you want to do something similar, but nevertheless beyond what is possible with those commands. You will then usually find that moving to `twoway` is the way to go.

First, see that we can do a good job at mimicking `graph dot` with `twoway dot`. See figure 4.

```
. twoway dot percent ageorder,  horizontal
> by(activityorder, col(2) note("") compact)
> subtitle(, pos(9) ring(1) nobexpand bcolor(none) placement(e))
> ysc(reverse) yla(, tlength(0) ang(h) valuelabels) xla(0(20)100)
> xtitle("") ytitle("")
```

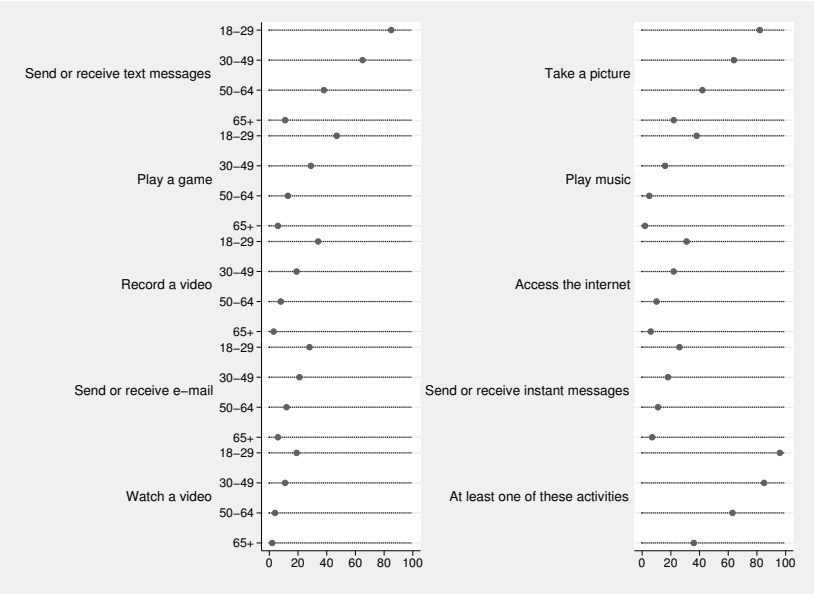

Figure 4. Dot chart of mobile phone and PDA uses, percent 2007. Separate displays for each age group and a switch to `by()`, but now putting graph titles one ring further out. This was obtained with `twoway dot`, not `graph dot`.

We have to work a little to get something acceptable, by insisting on particular choices for axis labels and scales, but no new programming is needed. Axis ticks are natural for numeric scales, but for categorical scales they are too busy, if not misleading, so we should remove them. One subtle, indeed arcane, detail is that with `by()` in operation using `tlength(0)` rather than `noticks` is the way to suppress axis ticks.

The official online help says: "`twoway dot` is of little, if any use. We cannot think of a use for it, but perhaps someday, somewhere, someone will. We have nothing against the dot plot used with categorical data—see [G] **graph dot** for a useful command—but using the dot plot in a twoway context would be bizarre. It is nonetheless included for logical completeness."

Just being able to mimic `graph dot` is not especially useful and is not enough to overturn such skepticism. But being able to do something else with `twoway dot` would be more interesting.

One idea from looking at our data table is to notice very simply that our data come as rounded percentages. (If they did not, we could always round them ourselves. Even if the percentages were all very small we could consider a change of scale to, say, per thousand or per million.) This raises the possibility of letting the digits of each value serve as marker labels. All that is needed is some standard `twoway` technique. See figure 5.

```
. twoway dot percent ageorder, horizontal
> by(activityorder, col(2) note("") compact)
> subtitle(, pos(9) ring(1) nobexpand bcolor(none) placement(e))
> ysc(reverse) yla(, tlength(0) ang(h) valuelabels) xsc(r(0 100)) xla(none)
> xtitle("") ytitle("")
> ms(none) mla(percent) mlabpos(0) mlabsize(*1.2) dcolor(bg)
```

