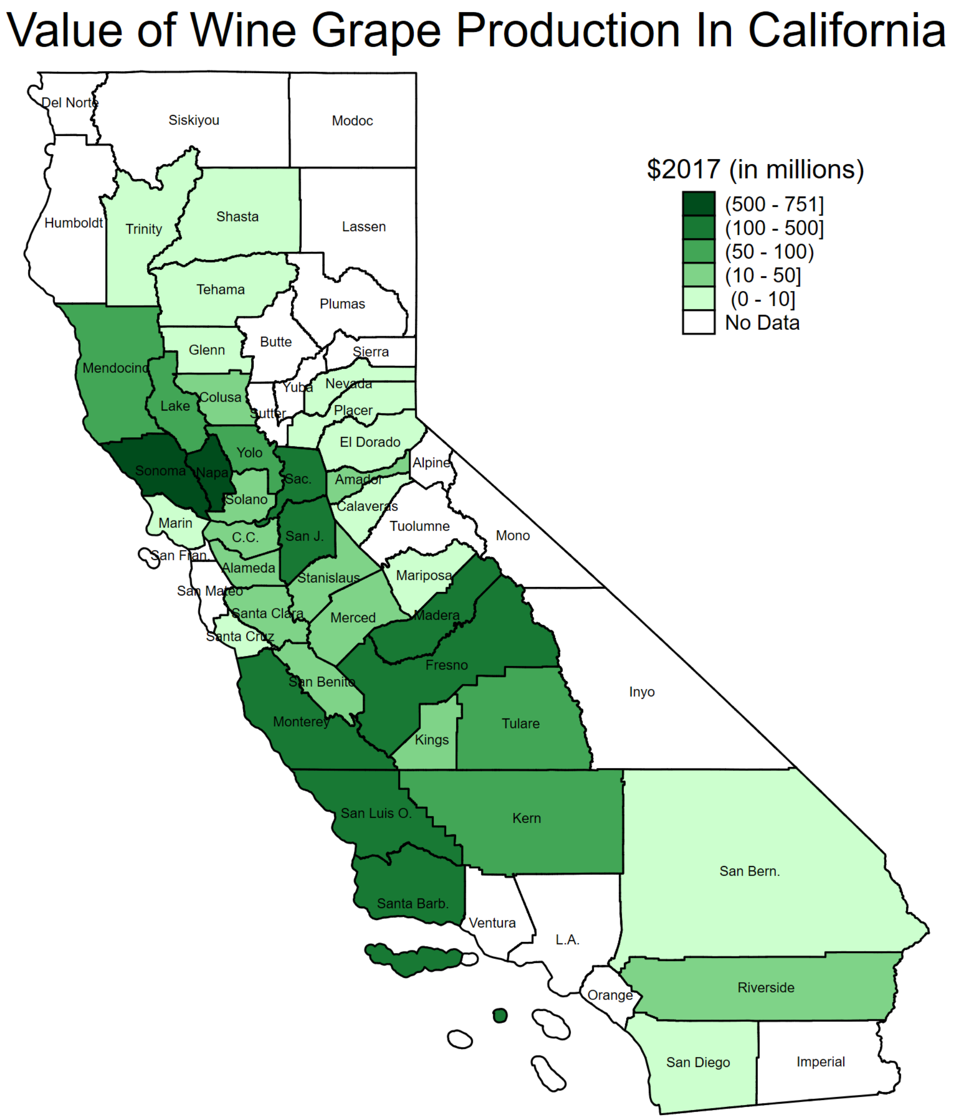
A Beginner's Guide to GIS Mapping in Stata*

Note: Supplemental materials can be downloaded at:
https://www.zachrutledge.com/uploads/1/2/5/6/125679559/stata_gis_guide.zip

Zachariah Rutledge

3/22/20



*For additional information, see the Stata FAQ page: "Working with spmap and maps," by Kevin Crow and William Gould at: https://www.stata.com/support/faqs/graphics/spmap-and-maps.

