GEOSPATIAL DATA VISUALISATIONS WITH PYTHON

A world map has always inspired wanderlust in me. However, in the world of technology, maps are becoming principal tools to analyse data for every industry vertical. The way we understand and use geospatial data today has completely revolutionised data analytics and business intelligence.

This article is an attempt to show few geospatial data visualisation options in Python using Folium. What is nice about Folium is that it was developed for the sole purpose of visualising geospatial data. While other libraries are available to visualise geospatial data, such as plotly, they might have a cap on how many API calls you can make within a defined time frame. Folium, on the other hand, is completely free.

INTRODUCTION TO FOLIUM

Folium is a powerful Python library that helps you create several types of Leaflet maps. The fact that the Folium results are interactive makes this library very useful for dashboard building.

From the official Folium documentation page:

Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Manipulate your data in Python, then visualize it in on a Leaflet map via Folium.
Folium makes it easy to visualize data that's been manipulated in Python on an interactive Leaflet map. It enables both the binding of data to a map for choropleth visualizations as well as passing Vincent/Vega visualizations as markers on the map.
The library has a number of built-in tilesets from OpenStreetMap, Mapbox, and Stamen, and supports custom tilesets with Mapbox or Cloudmade API keys. Folium supports both GeoJSON and TopoJSON overlays, as well as the binding of data to those overlays to create choropleth maps with color-brewer color schemes.

Generating the world map is straight forward in Folium. You simply create a Folium *Map* object and then you display it. What is attractive about Folium maps is that they are interactive, so you can zoom into any region of interest despite the initial zoom level.

Image1: World map created using Folium

You can customize this default definition of the world map by specifying the centre of your map and the initial zoom level. All locations on a map are defined by their respective *Latitude* and *Longitude* values. So you can create a map and pass in a centre of *Latitude* and *Longitude* values of **[0, 0]**.For a defined centre, you can also define the initial zoom level into that location when the map is rendered.

The higher the zoom level the more the map is zoomed into the centre.

Below is an example of a map of India created with a zoom level 5.

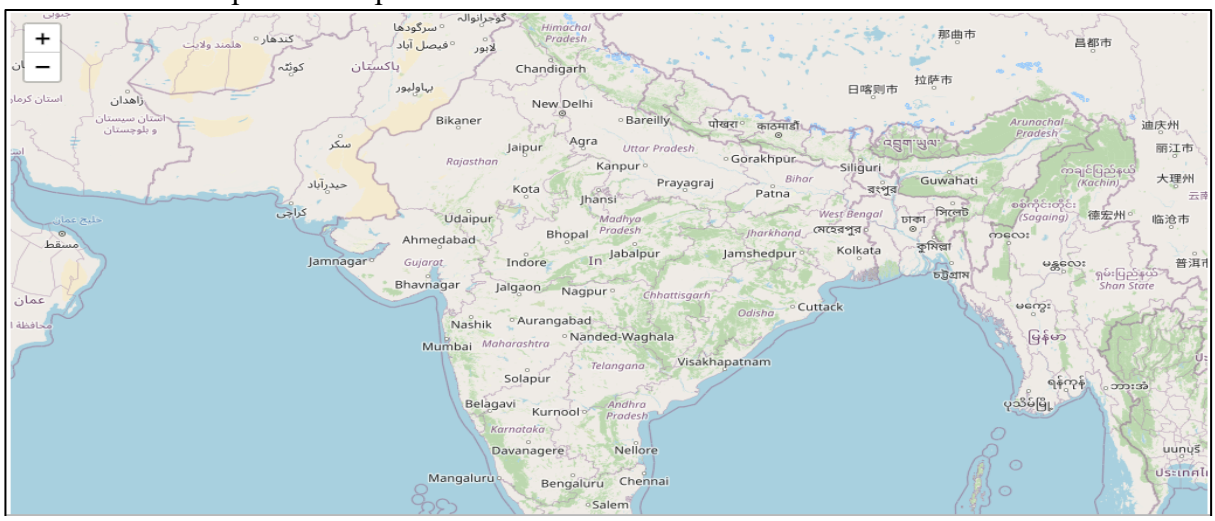

Image 2: Map of India

As you can see, the higher the zoom level the more the map is zoomed into the given centre. Another cool feature of **Folium** is that you can generate different map styles

A. Stamen Toner Maps

These are high-contrast B+W (black and white) maps. They are perfect for data mashups and exploring river meanders and coastal zones. Let's create a Stamen Toner map of canada with a zoom level of 4. Since the water bodies of Canada are very evident at zoom level 4 itself, its the best country that can be taken for example map.