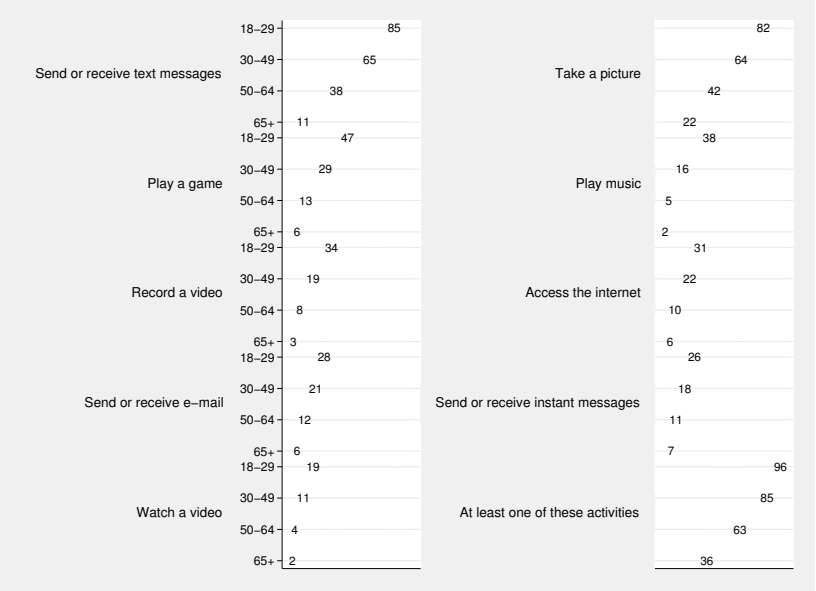

Figure 5. Dot chart of mobile phone and PDA uses, percent 2007. Numeric values as marker labels replace point symbols. This was obtained with `twoway dot`, not `graph dot`.

We suppress the marker symbols and use the numeric values of `percent` as marker labels in their place, in the manner used to produce stem-and-leaf plots and variants (Cox 2007). The x-axis labels are now redundant, but we still impose an axis range from 0 to 100 as context. The dots that come with `twoway dot` are a little obtrusive: one way to suppress them is to draw them in background color.

As always, further experiment remains possible. Two paths among others are indicated now. One is to see if `col(1)` yields good results, possibly together with a change in graph size. The other is to note that the end of all our exploring with this example will be to arrive back at what will be a very familiar place to most readers, namely, `scatter`. We can just revise, and indeed simplify, this last graph command to be a `scatter` command. The resulting graph is not shown here, as it is identical to figure 5.

```
. scatter ageorder percent,
> by(activityorder, col(2) note("") compact)
> subtitle(, pos(9) ring(1) nobexpand bcolor(none) placement(e))
> ysc(reverse) yla(, tlength(0) ang(h) valuelabels) xsc(r(0 100)) xla(none)
> xtitle("") ytitle("") ms(none) mla(percent) mlabpos(0) mlabsize(*1.2)
```

You may think the online help remains correct after all.

8 A second example

The following data (table 2) quantify home access to the Internet, again as a percentage, for different groups of people in Sweden in 2002.

Table 2. Home access to the Internet, as a percentage, for different groups of people in Sweden in 2002

Men	66.7
Women	60.3
16–24	75.5
25–34	75.0
35–44	80.0
45–54	75.4
55–64	59.9
65–74	29.8
75–84	10.3
Labourers	49.9
Lower white collar	60.8
Managers and officials	83.5
Entrepreneurs	66.3
Farmers	26.8
Old-age pensioners	22.1

Source: Statistiska centralbyrån. 2003. *Sweden in figures/Sverige i siffror 2004.* 52.

The main new twist in this example is that we have three separate little tables. The first, in particular, might seem too trivial to deserve graphing. Many researchers might feel the same way about the other two. But put the tables together and we have the potential for a useful and informative display.

We need a simple way to get what we envisage as a simple graphical display, the only slightly unusual detail being the gaps demarcating the division into separate tables. From the discussion so far, a strategy should be evident:

1. Define a variable containing integers to give a categorical axis. The integers will define axis positions.
2. Define a string variable containing category labels, "Men" and so forth.
3. Use `labmask` to link the two.

