Spring Stata Workshops
Data Visualization
University of Kansas

Contents

1 Introduction

An Introduction to Stata
Stata Workshop 2
University of Kansas

Goal: to learn about data visualization with Stata.
In this workshop we will focus on different types of visualizations and editing those graphs in Stata.

This workshop will use the World Bank World Development Indicators data again. A reminder that the steps to get that data into Stata are:

1. Open Stata on the KU virtual lab or use software installed on your laptop: Virtual Desktop Link.

2. In the command line Stata, run the command: ssc install wbopendata if you have not already installed the API before.

The indicators that we want to use in this workshop can be called with the wbopendata command as follows:

wbopendata, indicator(NY.GDP.MKTP.KN; NY.GDP.MKTP.KD.ZG; NV.AGR.TOTL.ZS; NV.IND.MANF.ZS; NV.SRV.TOTL.ZS; SP.POP.TOTL; SE.XPD.TOTL.GD.ZS; SL.AGR.EMPL.ZS; SL.SRV.EMPL.ZS; NV.IND.TOTL.ZS; SL.IND.EMPL.ZS) clear long

2 General features

2.1 Working Directory

The Working Directory
It is always important to know where the files that you are using are saved on the computer. This is so both you and Stata can access the correct files. Let's see how this would work for this problem.
For a data analysis project, your file system should be something similar to the following:

- Data
- Working Data
- Do Files
- Literature
- Output (the LaTeX file should be located in output for access to graphs)

Save the data for this project in a working directory and set the folder containing data as the working directory. The command **cd** will 'change directory' in stata, and you need to add the file path after that command.

Typing **pwd** into the command line well tell you the current directory and can be used to verify that you are in the correct folder.

2.2 Do Files

"Do-files" and comments

All the commands you enter into the Command Line for the lab can (and should) be put into a "do-file" to allow replication and access at a later date. You should save a "do-file" in the folder for do files and number them in the order that you would use them to transform and analyze the data.

You should heavily comment your do files to explain what you are doing and record the process that you follow.

3 Getting Started with Visualization

3.1 Bar Charts

A basic bar chart can be created in Stata using the following command: ***graph bar (mean) numeric_var, over(cat_var)***. In this command, the y-axis is numeric, and the x-axis is categorical.[1]

The first consideration with a bar chart is the statistic we want to display. By default, if we only specify the categorical variable, the bar chart will provide the percentages of observations for each level of the categorical variable, *catvar1*. This would look something like: **graph bar, over(catvar1)**. If we specify a numerical variable, the default statistic calculated will be the mean of that variable. This would look like **graph bar v1**.

What if we want to use a different statistic? We can select the statistic of interest by typing it in parentheses prior to the first numerical variable. Since the default statistic is the mean, **graph bar numeric_var, over(cat_var)** is equivalent to **graph bar (mean) numeric_var, over(cat_var)**. Common alternative statistics include *count*, *percent*, *max*, *min*, or *sd*, among others.[2]

[1] For a more detailed explanation, visit `https://www.stata.com/manuals/g-2graphbar.pdf`.

[2] For the full list of available statistics, visit the second page of `https://www.stata.com/manuals/dcollapse.pdf#dcollapse`.

3.2 Customizing Graph Options

Stata options can be very useful, but also very overwhelming. Options are where you can format and customize sections of your graph. These will follow the comma in each graph command.

To start, we will want to add titles to our graph. We can do this with the options **title(“”)**, **xtitle(“”)**, and **ytitle(“”)**. The text we want for each title will be encased in the quotation marks. by default, they should appear red in the do-file.

We can also adjust the labels of each axis. Using **xlabel()** and **ylabel()**, we can set the minimum and maximum value for each axis, as well as the steps taken for each tick. The option **ylabel(10(10)50)**, for instance, would start at 10, show ticks at 20, 30, and 40 before stopping at 50. [3]

In our case, we want to keep the scale of the y-axis, but need to adjust its formatting. In the **ylabel()** option, we can specify this with a comma and the **format** option. The format option uses a percent sign followed by the number of digits to format. Digits following the “.” indicate the number of decimals to round to if necessary, and “fc” indicates we want to include commas where applicable. In our code, we specify **ylabel(, format(%12.0fc))**. Note that if we chose to adjust the scale, that would be inserted before the comma. [4]

Additional stylistic options we can select is the bar color, the intensity of the color-shaded area (**intensity(*0.4)**), and the intensity of the bar outline (**lintensity(*0.9)**). Note that we select the numeric in **bar(1, color(green))** based on the order of the variables in the “over” option.