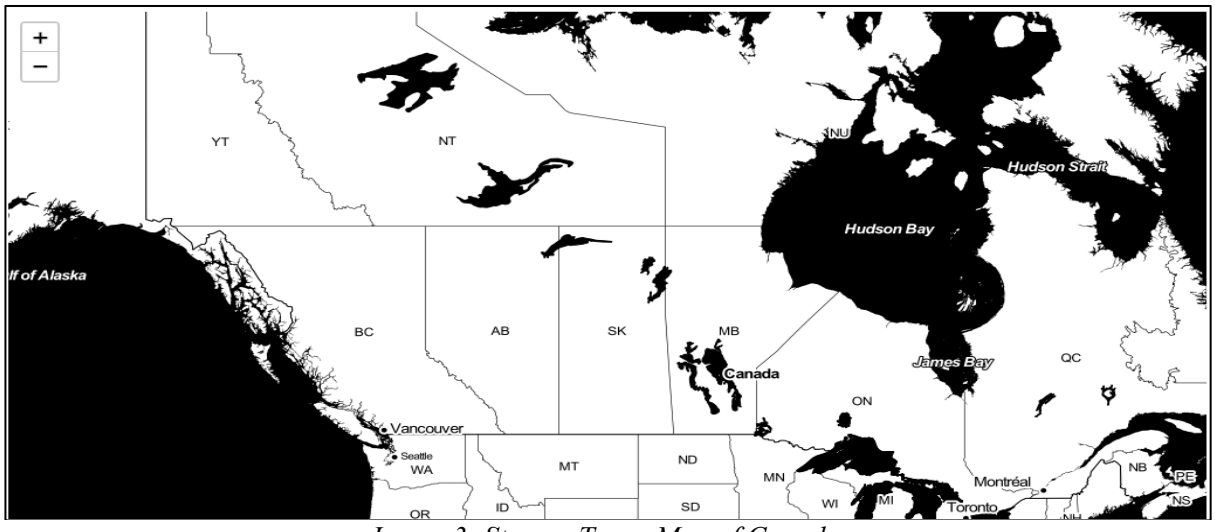

Image 3: Stamen Toner Map of Canada

B. Stamen Terrain Maps

These are maps that feature hill shading and natural vegetation colours. They showcase advanced labelling and linework generalization of dual-carriageway roads. Let's create a Stamen Terrain map of India with zoom level 6.

Image 4: Stamen Terrain Map of India

Maps with Markers

This is a very important feature that is used widely for creating maps with layers. The layers on the map can be any reasonable data which details or define a geography. For this section we are going to use the "San Francisco Police Department Incidents for the year of 2016".

Datasets

San Francisco Police Department Incidents for the year 2016 - Police Department Incidents from San Francisco public data portal. Incidents derived from San Francisco Police Department (SFPD) Crime Incident Reporting system. Updated daily, showing data for the entire year of 2016. Address and location has been anonymized by moving to mid-block or to an intersection.

I'm going to skip the data wrangling part as the article is more focused on Map visualization. However for the details steps involved in data wrangling please use the below mention link to my GitHub account. There is a detailed jupyter notebook on the same.

So the resulting data frame has 100 reported crimes as rows and each row consists of 13 features/attributes:

1. **IncidntNum**: Incident Number
2. **Category**: Category of crime or incident
3. **Descript**: Description of the crime or incident
4. **DayOfWeek**: The day of week on which the incident occurred
5. **Date**: The Date on which the incident occurred

6. **Time**: The time of day on which the incident occurred
7. **PdDistrict**: The police department district
8. **Resolution**: The resolution of the crime in terms whether the perpetrator was arrested or not
9. **Address**: The closest address to where the incident took place
10. **X**: The longitude value of the crime location
11. **Y**: The latitude value of the crime location
12. **Location**: A tuple of the latitude and the longitude values
13. **PdId**: The police department ID

Let's visualize where these crimes took place in the city of San Francisco. We will use the default style and we will initialize the zoom level to 12.

