Geospatial Analysis: 3rd lesson – Interactive Maps

In this tutorial, you'll learn how to create interactive maps with the **folium** package. Along the way, you'll apply your new skills to visualize Boston crime data.

import pandas as pd

import geopandas as gpd

import math

/opt/conda/lib/python3.7/site-packages/geopandas/\_compat.py:115: UserWarning: The Shapely GEOS version (3.9.1-CAPI-1.14.2) is incompatible with the GEOS version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions between both will be slow.

shapely\_geos\_version, geos\_capi\_version\_string

import folium

from folium import Choropleth, Circle, Marker

from folium.plugins import HeatMap, MarkerCluster

Your first interactive map:

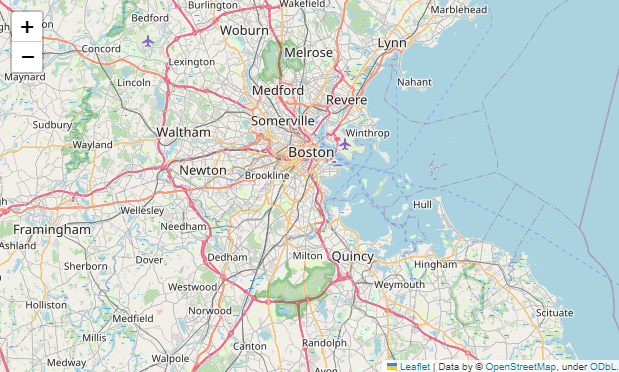
We begin by creating a relatively simple map with folium.Map().

*# Create a map*

m\_1 = folium.Map(location=[42.32,-71.0589], tiles='openstreetmap', zoom\_start=10)

*# Display the map*

m\_1



Several arguments customize the appearance of map:

* location sets the initial center of the map. We use the latitude (42.32° N) and longitude (-71.0589° E) of the city of Boston.
* tiles changes the styling of the map; in this case, we choose the OpenStreetMap style.
* zoom\_start sets the initial level of zoom of the map, where higher values zoom in closer to the map.

The data:

Now, we'll add some crime data to the map! We won't focus on the data loading step. Instead, you can imagine you are at a point where you already have the data in a pandas DataFrame crimes. The first five rows of the data are shown below.

*# Load the data*

crimes = pd.read\_csv("../input/geospatial-learn-course-data/crimes-in-boston/crimes-in-boston/crime.csv", encoding='latin-1')

*# Drop rows with missing locations*

crimes.dropna(subset=['Lat', 'Long', 'DISTRICT'], inplace=True)

# Focus on major crimes in 2018

crimes = crimes[crimes.OFFENSE\_CODE\_GROUP.isin([

'Larceny', 'Auto Theft', 'Robbery', 'Larceny From Motor Vehicle', 'Residential Burglary',

'Simple Assault', 'Harassment', 'Ballistics', 'Aggravated Assault', 'Other Burglary',

'Arson', 'Commercial Burglary', 'HOME INVASION', 'Homicide', 'Criminal Harassment',

'Manslaughter'])]

crimes = crimes[crimes.YEAR>=2018]

# Print the first five rows of the table

crimes.head()

INCIDENT\_NUMBER OFFENSE\_CODE OFFENSE\_CODE\_GROUP OFFENSE\_DESCRIPTION DISTRICT REPORTING\_AREA SHOOTING OCCURRED\_ON\_DATE YEAR MONTH DAY\_OF\_WEEK HOUR UCR\_PART STREET Lat Long Location

0 I182070945 619 Larceny LARCENY ALL OTHERS D14 808 NaN 2018-09-02 13:00:00 2018 9 Sunday 13 Part One LINCOLN ST 42.357791 -71.139371 (42.35779134, -71.13937053)

6 I182070933 724 Auto Theft AUTO THEFT B2 330 NaN 2018-09-03 21:25:00 2018 9 Monday 21 Part One NORMANDY ST 42.306072 -71.082733 (42.30607218, -71.08273260)

8 I182070931 301 Robbery ROBBERY - STREET C6 177 NaN 2018-09-03 20:48:00 2018 9 Monday 20 Part One MASSACHUSETTS AVE 42.331521 -71.070853 (42.33152148, -71.07085307)

19 I182070915 614 Larceny From Motor Vehicle LARCENY THEFT FROM MV - NON-ACCESSORY B2 181 NaN 2018-09-02 18:00:00 2018 9 Sunday 18 Part One SHIRLEY ST 42.325695 -71.068168 (42.32569490, -71.06816778)

24 I182070908 522 Residential Burglary BURGLARY - RESIDENTIAL - NO FORCE B2 911 NaN 2018-09-03 18:38:00 2018 9 Monday 18 Part One ANNUNCIATION RD 42.335062 -71.093168 (42.33506218, -71.09316781)

Plotting points:

To reduce the amount of data we need to fit on the map, we'll temporarily confine our attention to daytime robberies.

daytime\_robberies = crimes[((crimes.OFFENSE\_CODE\_GROUP == 'Robbery') & \

(crimes.HOUR.isin(range(9,18))))]

folium.Marker() function:

We add markers to the map with folium.Marker(). Each marker below corresponds to a different robbery.

