**Geovisualization Using PySAL**

**Team GetFramed**

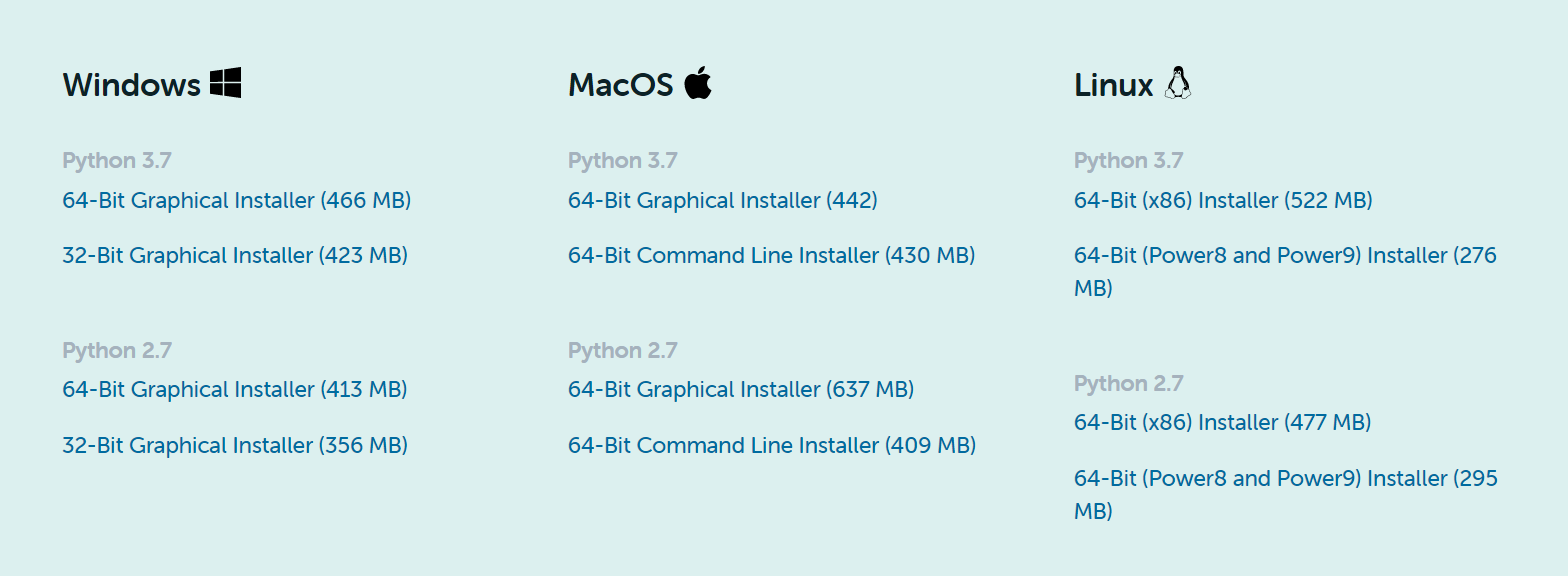
**4/27/20**

1. **Overview**

The focus of this portion of the project was to find ways to visualize datasets. PySAL provides by far the easiest methods of performing this task. In the examples that I used, the files pulled from included data from the Tokyo metropolitan area, Mexico, and Columbus. These datasets all come prepackaged with PySAL, but the goal is to have integration so that users could upload their own datasets and still attain a visual representation.

1. **Getting Started**

I began by trying to install PySAL through the native “pip install” function. However, I quickly ran into issues with making sure all the dependencies and necessary repositories were accounted for. One could certainly get PySAL up and running using this method, but I found that downloading Anaconda and using their installation process to be much more effective and simpler.



I would recommend downloading the latest version of Anaconda, as it comes with plenty of useful bonus features and tools outside of what we will be needing it for. Once downloaded, open the anaconda3 prompt terminal so you can begin downloading all that you will need. Using a virtual environment for python projects is something that I have become accustomed to so I will have that setup be the first step, but feel free to not use one. The installation process will be the same if you decide to skip this step.

To create an environment in anaconda, enter this into the terminal.

conda create --name myenv

“myenv” here is a sample name, you would enter what you want to call your virtual environment and replace that part of the terminal line.