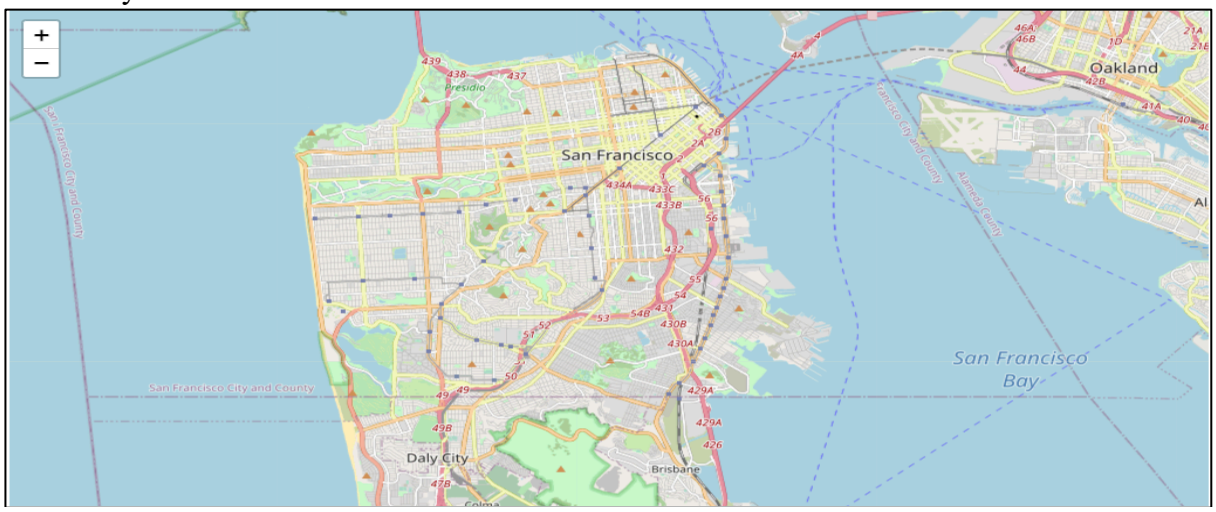

Image 5: Map of San Francisco created using Folium

Now let's superimpose the locations of the crimes onto the map. The way to do that in **Folium** is to create a *feature group* with its own features and style and then add it to the san Francisco map. This is done as follows.

A. Instantiate a feature group for the incidents in the data frame.
B. Loop through the 100 crimes (Geocodes - X & Y) and add each to the incidents feature group.

Image 6: Crime data superimposed on the map of SF.

You can also add some pop-up text that would get displayed when you hover over a marker. Let's make each marker display the category of the crime when hovered over.