Getting Started

Step 1: Install the "spmap" and "shp2dta" packages in Stata by running the following two commands:

```
ssc install spmap
ssc install shp2dta
```

Step 2: Download the shape files that contain the GIS boundary information for the geographic regions you want to map. The example I cover here is for California county boundaries. You will need the following two files:[1]

1. One shape file (.shp) specific to the geographic boundaries you want to map (call it FILE1.shp)
2. One database file (.dbf) specific to the geographic boundaries you want to map (call it FILE1.dbf)

Running the Code

Step 3: Unzip the compressed Tiger/Line folder (or the corresponding folder that contains the shape files), set your working directory to the folder where the .shp and .dbf files are located, then run the following command:

```
shp2dta using FILE1.shp, database(FILE2) coordinates(FILE3) genid(ID) gencentroids(Centro) replace
```
[2]

This command takes the .shp and .dbf file and generates two new .dta files. The first .dta file is a database file containing a variable with the geographic labels (this variable is called "NAME" in my example) and other information used by Stata to locate the center of the each geographic region to position the labels (e.g., county names) on the graph. This first file will be saved with the file name FILE2.dta, and the second .dta file is a file that contains coordinate information and is saved with the name FILE3.dta. The spmap command uses the coordinate information contained in the FILE3.dta data file to generate the GIS map. The genid(ID) option generates a new categorical variable called ID that identifies each of the geographic regions contained in the shape files and saves it in the FILE2.dta database file. In the example I use here, it assigns each of the 58 California counties a separate number ranging from 1 to 58. It is important to note that this ID variable will not necessarily match the FIPS codes (or other geographic codes) for the geographic regions in the data that you want to graph. In the example provided here, the crop production data that I want to graph is identified by a variable I have named "countycode," which contains the California county FIPS codes that range from 1 to 115. If you are not using FIPS codes (or some other commonly used geographic coding system that the authors have written into the shape files), it is likely that the ID variable will not correctly identify the regional codes in the data you want to graph (see Step 5 for more information). The "gencentroids(Centro)" option generates two variables (one called x_Centro and the other y_Centro) that, together, identify the X and Y coordinates of the geographic center of each region in your data. These X and Y coordinates are used to place the labels of each region (e.g., county names) using the "label" option of the spmap command if you want them to appear on your map.

Step 4: Make sure the data you want to graph are collapsed into cells that only contain one observation for each geographic region.[3] It may also be more convenient to save a copy of the data you want to graph into

[1]The shape files I use in this example come from the U.S. Census Bureau's "Tiger/Line" shape files for California counties in 2016 and can be downloaded from the California government's data website at: https://data.ca.gov/dataset/ca-geographic-boundaries. Tiger/Line shape files for other boundaries can be obtained online from the US Census Tiger/Line shape files website at: https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html. Tiger/Line shape files are not the only type of shape file you can use. As long as the shape files you use contain the .shp and .dbf files, you should be able to use them. An alternative method is to use the "mif2dta" package with a corrsponding .mif file instead of a .dbf file (not covered here).

[2]The text in red will differ depending on name of the .shp and .dbf files you download, the new .dta file that will be generated containing the coordinate information (you can name it whatever you want...here I have named it FILE3), and the new geographic categorical variable that will be generated in the new .dta database file (you can name this variable whatever you want...here I have named the variable ID). The rest of the command should be executed exactly as I have written it here.

[3]You can have multiple variables, but there can only be one observation per geographic region.

the directory where your shape files are located at this time. In this example, I am graphing the value of wine grapes produced in California counties in 2017, so I have collapsed my data by using the following commands:

```
use "CROP DATA 2010-2017.dta", clear
keep if year==2017
[4] gen FIPSCODE = countycode
collapse (sum)value (first)countycode, by(FIPSCODE)
save Winegrape_Value_2017.dta, replace
```

Step 5: There are two ways you can do this step depending on whether your geographic regions are identified by FIPS codes (or another commonly used geographic code provided by the authors of the shape files).

Method A: If the data you want to graph are already identified by a commonly used geographic identifier like a FIPS code (e.g., county or state), then the following steps should suffice:

Step 5A (i): Load the FILE2.dta database file into Stata.

Step 5A (ii): Destring the variable that identifies the FIPS code in this data set.

Step 5A (iii): Generate a new variable called "FIPSCODE" that contains the FIPS codes.

