SEA LEVEL RISE DATA LAYERING
Project Report & Code Documentation

Project Overview:
The purpose of this project is to visualize Sea Level Rise and Flood risk due to the rise. To achieve this, map visualization is done by adding a data-layer over Google maps and Wikimedia map. One set of visualization depicts Flood risk score at a national level for all the zip codes in the United States. The results can also be filtered to view the risk at a city as well as at a state level.
Another set of visualization depicts the Sea Level Rise in centimeters at a national level for various coastal cities of the United States.

Datasets:
The following datasets have been used to achieve the results. All these datasets are from verified sources and are publicly available to use.

- **First Street data:** This dataset provides a zip code wise records of average flood risk score due to sea level rise all over the US. It also has the latitude and longitude information for all the zip codes which is required for layering.
  https://firststreet.org/data-access/public-access/

- **Tides and currents data:** The National Oceanic and Atmospheric Administration (NOAA) provides city-wise sea level rise dataset for different scenarios. It has observed data from year 1960-2020, projection based data till year 2100 and also has the latitude and longitude information for the cities.
  https://tidesandcurrents.noaa.gov/sltrends/sltrends.shtml

- **US zip code data:** This dataset has information about the latitude, longitude, state, and city for all the zip codes in the United States. This extra dataset is merged with first street data to provide city wise filtering functionality to the visualization.
  https://www.unitedstateszipcodes.org/zip-code-database/

- **Google states data:** This dataset has information about the latitude and longitude of each state in the US and is merged with first street data to provide state level filtering.
  https://developers.google.com/public-data/docs/canonical/states_csv

Libraries:

The project is created in python3 programming language and following libraries are used:

- **Jupyter-gmaps:** Google maps library to add data layers over the map and plot visualizations.
  https://jupyter-gmaps.readthedocs.io/en/latest/index.html

- **Bokeh:** This library is used to enhance the visualization and add functionalities to it.
  http://docs.bokeh.org/en/latest/

- **Pandas, NumPy:** https://pandas.pydata.org/docs/,https://numpy.org/doc/

- **PyProj:** https://pyproj4.github.io/pyproj/stable/

Code documentation:

Coding for the project involved data cleaning and preprocessing, adding data layers to the map, and plotting visualizations. Following gives a brief overview of each step:

- **Data cleaning and preprocessing**
  The datasets collected from the mentioned sources has a lot of information which was not relevant to the project. Thus, some data cleaning was performed to keep the variables of interest.

```
us_zipcode = pd.read_csv("Datasets/uszips.csv")

us_zipcode = us_zipcode[us_zipcode.state_name.isin(states)]
columns_to_keep = ['zip', 'lat', 'lng', 'city', 'state_id',
'state_name', 'county_name']
us_zipcode = us_zipcode[columns_to_keep].reset_index(drop=True)
us_zipcode.set_index('zip',inplace=True)
```

```
aws_zip = pd.read_csv("Datasets/Zip_level_risk.csv")

aws_zip.rename(columns={'zipcode': 'zip'}, inplace = True)
```

Merging US zip codes and first street data

```
main_dataframe = pd.merge(us_zipcode, aws_zip ,on =
'zip',how='inner')
main_dataframe = main_dataframe[['county_name','zip', 'lat',
'lng', 'city', 'state_name', 'avg_risk_score_all']]
```

Categorizing the NOAA dataset

```
noaa = pd.read_csv("Datasets/NOAA.csv")
scenarios = list(noaa.Scenario.unique())
scenarios_high = [i for i in scenarios if 'HIGH' in i]
scenarios_med = [i for i in scenarios if 'MED' in i]


scenarios_low = [i for i in scenarios if 'LOW' in i]

noaa_high = noaa[noaa.Scenario.isin(scenarios_high)]
noaa_med = noaa[noaa.Scenario.isin(scenarios_med)]
noaa_low = noaa[noaa.Scenario.isin(scenarios_low)]

noaa_high.reset_index(drop=True,inplace=True)
noaa_med.reset_index(drop=True,inplace=True)
noaa_low.reset_index(drop=True,inplace=True)
```