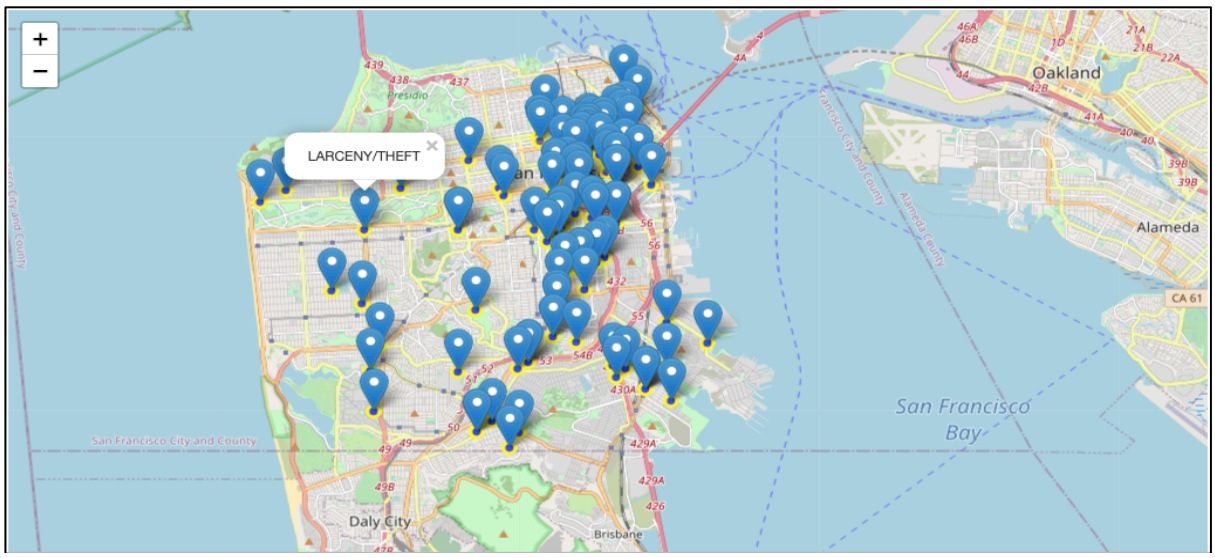

Image 7: Pop-ups added to each crime spots.

Isn't this really cool? Now you are able to know what crime category occurred at each marker.

If you find the map to be so congested will all these markers, there are two remedies to this problem. The simpler solution is to remove these location markers and just add the text to the circle markers themselves.

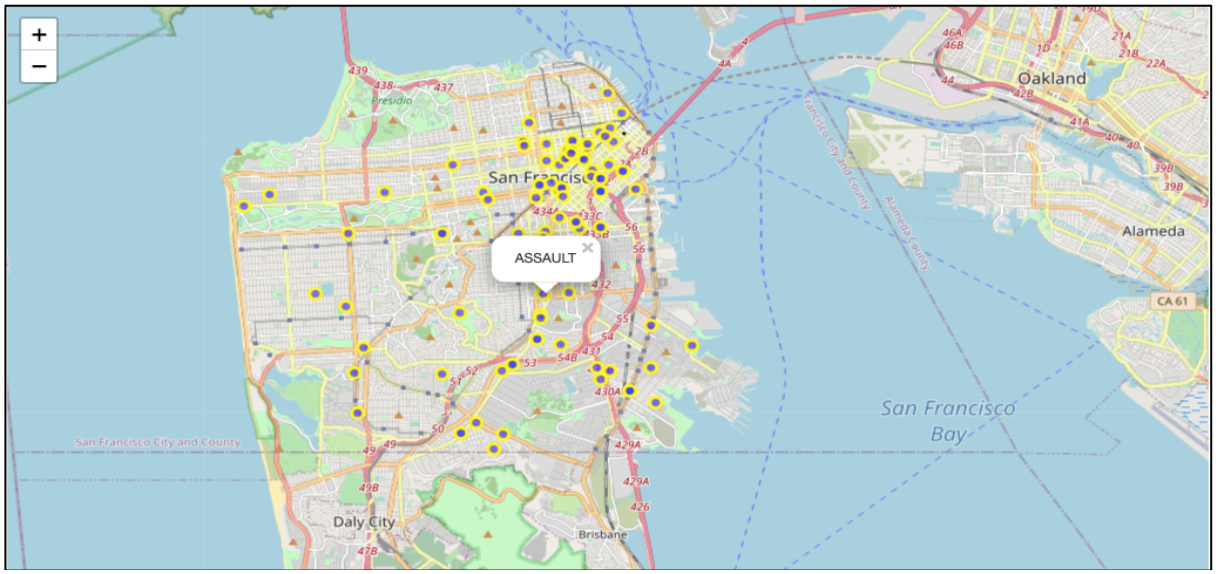

Image 8: Markers added to the crime spots

The other proper remedy is to group the markers into different clusters. Each cluster is then represented by the number of crimes in each neighbourhood. These clusters can be thought of as pockets of San Francisco which you can then analyse separately.

To implement this, we start off by instantiating a *MarkerCluster* object and adding all the data points in the data frame to this object.