*# Create a map*

m\_2 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom\_start=13)

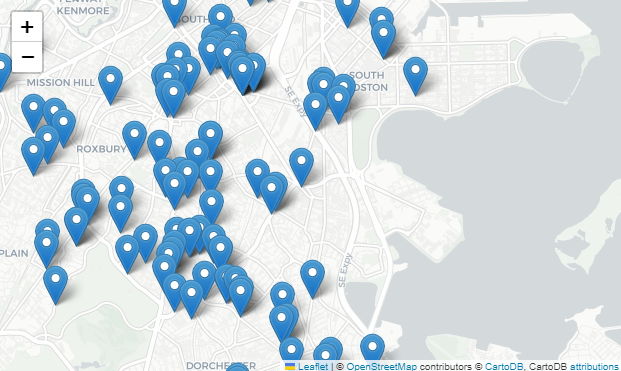
*# Add points to the map*

for idx, row **in** daytime\_robberies.iterrows():

Marker([row['Lat'], row['Long']]).add\_to(m\_2)

*# Display the map*

m\_2



folium.plugins.MarkerCluster() function:

If we have a lot of markers to add, folium.plugins.MarkerCluster() can help to declutter the map. Each marker is added to a MarkerCluster object.

*# Create the map*

m\_3 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom\_start=13)

*# Add points to the map*

mc = MarkerCluster()

for idx, row **in** daytime\_robberies.iterrows():

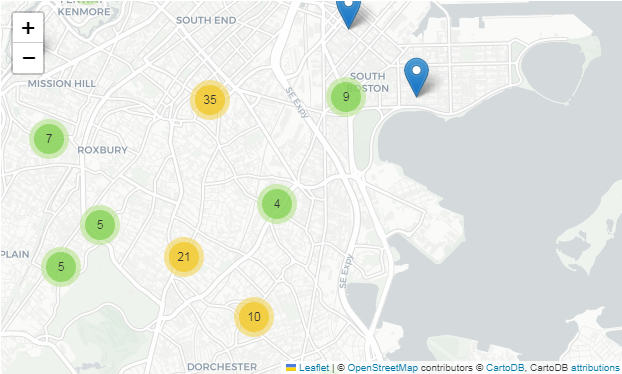
if **not** math.isnan(row['Long']) **and** **not** math.isnan(row['Lat']):

mc.add\_child(Marker([row['Lat'], row['Long']]))

m\_3.add\_child(mc)

*# Display the map*

m\_3



Bubble maps:

A bubble map uses circles instead of markers. By varying the size and color of each circle, we can also show the relationship between location and two other variables.

We create a bubble map by using folium.Circle() to iteratively add circles. In the code cell below, robberies that occurred in hours 9-12 are plotted in green, whereas robberies from hours 13-17 are plotted in red.

*# Create a base map*

m\_4 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom\_start=13)

def color\_producer(val):

if val <= 12:

return 'forestgreen'

else:

return 'darkred'

*# Add a bubble map to the base map*

for i **in** range(0,len(daytime\_robberies)):

Circle(

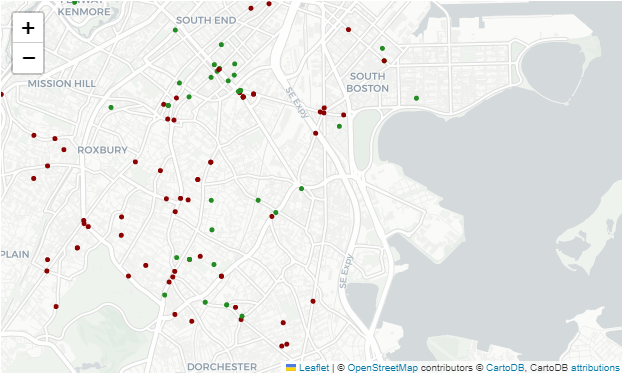
location=[daytime\_robberies.iloc[i]['Lat'], daytime\_robberies.iloc[i]['Long']],

radius=20,

color=color\_producer(daytime\_robberies.iloc[i]['HOUR'])).add\_to(m\_4)

*# Display the map*

m\_4



Note that folium.Circle() takes several arguments:

* location is a list containing the center of the circle, in latitude and longitude.
* radius sets the radius of the circle (note that in a traditional bubble map, the radius of each circle is allowed to vary, we can implement this by defining a function similar to the color\_producer() function that is used to vary the color of each circle).
* color sets the color of each circle (the color\_producer() function is used to visualize the effect of the hour on robbery location).