For nationwide flood risk score, WIKIMEDIA map is used from the bokeh library. This map uses "Mercator" projections to plot points.
https://www.britannica.com/science/Mercator-projection

- Calculating Mercator projections and adding them to the dataset. The code has been referenced from https://coderzcolumn.com/tutorials/data-science/plotting-maps-using-bokeh

```
inProj = Proj(init='epsg:3857')
outProj = Proj(init='epsg:4326')
lons, lats = [], []
for lon, lat in list(zip(main_dataframe["lng"],
main_dataframe["lat"])):
    x, y = transform(outProj,inProj,lon,lat)
```

```
    lons.append(x)
    lats.append(y)

main_dataframe["MercatorX"] = lons
main_dataframe["MercatorY"] = lats
```

Finally, exporting the datasets

```
main_dataframe.to_csv("Datasets/SLR_Zipcode.csv")
noaa_high.to_csv("Datasets/NOAA_High.csv")
noaa_med.to_csv("Datasets/NOAA_Med.csv")
noaa_low.to_csv("Datasets/NOAA_Low.csv")
```

Complete code can be found in the notebook "*SLR-DataPreprocessing.ipynb*".

- **Layering and plotting the visualization:**
  For adding data layers to the map and plotting, a function "*plotSeaLevelRisk*" is created which take three arguments i.e., *dataset*, *city*, and *state* name. City and state names are the optional parameters and if they're not passed a nationwide flood risk map is plotted. Passing the state name parameter will filter the results according to the state and passing the city name along with the state will show results for the particular city.
  **Please note that passing only city parameter will not return any plot as city names are not unique and there can be multiple cities with the same name in different states. This is done to avoid ambiguity in visualization.

Adding layer to the WIKIMEDIA map. The *tooltip* argument defines the information to be displayed when hovering over a zip code.

```
wikimedia = get_provider(WIKIMEDIA)
inProj = Proj(init='epsg:3857')
outProj = Proj(init='epsg:4326')

maxRisk = max(df.avg_risk_score_all.values)
minRisk = min(df.avg_risk_score_all.values)

us_lon1, us_lat1 = transform(outProj,inProj,-140,10)
us_lon2, us_lat2 = transform(outProj,inProj,-50,55)
```

```
p = figure(plot_width=1420, plot_height=650,


           x_range=(us_lon1, us_lon2), y_range=(us_lat1,
            us_lat2),
           x_axis_type="mercator", y_axis_type="mercator",
          tooltips=[
                    ("Zipcode", "@zip"),
                    ("Risk Score","@avg_risk_score_all")])

p.add_tile(wikimedia)
```

Adding color bar to the plot to enhance the visualization and color-code the zip codes according to the flood risk score

```
mapper = linear_cmap('avg_risk_score_all', palette,
       minRisk, maxRisk)

p.circle(x="MercatorX", y="MercatorY",
         size=2,
         color=mapper,
         fill_alpha=0.3,
         source=df)

color_bar = ColorBar(color_mapper=mapper['transform'],
                     location=(0,0))

p.add_layout(color_bar, 'right')

show(p)
```

For plotting city and state wise data, google map is used. In order to use the maps an API key needs to be generated. Below link provide the instructions to do that:
https://developers.google.com/maps/documentation/embed/get-api-key
Once the key is generated, copy/paste the key in a text file and save it.

Fetching API key from the file

```
with open('APIKey.txt') as f:
    api_key = f.readline()
    f.close
```

Adding layer and color-bar to the gmap

```
gmap_options = GMapOptions(lat=lat, lng=lng,
    map_type=map_type,
    zoom=zoom)

hover = HoverTool(
    tooltips = [
        ('Zip', '@zip'),
        ('Risk Score', '@avg_risk_score_all'),
    ]
)

p = gmap(api_key, gmap_options,
    title='Flood risk due to Sea Level Rise',
         width=950, height=700,
         tools=[hover, 'reset', 'wheel_zoom', 'pan'])

# Defining data source
source = ColumnDataSource(df)

maxRisk = max(df.avg_risk_score_all.values)
minRisk = min(df.avg_risk_score_all.values)

mapper = linear_cmap('avg_risk_score_all',
    palette, minRisk, maxRisk)

center = p.circle('lng', 'lat', size=size,
    color=mapper, source=source)

color_bar = ColorBar(color_mapper=mapper['transform'],
                     location=(0,0))

p.add_layout(color_bar, 'right')
p.title.text_font_size = "12pt"
show(p)
```

Complete code can be found in the notebook "*SLR-Layering.ipynb*".