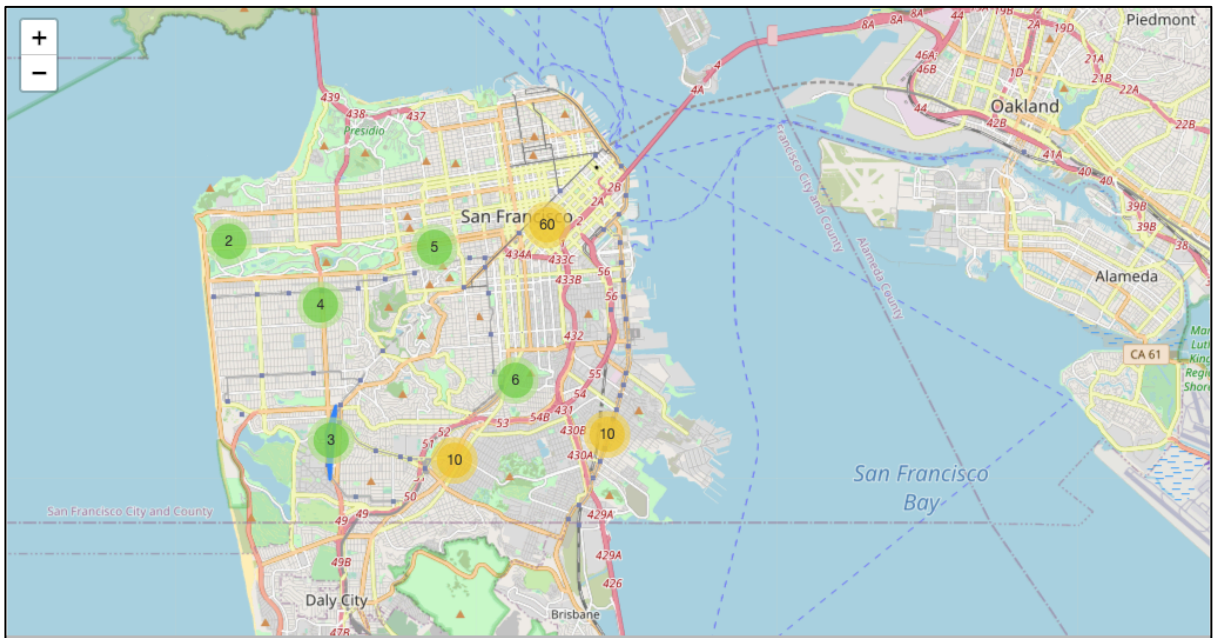

Image 9: MarkerCluster created for crime spots.

On this map, when you zoom out all the way, all markers are grouped into one cluster, *the global cluster*, of 100 markers or crimes, which is the total number of crimes in our data frame. Once you start zooming in, the *global cluster* will start breaking up into smaller clusters. Zooming in all the way will result in individual markers.

CHOROPLETH MAPS

A Choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map, such as population density or per-capita income. The choropleth map provides an easy way to visualize how a measurement varies across a geographic area or it shows the level of variability within a region.