4. Define a variable containing the percentages.

5. Use `twoway` because it offers the most flexibility.

6. Underline a principle tacit so far: The percentages are a response or outcome variable, conventionally suggesting their display on the y axis. But what guides axis choice here is a more awkward fact. Once you have more than a few category labels, or have some labels that are rather long, placing categories on the x axis becomes problematic. Either you run out of space, or you are pushed into devices such as reducing text size, alternating label positions, or placing text at an angle to the axis. Even labels such as "Lower white collar" and "Managers and officials", not especially long by many standards, would pose difficulties. Horizontal display of categorical labels is, in this example and indeed in most others, easily the more congenial and convenient choice.

We can use a new tool to make task 1 a little easier, a new command `seqvar` introduced in the next section.

9 Introducing seqvar

`seqvar` is for assigning integer *numlists* to variables. A *numlist* in Stata is a list of numbers, except that extra notation allows concise specification. Thus 1/4 and 10(10)50 are *numlist*s, the first specifying the integers 1 to 4 and the second specifying the integers 10 20 30 40 50. See the online help for `numlist` for more details.

A more formal description of `seqvar` is given at the end of this column. The key idea is that `seqvar` unpacks a supplied *numlist* and assigns it to a variable *varname*, the first value in the *numlist* becoming *varname*`[1]`, the second *varname*`[2]`, and so on.

`seqvar` is designed for one main purpose, under discussion here: easy definition of numeric axis positions as part of setting up what is, in essence, a categorical axis for a graph to be shown using `graph twoway`. That intent does not rule out other uses, but it is likely that other purposes will be better served by use of `egen` functions or direct use of `_n` and/or `_N`.

The motivation for restriction to integers may also be apparent. We want to be able to assign value labels to the values produced. While other purposes may require noninteger, their calculation often entails small precision puzzles arising from the fact that computers work in binary, not decimal (Cox 2006; Gould 2006). Here the programmer wishes to protect himself from a predictable class of problem reports.

The use of `seqvar` may create as many problems as it solves if your data are not in exactly the sort order you need, or if your data include observations that are not to be included in a graph.

10 Exploiting twoway once more

The data for the second example can be found in the file `swedish_internet.dta`. We want two gaps on the y axis and so we use `seqvar` to specify a *numlist* with such gaps as the values of a new variable `axis`. We then use `labmask` to assign the values of the string variable `text` as the value labels of `axis`.

```
. seqvar axis = 1/2 4/10 12/17
. labmask axis, values(text)
```

The first graph with this data is like a dot chart, but is produced using `scatter`. See figure 6.

```
. scatter axis access,
> yla(1/2 4/10 12/17, valuelabels ang(h) noticks grid glc(gs12))
> ysc(reverse) plotregion(style(outline))
```

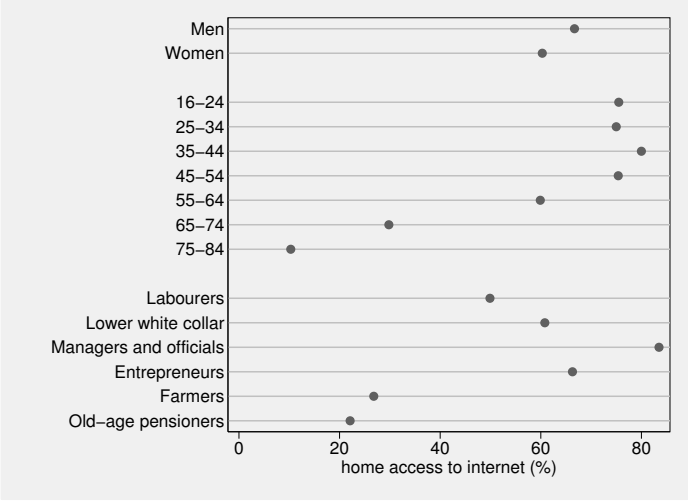

Figure 6. Dot chart showing access to the Internet for various groups of people in Sweden. This was made by `scatter`, not `graph dot`.

As the axis is categorical, a reversed scale and omission of ticks are natural. Horizontal axis labels are essential for legibility. I added a grid and an outline to the plot region: those are matters of taste.

If we want finer control of gap sizes, that means using larger integers. The axis positions above have gaps between groups of 1, which is equal to gaps within groups of 1. Suppose you wanted the gaps between groups to be 50% larger than the gaps within groups. Then specifying 10 20 35(10)95 110(10)160 is one of several ways to do that. The numerical axis positions are never explicit, as value labels are always shown instead, so only their relative values matter.