Step 5A (iV): Save this file with a new name such as FILE2_V2.dta.

Method B: If the data you want to graph is not already identified by a FIPS code (e.g., county or state), you may have to perform the following steps.

Step 5B (i): Manually inspect the FILE2.dta to see if the values in the ID variable (or one of the other categorical geographic variables in the FILE2.dta file) match the values for the geographic variable (e.g., "countycode") in the data you want to graph. If they do match, then rename the ID variable (or the other geographic variable in the FILE2.dta file that matches it) "FIPSCODE" and skip to Step 6, Method A.

Step 5B (ii): If the values for the ID variable (or one of the other categorical geographic variables) in the FILE2.dta file do not match the values for the geographic variable (e.g., "countycode") in the data set you want to map (e.g., wine grape crop value at the county level), you will need to create a separate .dta file that you can use to merge the FILE2.dta data set with the data set that you want to map.[5] In some cases, this may require manually matching the codes in the ID variable to those in the data you want to map.[6] This can be achieved by exporting the FILE2.dta file into an Excel file, creating a new column with a heading name that corresponds to the name of the geographic variable in the data set that you want to graph (e.g., "countycode"), and manually matching the codes from that geographic variable to the ID variable in the Excel file. Then you will have to save a version of that file in .dta format (call it trans.dta) that only contains the ID variable and the geographic variable from the data set that you want to graph (e.g. "countycode"). The trans.dta file will be used to merge the FILE2.dta data file into the data set that you want to graph.

[4]This line of code corresponds to Method A outlined below. If you fall in the case of Method B, you can omit this command. However, if you fall into the case of Method B below, you should use the following command (instead of the command on the next line) to collapse your data: collapse (sum)value, by(countycode).

[5]Alternatively, you can change the codes in the data set you want to graph to match those in the FILE2.dta file and then merge on that variable.

[6]Alternatively, you could try to match one of the other geographic variables (e.g., a variable other than the ID variable) in the FILE2.dta file with the geographic variable in the data you want to graph. There may also be an easier way to perform this step, but I am not aware of one. You will have to use your discretion to figure out which method seems more efficient to you.

Step 6: Follow the steps corresponding to **Step 5: *Method A*** or **Step 5: *Method B*** below.

Method A:

Step 6A (i): Open the data set containing the variables you want to graph. This file should now have a geographic variable called "FIPSCODE."

Step 6A (ii): Merge in the FILE2_v2.dta file into this data file using the variable "FIPSCODE."

Method B:

Step 6B (i): Open the FILE2_v2.dta data file.

Step 6B (ii): Merge in the trans.dta file on the ID variable.

Step 6B (iii): Merge in the data set that you want to graph on the geographic identifier variable (e.g., "countycode").

Step 7: Run the "spmap" commmand. Here is the command I use in my example to generate the graph on the cover page:

spmap value using CA_Counties_Coordinates, id(ID) fcolor(Greens2) plotregion(color(gs16)) label(x(x_Centro) y(y_Centro) label(NAME) size(tiny)) clmethod(custom) clbreaks(0 10 50 100 500 751) title("Value of Wine Grape Production In California") legend(order(6 "(500 - 751]" 5 "(100 - 500]" 4 "(50 - 100)" 3 "(10 - 50]" 2 "(0 - 10]" 1 "No Data")) legend(pos(1) ring(0) bmargin(large) title("$2017 (in millions)", size(small)))

If you were using the file names used in this handout, your spmap command would look like this:[7]

spmap value using FILE3, id(ID) fcolor(Greens2) plotregion(color(gs16)) label(x(x_Centro) y(y_Centro) label(NAME) size(tiny)) clmethod(custom) clbreaks(0 10 50 100 500 751) title("Value of Wine Grape Production In California") legend(order(6 "(500 - 751]" 5 "(100 - 500]" 4 "(50 - 100)" 3 "(10 - 50]" 2 "(0 - 10]" 1 "No Data")) legend(pos(1) ring(0) bmargin(large) title"$2017 (in millions)", size(small)))

[7] Of course you will have to play around with the options to get your graph the way you want it. Type "help spmap" into your command window to explore the options.