Stata Walkthrough: Graphics.

For this walkthrough, you will need to download the database on US workers in the March 2005 CPS *and* the one on cod catches. Start with the CPS (If you have an error loading the database, try typing the command **`set mem 100m`** to increase the memory used by Stata.)

Highest educational attainment is a categorical variable, described in six levels: less than high school, some high school, high school, technical or associate's degree or some college, college degree, and professional or graduate degree. These are captured in the variable **`educ`** (which takes values 1 through 6) or equivalently in the dummy variables **`educ1`** through **`educ6`**.

The simplest way to illustrate these graphically is through a bar graph or pie graph. The command for the first of these is **`graph bar educ1-educ6`** (the dash indicates variables **`educ1`** *through* **`educ6`**, as they are listed in the variable window.) Try both of these commands.

One thing you will notice is that the bar graph returns decimal values. It is telling you the fraction of the sample in each category. This is because the default option for a bar graph is to report the mean of the variable (and the mean of a dummy variable is the fraction in that group). If you want to know the total number in each category, try

```
graph bar (sum) educ1-educ6
```

You can also specify other descriptive statistics in parentheses after the word bar; check the help files.

To construct a pie chart, type

```
graph pie educ1-educ6
```

This produces a colorful, uninformative pie chart. You can add in additional information:

```
graph pie educ1-educ6, plabel(_all percent)
```

will superimpose the fraction in each category;

```
graph pie educ1-educ6, plabel(_all sum)
```

will tell you the absolute numbers in each group. (Read the help files for other information that you can superimpose.) Finally, rather than creating a bunch of dummy variables **educ1** through **educ6**, you can also the categorical variable **educ** to produce a pie chart. In this case, you type:

```
graph pie, over(educ)
```

The output is the same, and you can superimpose information in the same way. However, you know that I don't particularly care for pie charts, since I find that they rarely *invite comparisons*, which is one principle for good graphics. We can create a bunch of side-by-side pie charts using the command:

```
graph pie educ1-educ6 if race<5, by(race)
```

The option **by** signals "side-*by*-side". Also, note that I've added a qualifier to this statement, to include people with race less than 5. In the CPS, the race codes are 1 for Caucasian, 2 for African, 3 for Native American, and 4 for Asian; codes of 5 and above are for smaller minorities (mostly multiracial people, but also Pacific islanders). I just wanted to restrict our attention to the big four groups.

You can also create bar graphs to tell some summary statistic across different groups. If you want to display the mean salaries of workers with different educational levels, you would type:

```
graph bar (mean) salary, over(educ)
```

You could replace **mean** with **median**, if that's the statistic you're interested in. What's the difference between **over** and **by**? Try typing:

```
graph bar (mean) salary, by(educ)
```

This generates side-by-side bar graphs, displaying a single statistic in each: the mean of the variable. You'd never do something as simple as this, but I wanted to show you the difference between these commands. However, you can do some other cool stuff. Try this one:

```
graph bar (mean) salary if race<5, over(educ) by(race)
```

This displays graphics with the returns to education for different racial groups. The labels are a bit cluttered, but we'll deal with that later.

Anyhow, we've started describing bivariate data (and even trivariate, in that last case!). As we know, the most common ways to describe a joint distribution are with a time-series graph or a scatterplot. This database doesn't have a time series, so we'll have to switch. Type **`clear`** to unload the CPS data, and then open the one on cod.

A time series is simple. Type

```
sort year
```

to ensure that your data is in the correct order (otherwise, you'll get funky graphs) and then

```
graph twoway line cod year
```

There's your time series graph, showing how cod catches have evolved over time. Again, I dislike graphs that don't *invite comparison*, so I'd recommend showing at least two related variables over time. You can do this easily by giving a list of the variables you want plotted; Stata always interprets the last one to be the x-axis. Try:

```
graph twoway line cod canada year
```

Now you can see how Canadian catches compare to the rest of the world—during the 1980s, Canada was virtually monopolizing the trade.

Now let's move on to scatterplots. Let's suggest the relationship between worldwide cod catches and Canadian cod catches in a scatterplot. There are two syntaxes that gives us exactly the same result:

```
scatter cod canada
```

```
graph twoway scatter cod canada
```

Generally, the command for graphic bivariate data is **`graph twoway`**, followed by the type of graph you want—**`scatter`**, in this case—followed by the list of variables. Because scatterplots are so common, we're allowed to use the first syntax as a shortcut. However, there are times where the lengthier version is useful. One would be when we want to superimpose two graphics. Here's an example:

```
graph twoway (scatter cod canada) (lfit cod canada)
```

`lfit` is a special type of graph that produces the best *l*inear *fit* of the data, exactly the same line that I asked you to produce in a "perfect scatterplot". Another ingredient of the perfect scatterplot was (possibly) labeling the datapoints, which can also be done in Stata:

```
graph twoway (scatter cod canada, mlabel(year)) (lfit
cod canada)
```

This now tells us which datapoint is from which year. Perfect! The only thing that's missing are box-and-whisker plots on the axes, which is a bit more challenging. Let's deal with some other things first.