You may find yourself changing the values of a variable like `axis` in this way as you fine-tune the layout of the graph or make small mistakes in working out the sequence. If so, you will need to repeat any application of `labmask`. This is easily done by just reissuing the previous `labmask` command and is still easier than typing out a new `label define` command. Exactly the same applies to any changes of the category labels, such as if you decide to shorten some long labels to give more space to the data region.

Once it is seen how to use `twoway` to display data with this structure, all kinds of variation in graph form become possible using one or more members of the `twoway` family. I restrain myself to three examples.

Horizontal drop lines are used in figure 7. The y-axis grid is suppressed and `recast(dropline) horizontal` added.

```
. scatter access axis, yla(1/2 4/10 12/17, valuelabels ang(h) noticks nogrid)
> ysc(reverse) plotregion(style(outline)) recast(dropline) horizontal
```

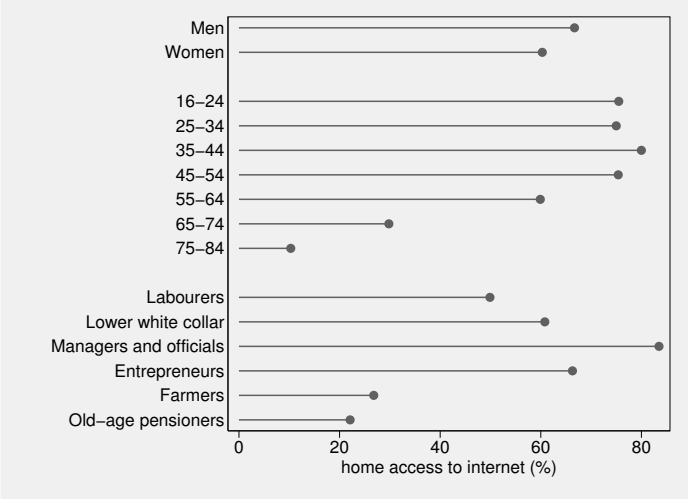

Figure 7. Horizontal drop lines are used to convey magnitude.

Horizontal bars are used in figure 8. Again, the y-axis grid is suppressed, and `recast(bar) horizontal` is added. I prefer not to have bars touching whenever the scale is categorical, so I specified `barw(0.6)`.

```
. scatter access axis, yla(1/2 4/10 12/17, valuelabels ang(h) noticks nogrid)
> ysc(reverse) plotregion(style(outline)) recast(bar) barw(0.6) horizontal
```

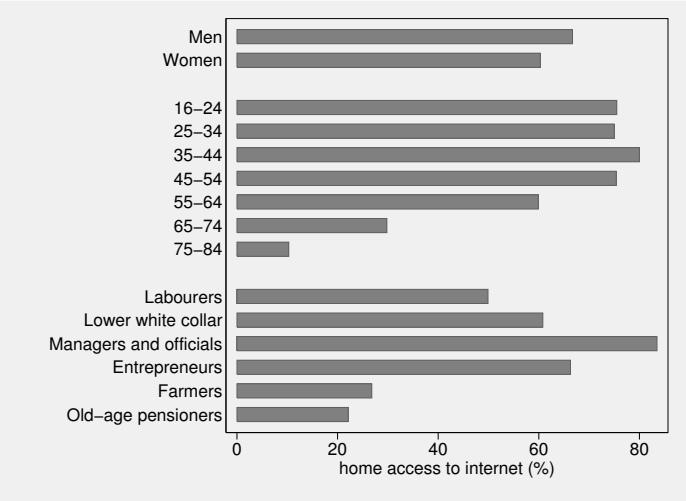

Figure 8. Horizontal bars are used to convey magnitude.