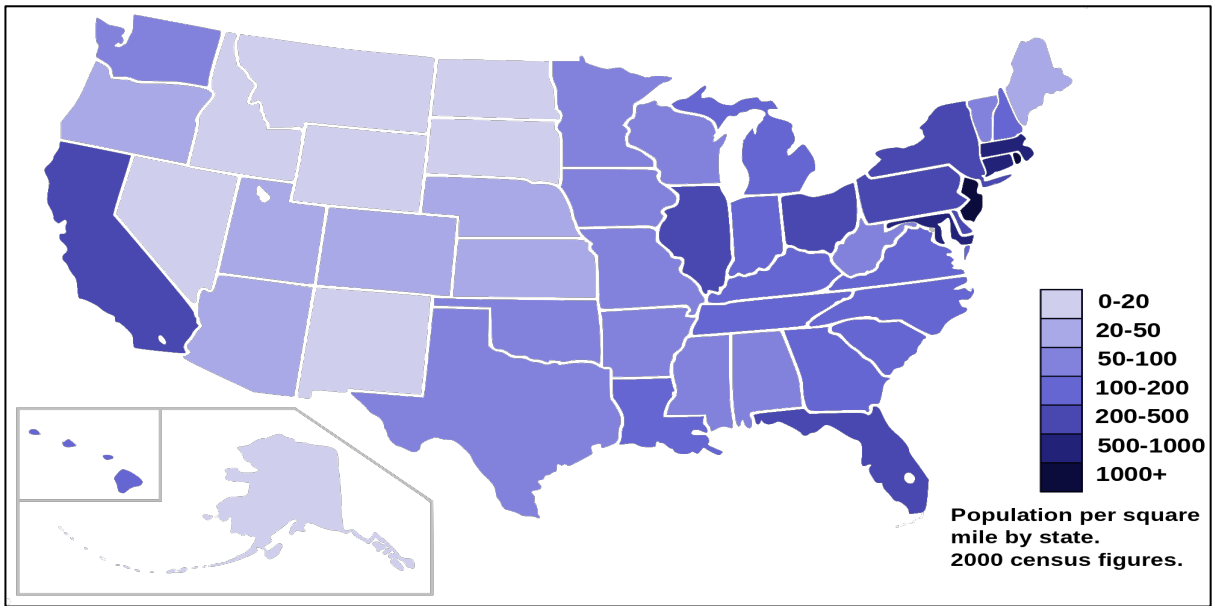

Image 10: Choropleth map of the US depicting the population by square mile per state.

We would use the "San Francisco Police Department Incidents for the year of 2016" data to create this map. Again I'm going to skip the data wrangling part to create this map however, we will use the total number of crimes in each neighbourhood of San Francisco to create it. This is how the final data would look like.

NEIGHBOURHOODS	COUNT OF CRIMES
BAYVIEW	14303
CENTRAL	17666
INGLESIDE	11594
MISSION	19503
NORTHERN	20100
PARK	8699
RICHMOND	8922
SOUTHERN	28445
TARAVAL	11325
TENDERLOIN	9942

Table 1: Table showing the data used for creating chloropleth map for SF

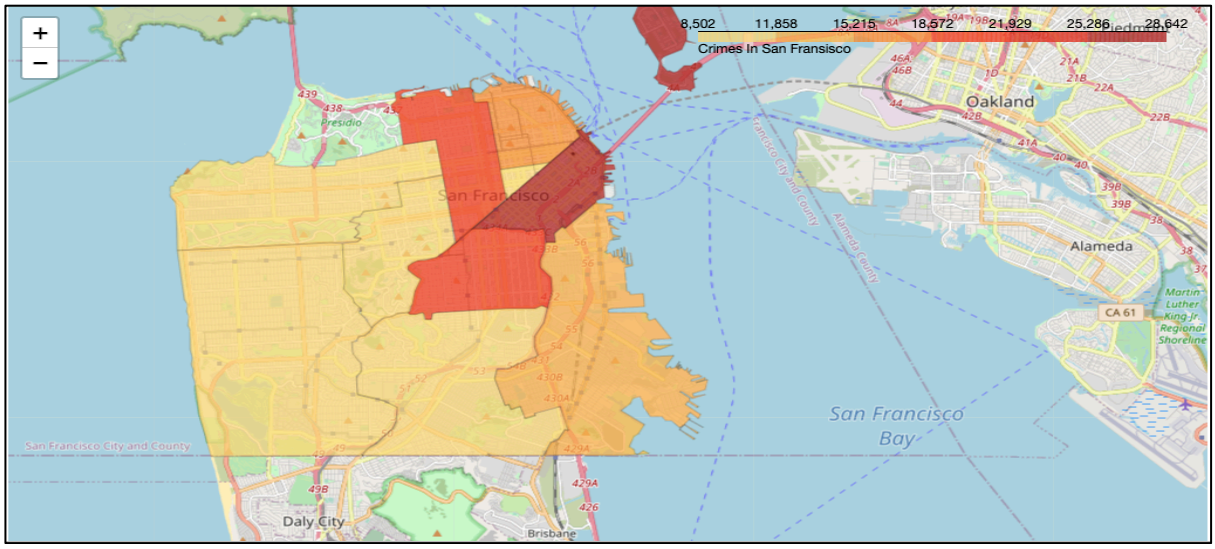

Image 10: Choropleth map of SF showing the crime rate

The above map shows the density variation of the crimes in the city of San Francisco.

This was my attempt to show how well the Folium library in python can be used for generating map based visualizations. So do use Folium to create wonderful maps and visualizations.

For a detailed version of this article with python codes for generating these maps and to understand the background data wrangling, please visit my GitHub account using the link below.

Note: This article is an outcome of a professional certification course I recently took to expand my knowledge of Data Science using Python. Hence some of the datasets used for creating visualizations are from the same course curriculum. Also, the examples in this article heavily relies on *pandas* and **Numpy** for data wrangling, analysis, and visualization. The primary plotting library we will explore in this lab is **Folium**.