In most cases, a single download of PySAL should get you everything. Enter this into the terminal.

conda install -c anaconda pysal

To be safe, let us run two more anaconda installations to make sure we have all the necessary dependencies.

The next installation will be matplotlib and splot

conda install -c anaconda matplotlib

conda install -c anaconda splot

Splot will allow us to connect spatial analysis that is performed in PySAL with various visualization toolkits found within matplotlib. Both static plots and interactive visualizations are available. Matplotlib is a plotting library that displays in 2D. It is primarily used alongside python but can interact with other software such as MATLAB, shells, and web servers.

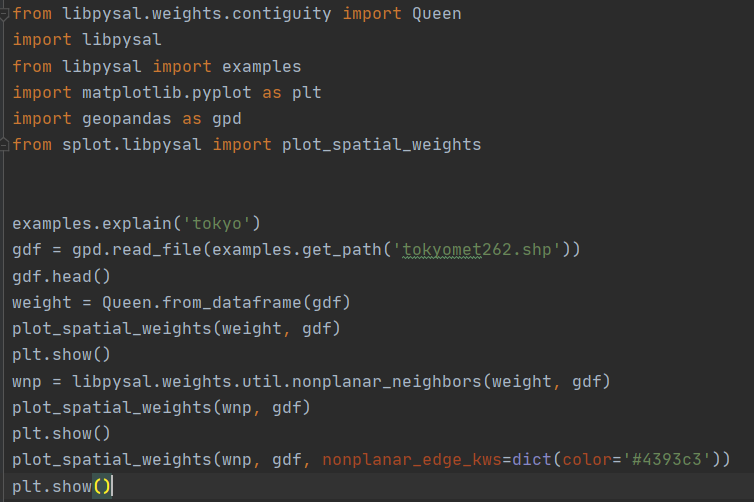
**3.0 Assessing Neighbors & Spatial Weights**

Now that you have everything installed, go ahead and open how your coding environment. For this project I used PyCharm.

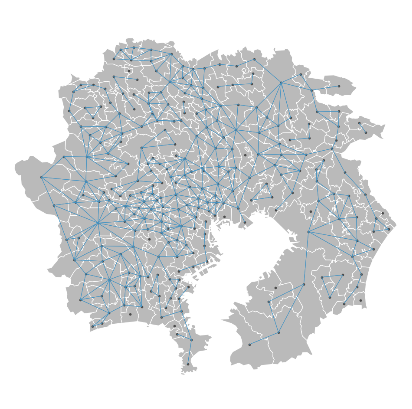
<https://github.com/J-Baka/Python-Internship>

Head over to this repository to grab the files for these visualization methods. It will be easier than coding them yourselves.

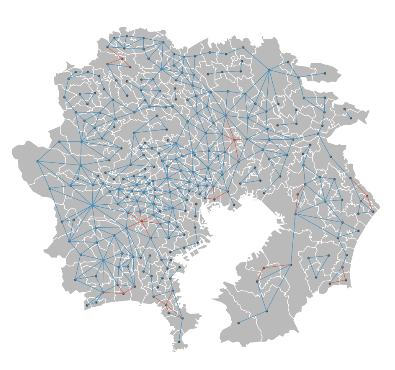
The first file that you can grab is title “Map.py”.



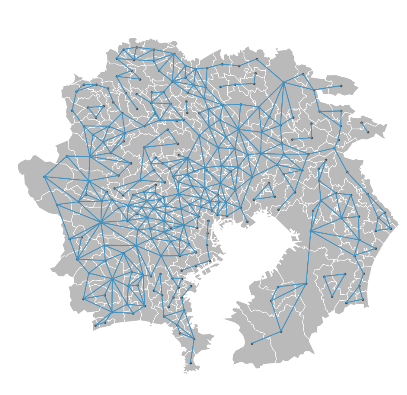
Using a preloaded dataset of the Tokyo metropolitan area, the first plot will display the data using the spatial weights command. Your result should look like the map below.



A solid representation, but you can see that there are some dots that by themselves. The next plot will fix that. Pulling from the libpysal library, a way to visualize nonplanar neighbors will be implemented. The next plot in the line will produce this map.