3.3 Adjusting Graph Size

Sometimes components of your graph will not fit in the space you want it to. There are a couple of different ways to approach this. If there is a crowded x-axis, the simplest solution would be to adjust the size using the option **xsize()** with a value from 1-100 placed in the parentheses. This does not alter the scale of the values on the x-axis, just the physical space it takes in the graphing environment. This also works with **ysize()**. Additional spacing options that are useful include **bargap()** and **aspectratio()**. [5]

If the graph is still cramped, you can also adjust the size of the labels and their angle. You can see in our code we use the **label(angle(45) labsize(small))** to do so. Another option you have is to use **graph hbar** instead of **graph bar**, which will switch the axis which each variable is displayed. Note that using **hbar** adjusts the only the visual display, so options using *xlabel()* and *ylabel()* remain the same.

[3] Visit `https://www.stata.com/manuals/g-3axis_label_options.pdf` for more options
[4] Visit `https://www.stata.com/manuals/dformat.pdf` for more options
[5] Visit `https://www.stata.com/manuals13/g-2graphdisplay.pdf` for more options

3.4 Scatter Plots

A scatter plot in Stata is created using the command **scatter yvar xvar**. This plots individual points based on the values of two numeric variables. Scatter plots are useful for visualizing relationships between variables.

We can enhance scatter plots with options such as *xtitle("")*, *ytitle("")*, and *title("")*, just as we did for the bar charts. Adding a regression line is possible with **lfit**, which overlays a fitted line on top of the scatter plot. This requires the use of the **twoway** command in Stata.

The **twoway** command allows us to combine multiple graph types in a single plot by layering them on top of each other. Each graph component is enclosed in parentheses and separated by spaces.[6] For example, we use **twoway (scatter agriculture_per agriculture_employment) (lfit agriculture_per agriculture_employment)** to add a line of best fit to our scatter plot, which visualizes agricultural employment and the percentage of GDP contributed by the agricultural sector.

3.5 Selecting Data for Graphing

What if we want to graph a subsection of the data? The most straightforward option to select data for graphing is to include an if statement in our command. You can see we did that at the start of out do-file with **line gdp_growth year if countryname == "United States"**. If we are using the **twoway** command, the code can often become long and cumbersome.

An alternative option you see in the do-file is the use of **preserve** and **restore**. These commands temporarily modify your dataset for specific operations and then restore the original dataset structure. With these, you can trim the data with **keep if** and **drop if** prior to generating graphs. This can be helpful when planning out longer strings of commands.

3.6 Line and Area Charts with Twoway

Line graphs allow us to track changes over time. The command **line yvar xvar** plots a line connecting points sequentially based on the x-variable. This is commonly used for time series data. To compare multiple lines, we can add them in the same command using **twoway**. The **lcolor()** option changes line colors: **twoway (line yvar1 xvar, lcolor(blue)) (line yvar2 xvar, lcolor(red))**.

Area charts are similar to line charts but fill the area beneath the lines with color. This is useful for stacked data visualization: **twoway (area yvar xvar, color(green)) (area yvar2 xvar, color(green))**. In our do-file we use the user-written package called *Stckar* as another option for the stacked area charts.

[6] Visit `https://www.stata.com/manuals/g-2graphtwoway.pdf` for more options

3.7 Box Plots

Box plots help visualize the distribution of a numerical variable. The command **graph box yvar, over(xvar)** creates a box plot grouped by a categorical variable.[7] This helps compare medians and identify outliers. Additional options include adding a horizontal reference line using **yline(0)** and **inrange(year,)**, which you see in **graph box gdp_growth if inrange(year, 2000, 2010), over(year) ytitle("GDP Growth") yline(0)**.

If we want to visualize this trend differently, we can generate new variables for the min, max, and mean of *gdp_growth* by *year*, and use **rcap** and **scatter** to plot them on top of one another. These plots can be helpful for visualizing changes over time by groups or across regions.

[7] Visit `https://www.stata.com/manuals13/g-2graphbox.pdf` for more options