For plotting Sea Level Rise in cm using NOAA data, google map is used. The function "*plotSeaLevel*" is created which takes *dataset* and as an argument. Other optional parameters taken by the function along with the complete code can be found in the notebook "*SLR-Layering.ipynb*".

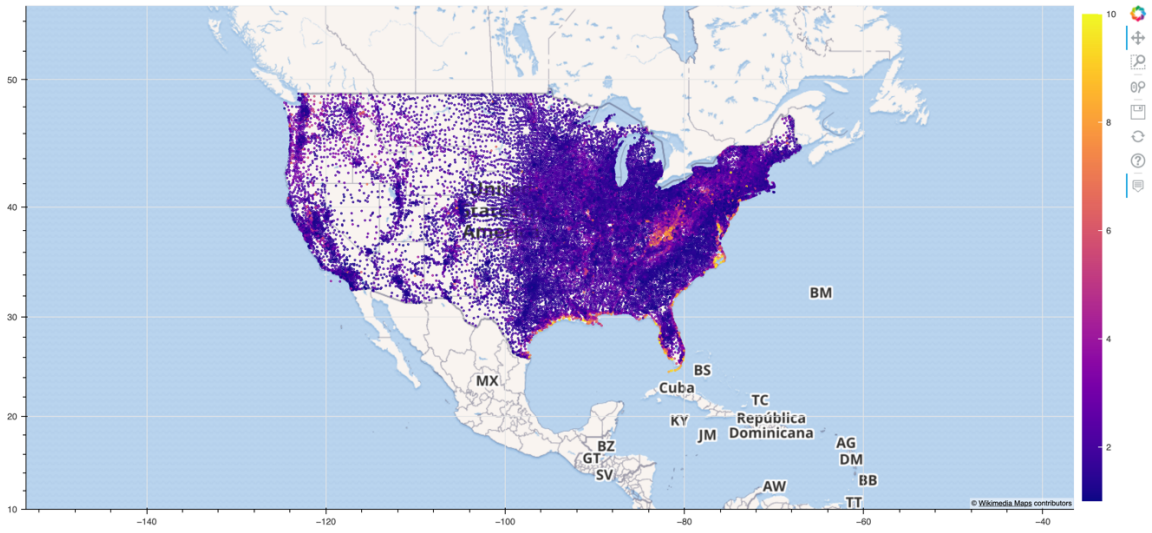
Adding data layer to the gmap for NOAA dataset

```
with open('APIKey.txt') as f:
    api_key = f.readline()
    f.close

gmaps.configure(api_key=api_key)
df = df[['Site', 'Latitude', 'Longitude', 'Scenario',
'RSL in 2020 (cm)']]
df = df.rename(columns={"RSL in 2020 (cm)": "RSL"})
df = df[df.Scenario==scenario]
df.reset_index(drop=True,inplace=True)

site_info = []
for i in range(len(df[['Site','RSL']].values)):
    city = df[['Site','RSL']].values[i][0]
    RSL = df[['Site','RSL']].values[i][1]
    site_info.append({'City':city,'RSL':RSL})

template = """
<dl>
<dt>City</dt><dd>{City}</dd>
<dt>Sea Level Rise(cm)</dt><dd>{RSL}</dd>
</dl>
"""
info = [template.format(**site) for site in site_info]

SLR_layer = gmaps.symbol_layer(df[['Latitude', 'Longitude']],
    fill_color='yellow', stroke_color='yellow',
    stroke_opacity=0.7, fill_opacity=0.8,scale=3,
    info_box_content=info)

fig = gmaps.figure(layout={'width':'900px','height':'700px'},
                   map_type=map_type,
                   center=(lat,lng),zoom_level=zoom)

fig.add_layer(SLR_layer)
return fig
```