Heatmaps:

To create a heatmap, we use folium.plugins.HeatMap(). This shows the density of crime in different areas of the city, where red areas have relatively more criminal incidents. As we did expect for a big city, most of the crime happens near the center.

*# Create a base map*

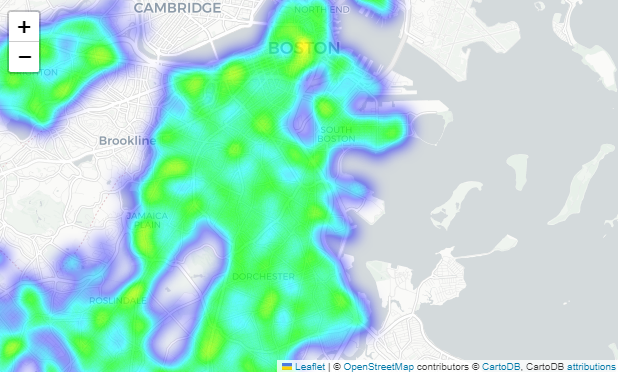
m\_5 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom\_start=12)

*# Add a heatmap to the base map*

HeatMap(data=crimes[['Lat', 'Long']], radius=10).add\_to(m\_5)

*# Display the map*

m\_5



As you can see in the code cell above, folium.plugins.HeatMap() function takes a couple of arguments:

* data is a DataFrame containing the locations that we did like to plot.
* radius controls the smoothness of the heatmap. Higher values make the heatmap look smoother (i.e., with fewer gaps).

Choropleth maps:

To understand how crime varies by police district, we'll create a choropleth map. As a first step, we create a GeoDataFrame where each district is assigned a different row, and the "geometry" column contains the geographical boundaries.

*# GeoDataFrame with geographical boundaries of Boston police districts*

districts\_full = gpd.read\_file('../input/geospatial-learn-course-data/Police\_Districts/Police\_Districts/Police\_Districts.shp')

districts = districts\_full[["DISTRICT", "geometry"]].set\_index("DISTRICT")

districts.head()

geometry

DISTRICT

A15 MULTIPOLYGON (((-71.07416 42.39051, -71.07415 ...

A7 MULTIPOLYGON (((-70.99644 42.39557, -70.99644 ...

A1 POLYGON ((-71.05200 42.36884, -71.05169 42.368...

C6 POLYGON ((-71.04406 42.35403, -71.04412 42.353...

D4 POLYGON ((-71.07416 42.35724, -71.07359 42.357...

We also create a Pandas Series called plot\_dict that shows the number of crimes in each district.

*# Number of crimes in each police district*

plot\_dict = crimes.DISTRICT.value\_counts()

plot\_dict.head()

D4 2885

B2 2231

A1 2130

C11 1899

B3 1421

Name: DISTRICT, dtype: int64

It's very important that plot\_dict has the same index as districts - this is how the code knows how to match the geographical boundaries with appropriate colors. Using the folium.Choropleth() class, we can create a choropleth map.

*# Create a base map*

m\_6 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom\_start=12)

*# Add a choropleth map to the base map*

Choropleth(geo\_data=districts.\_\_geo\_interface\_\_,

data=plot\_dict,

key\_on="feature.id",

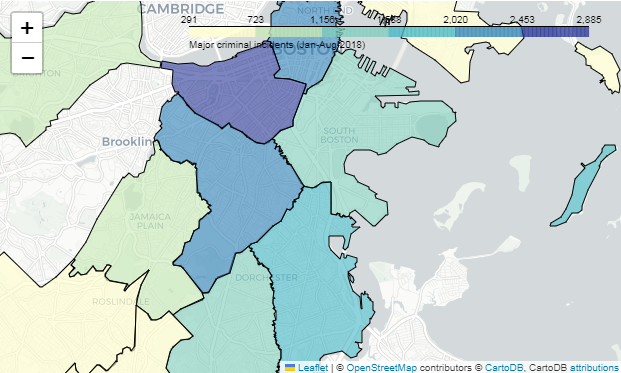
fill\_color='YlGnBu',

legend\_name='Major criminal incidents (Jan-Aug 2018)'

).add\_to(m\_6)

*# Display the map*

m\_6



Note that folium.Choropleth() function takes several arguments:

* geo\_data is a GeoJSON FeatureCollection containing the boundaries of each geographical area (in the code above, we convert the districts GeoDataFrame to a GeoJSON FeatureCollection with the \_\_geo\_interface\_\_ attribute).
* data is a Pandas Series containing the values that will be used to color-code each geographical area.
* key\_on will always be set to feature.id().This refers to the fact that the GeoDataFrame used for geo\_data and the Pandas Series provided in data have the same index. To understand the details, we'd have to look more closely at the structure of a GeoJSON Feature Collection (where the value corresponding to the "features" key is a list, wherein each entry is a dictionary containing an "id" key).
* fill\_color sets the color scale.
* legend\_name labels the legend in the top right corner of the map.