Labels indicating values are added at the ends of bars in figure 9. That requires an overlay of another scatterplot with marker labels and no visible marker symbols. The default position of the marker labels is to the right of the marker symbols that are not visible: that position corresponds to just beyond the top of each bar. Other positions would be equally easy to arrange. For example, labels at the base of each bar would be obtained by generating a variable that was identically 0 and using that for the horizontal coordinate in the second `scatter` command. Given the marker labels, x-axis labels are redundant but we still impose an axis range from 0 to 100 as context.

```
. scatter access axis, yla(1/2 4/10 12/17, valuelabels ang(h) noticks nogrid)
> ysc(reverse) plotregion(style(outline)) recast(bar) barw(0.6) base(0) horizontal
> || scatter axis access, ysc(reverse) ms(none) mla(access)
> legend(off) xla(none) xsc(r(0 100) titlegap(*6))
```

(*Continued on next page*)

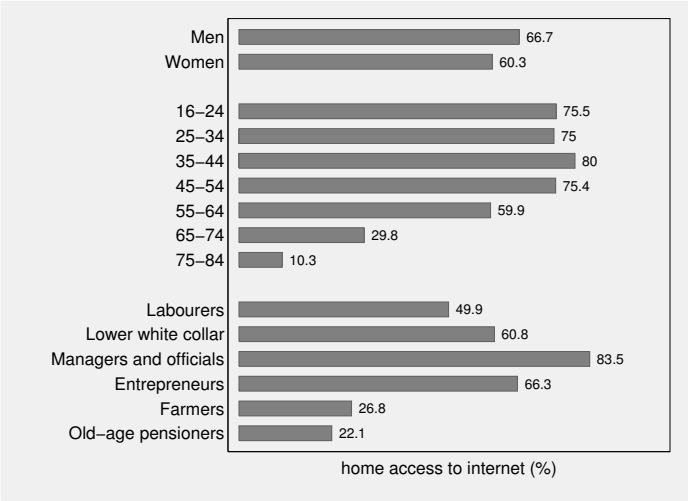

Figure 9. Text labels are added to show magnitude numerically, so that the axis labels can be suppressed.

Finally, `graph dot` can also produce good graphs for this problem. Set up a grouping variable:

```
. gen group = cond(_n < 3, 1, cond(_n < 10, 2, 3))
```

Then use it in an `over()` option. Here we suppress the labels 1 2 3 that would be shown by default. Applying `relabel()` with explicit empty labels will not do that, but `labsize(zero)` does the trick, echoing the `tlength(0)` device used earlier. An alternative would be to add annotations such as "Sex" and "Age group", but they do not seem necessary. The result is similar enough to figure 6 not to be shown here.

```
. graph dot (asis) access, over(axis) over(group, label(labsize(zero))) nofill
```

11 Details on new commands

11.1 labmask

Syntax

`labmask` *varname* [*if*] [*in*], `values(`*valuesname*`)` [`lblname(`*lblname*`) decode`]

Description

`labmask` assigns the values (or optionally the value labels) of one variable *valuesname* as the value labels of another variable *varname*. Any existing value labels will be over-

written. The idea behind the program name is that henceforth the face that *varname* presents will not be its own, but a mask borrowed from *valuesname*. Thus, for example, a year variable might be coded by party winning at election and those value labels then shown as labels on a graph axis.

varname must take on integer values for the observations selected. *valuesname* must not vary within groups defined by the distinct values of *varname* for the observations selected. However, there is no rule that states that the same label may not be assigned to different values of *varname*.

Options

`values(`*valuesname*`)` specifies a variable whose values (by default) or value labels (optionally) will be used as the value labels of *varname*. `values()` is required.

`lblname(`*lblname*`)` specifies that the value labels to be defined will have the label name *lblname*. The default is that they will have the same name as *varname*. Any existing value labels for the same values will be overwritten in either case.

`decode` specifies that the value labels of *valuesname* should be used as the value labels of *varname*. The default is to use the values of *valuesname*.

11.2 seqvar

Syntax

`seqvar` *varname* `=` *numlist* [`, replace`]

Description

`seqvar` assigns the values in an integer *numlist* to the successive observations (observation number 1 and up) of a variable *varname*. If the *numlist* contains fewer values than the number of observations in the dataset, the remaining values will be defined as missing. If the *numlist* contains more values than the number of observations, only as many values as can be assigned will be used and the rest will be ignored. The *numlist* must include integers only and may not include missing values.