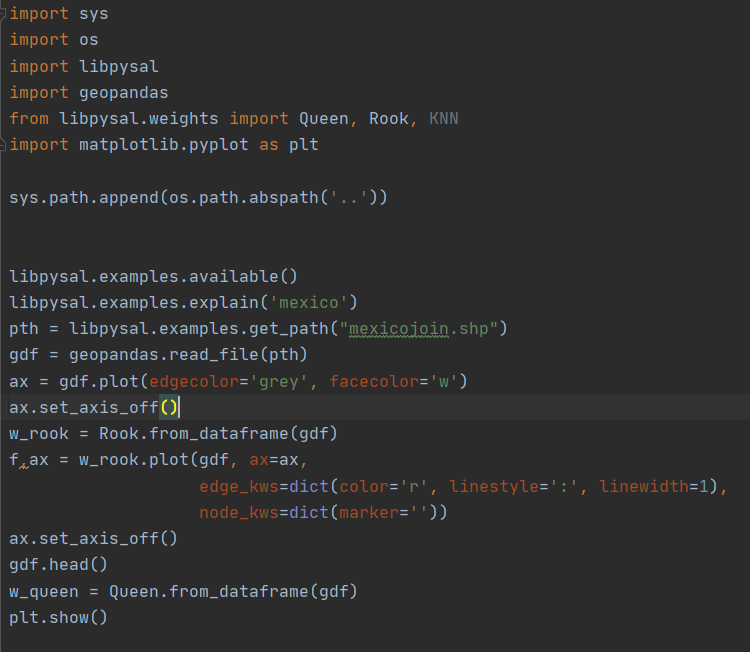


Now the dots that were previously alone are now connected to those that are nearest to them. They are also displayed in red to differentiate. The final plot will take all of the previous information and simply make the map more clear.



The lines are now darker, thicker, and all the same color.

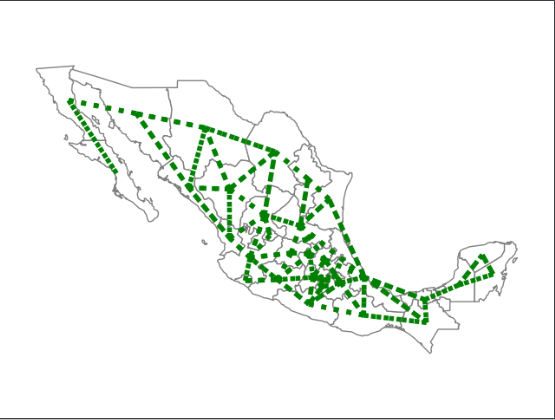
Next, go back to the repository and pull the “weights.py” file. This will be very similar to what the “map.py” accomplishes, but with the ability to fine tune the map a bit more.



This is the code that will be used. PySAL again makes things easy by having the input that is necessary for all of these files to work being a single .shp file. We will be using the Rook function, a rook being a neighbor whose states share an edge on their borders. This code also provides the ability to chose the color of the line that we want, the style of the line, and how thick the line is. The result comes out to be a more dynamic looking map.



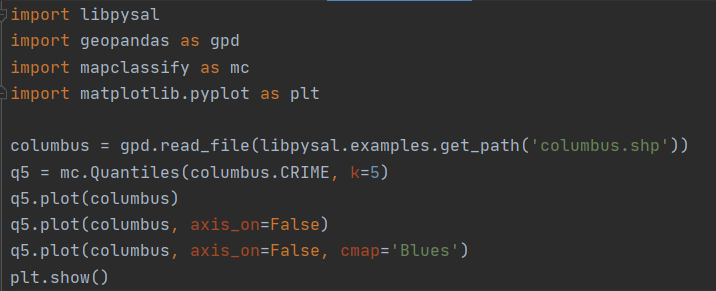
For the sake of example, here is how it looks when the line width is changed to 4 and the color to green.



1. **Quantiles**

A quantile where a sample of data is divided into equal-sized, adjacent, subgroups. PySAL makes this easy to implement as there is a function simply called “QUANTILES”.

From the repository, grab the file named “Quantiles.py”. It will look like this,



The code functions in the same regard as the previous section, where all that is needed is a .shp file. The “Quantiles” function will break up the CRIME data from the Columbus file, and then the plots will display that information according to the inputs the user desires. The results of the three plots that I used are shown below.

