CODERS' CORNER

CENTRE FOR THE STUDY OF AFRICAN ECONOMIES

CREATING MAPS IN STATA

A few recent chats about fast ways to create decent maps with my DPhil colleagues have motivated this blog post. While R, QGIS and other programmes are certainly more powerful when it comes to creating cool maps, the popular *spmap* Stata command has a bunch of cool, often overlooked, features.

Let's say we need to map county level statistics compiled by the Kenyan National Bureau of Statistics on the number of formal firm establishments in Kenya. In addition, we want to highlight the largest cities and their population size. Finally, we would like to add information on the set-up of the major urban centres and what share of their population is concentrated in core areas vs the periphery.

Part I – The basics

What do we need?

- A shapefile that contains information on the boundaries of Kenya and its counties
- Geo-coordinates of major Kenyan cities
- Any geographically disaggregated data we want to map (e.g. county population, number of firms)

The second and third piece of information we extract from two excel sheets. The first one contains the number of firms in each county according to the most recent Census of Establishments. The second one lists the latitude and longitude (geo-coordinates) of eleven Kenyan cities and their population count.[1] I first import both files into Stata, label key variables and save them as *.dta* files.

*Import the county level firm statistics

```
import excel using "CoE_county_breakdown", first clear
label variable count_county "Number of firms per county"
save "CoE_county_breakdown", replace
```

*Import the geo-location and population figure of major Kenyan cities

```
import excel using "cities_population", clear first
label variable peri_urban "Peri urban population"
label variable core_urban "Core urban population"
save "cities_population", replace
```

Now we are set up for the core task.

For anyone new to the art of creating maps, the first thing you need is a good base map to work with. Depending on what you are after, a simple map of the boundaries of Kenya or Nairobi might be sufficient. In our case we are looking for information on the shape of the boundaries of Kenyan counties. As mentioned above, we need a suitable *shapefile* to work with (and related files, e.g. a *.dfb* file, that come with it).[2] Usually a quick Google search à la "Kenya shapefile" does the job. Here we use a shapefile compiled by Kenya's Independent Electoral and Boundaries Commission that captures the boundaries of Kenyan counties (file name "*ken_admbnda_adm1_iebc_20180607*" in folder with supplementary material).

We now import the shapefile into Stata. The relevant command is called *shp2dta* and creates two linked Stata files: One is a list of all geographical units, here counties (ken_counties), and one a list of coordinates (kencoord) that describe the related boundaries.

[1] The related statistics can be found here: https://www.knbs.or.ke/?page_id=3142.
[2] Quoting the ESRI website: "A shapefile is an Esri vector data storage format for storing the location, shape, and attributes of geographic features. It is stored as a set of related files and contains one feature class." https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm

In the *spmaps* help file the former is called *master* and the latter *base map*. The two can be linked using a newly generated ID variable that we specify in the options (here I am calling it *county_id*).

*Import shapefile and related information

```
shp2dta using "ken_admbnda_adm1_iebc_20180607", database(ken_counties)
coordinates(kencoord) genid(county_id) replace
```

We now link the information we obtained from the Kenyan National Bureau of Statistics with the master file, i.e. the data set that contains the list of all counties (ken_counties). To do so we need a variable that is common to both the master file (ken_counties) and the file with the firm statistics – in our example we opt for the county name.