Some of these graphs have had cluttered labels, and some have been missing labels entirely. By default, Stata will label your axes with the labels attached to your variables, if there is one. Type **`describe`** to see how each variable is labeled.

You might want to add labels or change them. One way to do this is to relabel the variables themselves (the other is to change the labels in a particular graphic). To do this, you might type:

```
label variable canada "Canadian fish"
```

```
label variable cod "Worldwide fish"
```

Now when you type **scatter cod canada**, you'll see your new labels on the axes of the graphic. However, if you want to override these labels for a particular graphic, you can instruct Stata to add titles to the axes:

```
scatter cod canada, xtitle("Fish from Canada")
ytitle("Fish worldwide")
```

You can also play with the numbers that are labeled on the particular axes. By default (in this graphic—not generally), Stata wants to put a tick mark and a label at each multiple of 200 on the vertical axis, and at each multiple of 50 on the horizontal axis. That's fine, but let's say that you wanted a tick mark at every multiple of 50 on the vertical axis. You would type:

```
scatter cod canada, ytick(0 (50) 800)
```

This places a tick mark at every multiple of 50 from 0 to 800 on the y-axis. You could add something similar for the x-axis:

```
scatter cod canada, ytick(0 (50) 800) xtick(0 (25) 250)
```

You can also tell Stata which numbers you want labeled; it's still labeling only every multiple of 200 on the vertical axis. We could change that to multiples of 100 by simply entering:

```
scatter cod canada, ytick(0 (50) 800) ylabel(0 (100) 800)
```

You can use any or all of the commands **xlabel**, **ylabel**, **xtick**, and **ytick** in a graphic, depending on your needs.

The last thing we're going to do is to combine several graphics into a single image, in my "perfect scatterplot." The first step is to create a boxplot each axis:

```
graph box canada, fxsize(20) saving(leftgraph)

graph hbox cod, fysize(20)  saving(bottomgraph)
```

There are two novelties in each case. First, we restrict the size of the first graph to be only 20% of the normal x-dimension, and the second to be only 20% of the ydimension. Next, we save the output of each to a Stata file in your current directory. We can access these later, which is exactly what we want to do. (Note: if you screw up a graph the first time but you have saved your bad results to a file, you will have to type something like **`saving(leftgraph, replace)`** to write over the original results.)

Next, we want to create the main graph. We will type:

```
graph twoway (scatter cod canada) (lfit cod canada),
saving(maingraph) ylabel(none) ytitle("") xlabel(none)
xtitle("") legend(off)
```

This creates two graphs in the same space: one which is the scatterplot of the two variables, and the other which gives the best explanation of the relationship. We have told Stata to save the results to a file, and we have also suppressed the titles on the x-axis and y-axis, as well as the numerical labels. This makes the graph look silly on its own, but it will look better in combination with the others. We've also told it not to print a legend on the graph, which will make the center image the same size as the other two graphs. (Stata usually adds a legend, explaining that the dots are actual values, and the line is the fitted relationship.) We really should add this in at a later point, but we're not going to bother.

Finally, to combine the three graphics in a single image, we need to type:

```
graph combine leftgraph.gph maingraph.gph
bottomgraph.gph, cols(2) holes(3)
```

(Although generally Stata doesn't care about file extensions, you do need to specify **`.gph`** here.) The **`graph combine`** command tells Stata to place all of the following graphs into a single one, arranged in the number of columns we tell it (two, in this case). What's that **`hole(3)`** option? Generally, the output of a command like **`graph combine a b c d e f, cols(2)`** would look like this:

Graph A	Graph B

Graph C	Graph D
Graph E	Graph F

However, if we want to leave one of the spots blank—the third one, for example—we tell Stata to put a "hole" in that spot. So **`graph combine a b c d e, cols(2) hole(3)`** would look like:

Graph A	Graph B
--	Graph C
Graph D	Graph E

In this case, we want to place the horizontal box-and-whisker plot immediately below the main graph, so we needed to tell Stata to place a hole in the third entry.

Anyhow, now we're done: we've created a very informative, super sophisticated graphic in Stata. It combines a simple description of the two variables, a plot of their joint distribution, and a line suggesting the relationship between the two.