Option

`replace` specifies that if *varname* exists then all its values will be replaced. This option is essential for overwriting an existing variable. Nonmissing values will be replaced by missing values whenever the *numlist* does not specify sufficient integers to be assigned.

12 Conclusion

Table-like graphs can be interesting, useful, and even mildly innovative. This column has outlined some Stata techniques for producing such graphs.

If one or more variables, often controlling or contextual, are categorical, then `graph box`, `graph bar`, `graph pie`, and `graph dot` are possible commands. Of these commands, `graph dot` is likely to be the most under-appreciated.

Using `by()` with various choices is a good way to mimic a categorical axis. That applies to many `graph` commands.

When the `bar` or `dot` commands are not flexible enough to do what you want, moving to `twoway` is usually the answer. `twoway` is enormously flexible.

`labmask` is a new tool for the situation in which you want the values, or the value labels, of one variable to be the value labels of another variable. It thus automates the definition and attachment of a set of value labels.

`seqvar` is a new tool for assigning integers to a variable. Used with `labmask`, it allows you to produce categorical axes with gaps to indicate groupings.

The underlying theme is hardly exhausted by this column. Plots of confidence intervals or other intervals indicating uncertainty often benefit from similar devices, but that is a large enough story to merit separate treatment. Overlaying of two or more graphs looms much larger in that story than it does in this one.

13 Acknowledgments

I have benefited from questions, suggestions, and interest in this territory from Kit Baum, Ronán Conroy, Friedrich Huebler, Hiroshi Maeda, and Austin Nichols. Vince Wiggins told me about the `tlength(0)` trick. Omitting ticks by setting their length to 0 is all too obvious in retrospect.

14 References

Cleveland, W. S. 1984. Graphical methods for data presentation: Full scalebreaks, dot charts, and multibased logging. *American Statistician* 38: 270–280.

———. 1993. *Visualizing Data*. Summit, NJ: Hobart Press.

———. 1994. *The Elements of Graphing Data*. Summit, NJ: Hobart Press.

Cox, N. J. 2003. Speaking Stata: Problems with tables, Part 1. *Stata Journal* 3: 309–324.

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

———. 2006. Stata tip 33: Sweet sixteen: Hexadecimal formats and precision problems. *Stata Journal* 6: 282–283.

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

———. 2008. Stata tip 59: Plotting on any transformed scale. *Stata Journal* 8: 142–145.

Ehrenberg, A. S. C. 1975. *Data Reduction*. London: Wiley.

Gelman, A., C. Pasarica, and R. Dodhia. 2002. Let's practice what we preach: Turning tables into graphs. *American Statistician* 56: 121–130.

Gould, W. 2006. Mata Matters: Precision. *Stata Journal* 6: 550–560.

Harris, R. J., M. J. Bradburn, J. J. Deeks, R. M. Harbord, D. G. Altman, and J. A. C. Sterne. 2008. metan: fixed- and random-effects meta-analysis. *Stata Journal* 8: 1–28.

Helsel, D. R., and R. M. Hirsch. 1992. *Statistical Methods in Water Resources*. Amsterdam: Elsevier. http://pubs.usgs.gov/twri/twri4a3/.

Lang, T., and M. Secic. 2006. *How to Report Statistics in Medicine: Annotated Guidelines for Authors, Editors, and Reviewers*. Philadelphia: American College of Physicians.

Newson, R. 2004. Stata tip 13: generate and replace use the current sort order. *Stata Journal* 4: 484–485.

Robbins, N. M. 2005. *Creating More Effective Graphs*. Hoboken, NJ: Wiley.

Spence, I. 2005. No humble pie: The origins and usage of a statistical chart. *Journal of Educational and Behavioral Statistics* 30: 353–368.

Statistiska centralbyrån. 2003. *Sweden in Figures/Sverige i Siffror 2004*. Örebro: Statistiska centralbyrån.

The Economist. 2008. Nomads at last: A special report on mobile telecoms. April 12, 4.

Tufte, E. R. 2001. *The Visual Display of Quantitative Information*. 2nd ed. Cheshire, CT: Graphics Press.

Tukey, J. W. 1977. *Exploratory Data Analysis*. Reading, MA: Addison–Wesley.

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 wrote several inserts in the *Stata Technical Bulletin* and is an editor of the *Stata Journal*.