*Clean the county names in the newly created "ken_counties" file containing the list of all counties

```
use ken_counties, clear
rename ADM1_EN county_name
replace county_name=upper(county_name)
```

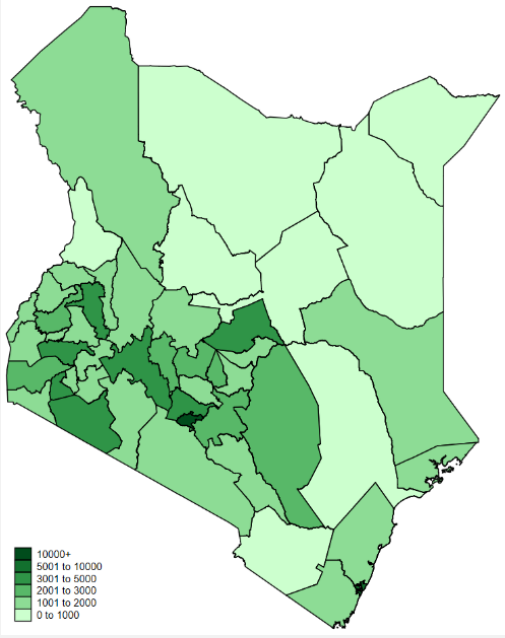
*Now merge with the firm count from the Census of Establishments

```
merge 1:1 county_name using "CoE_county_breakdown"
cap drop _merge
```

Having merged the county level information on firms with the master file, we can now use the Stata command *spmap* to create our first map. We specify *count_county*, the variable capturing the number of establishments in the county, as the variable of interest we want to base our colouring scheme on.[3] Most importantly we need to remind Stata that the file containing the base map is called *kencoord* and the ID that identifies each geographical unit is called *county_id*. I further altered the legend of the map a bit to improve the readability of the map.

```
spmap count_county using kencoord, id(county_id) legend(on)
fcolor(Greens2) ///
 /*specify base map (kencoord) and variable identifying relevant geographic units (county_id)*/
clbreaks(0 1001 2001 3001 5001 10001 50000) clmethod(custom) ///
/*alter categories used for colour coding*/
legend(label(2 "0 to 1000") label(3 "1001 to 2000")  label(4 "2001 to
3000" )  label(5 "3001 to 5000" ) label(6 "5001 to 10000" ) label(7
"10000+" )) //
/*change default labels (just cosmetics really) */
graph export establishments_census.png, replace
```

The result



[3] If you choose to not specify a variable of interest, *spmap* will create a simple map of the boundaries.

Part II – Exploring additional features

We can now add various other features to this map: diagrams, points, lines and polygons. As a first step we will add information on the size and location of the largest Kenyan cities using the *point()* option embedded in the command.

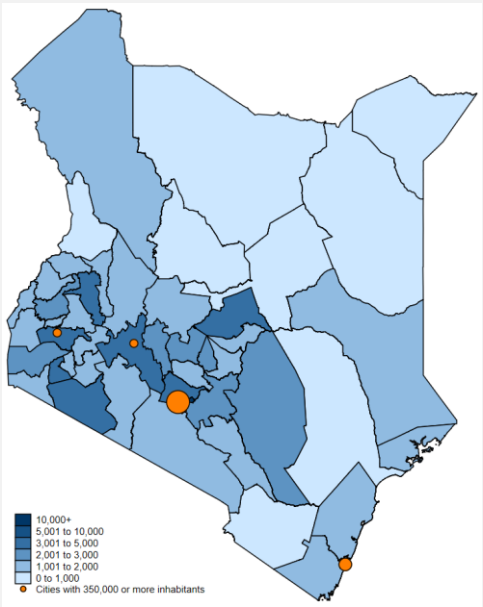
Adding points

```
spmap count_county using kencoord, id(county_id)
point(data("cities_population") xcoord(longitude) ycoord(latitude)
proportional(population) fcolor(orange) ocolor(black) select(keep if
population>350000) legenda(on) leglabel(Cities with population >
350,000)) legend(on) fcolor(Blues) clbreaks(0 1001 2001 3001 5001 10001
50000) clmethod(custom) legend(label(2 "0 to 1,000") label(3 "1,001 to
2,000") label(4 "2,001 to 3,000" ) label(5 "3,001 to 5,000" ) label(6
"5,001 to 10,000" ) label(7 "10,000+" ))
graph export establishments_census_cities.png, replace
```

While the basics remained unchanged, lets have a look at the newly added, highlighted bit of code:

```
point(data("cities_population") xcoord(longitude) ycoord(latitude)
proportional(population) fcolor(orange) ocolor(black) legenda(on)
leglabel(Cities 350,000 or more inhabitants) select(keep if
population>350000))
```

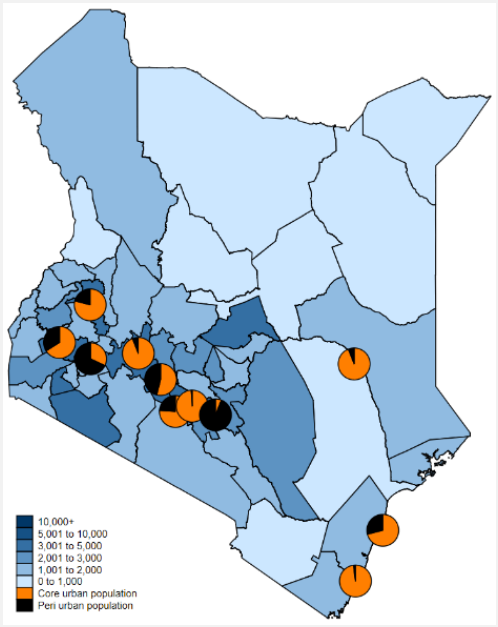
The parenthesis of the *point()* option contain everything Stata needs for correctly mapping the points: (i) the name of the data set that lists the latitude and longitude of the cities (*cities_population*) and (ii) which variables to use as x and y coordinates (*longitude* and *latitude)*. The fun does not end here. We can further specify the size of the points to reflect the relative size of the cities using the *population* variable. We further alter the colour, add a legend for the points and overwrite the default legend. Finally, we specify to only map cities with a population of more than 350,000 using the *select()* option, which can be combined with either *keep* or *drop*.



Adding diagrams

spmaps's *diagram()* option can be used to highlight the share of a city's population living in core urban areas vs the periphery. We again tell Stata to use information from the data set *cities_population*, but now specify the two variables *core_urban* and *peri_urban* as our variables of interest. If two variables of interest are specified the default diagram is a pie chart, if only one is specified, the default is a bar chart. Here we use the sub-option *select()* to exclude the diagram for Nairobi as 100% of its population lives in so-called core urban areas.

```
spmap count_county using kencoord, id(county_id)
diagram(data("cities_population") xcoord(longitude) ycoord(latitude)
variables(core_urban peri_urban) fcolor(orange) legenda(on) select(drop
if city=="Nairobi")) legend(on) fcolor(Blues2) clbreaks(0 1001 2001
3001 5001 10001 50000) clmethod(custom) legend(label(2 "0 to 1,000")
label(3 "1,001 to 2,000") label(4 "2,001 to 3,000" ) label(5 "3,001
to 5,000" ) label(6 "5,001 to 10,000" ) label(7 "10,000+" ) )
graph export peri_core_urban.png, replace
```



Other spatial reference systems

In some cases, our complementary data might be expressed in another spatial reference system than our base map. It is quite common for geographic locations to be expressed in geo-coordinates (like the Kenyan cities above), but we need cartesian coordinates instead (you will notice that when calling *spmap* either your base map or the points are completely off). Luckily there is a handy Stata command called *geo2xy* that transforms geographic latitude and longitude into cartesian coordinates.

```
use cities_population, clear
```
*Transform the geographic latitude and longitude into cartesian coordinates
```
geo2xy latitude longitude , generate(y_lat x_long) proj(mercator,
6378137 298.257223563 0)
```

Two very useful related sources are this forum post by Robert Picard, who discusses how to improve the legend of your map, and these slides by Maurizio Pisati that discuss more features of the command and other cool applications like mapping Lombarida's rail network.

Verena Wiedemann, DPhil Candidate in Economics, St Antony's College, Oxford
27 January 2020