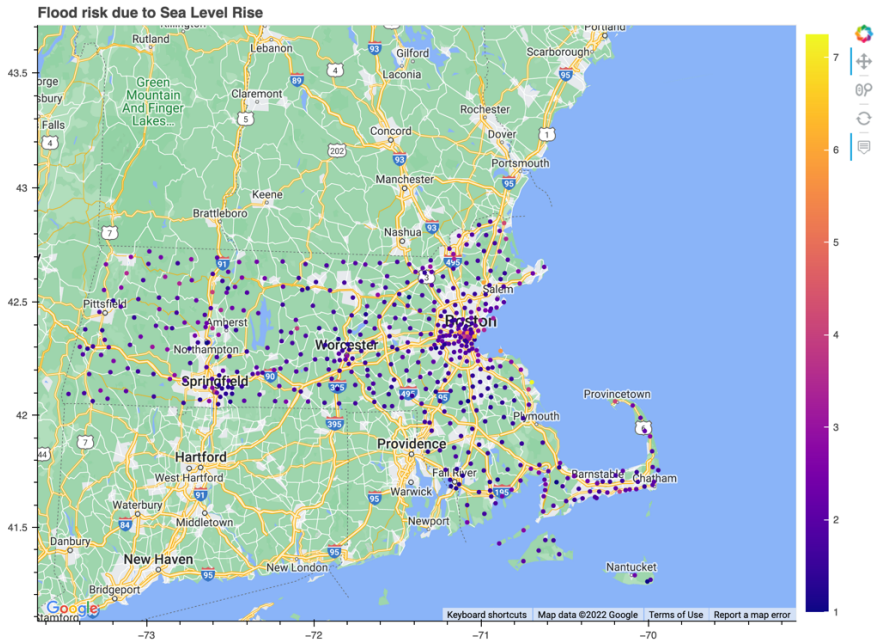
Results:

Following are the results generated from running the defined functions:

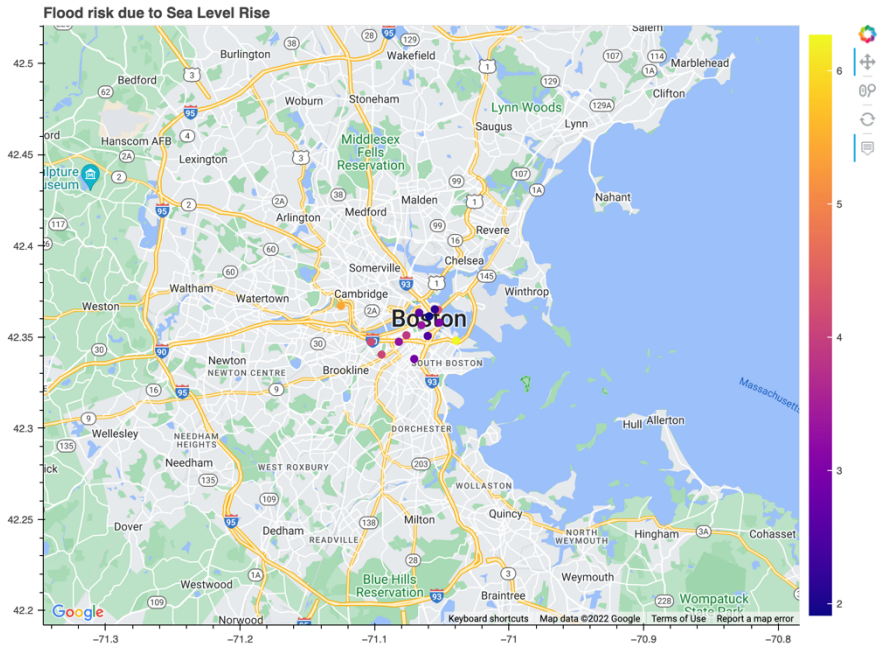
```
p = plotSeaLevelRisk(slr_zip)
```



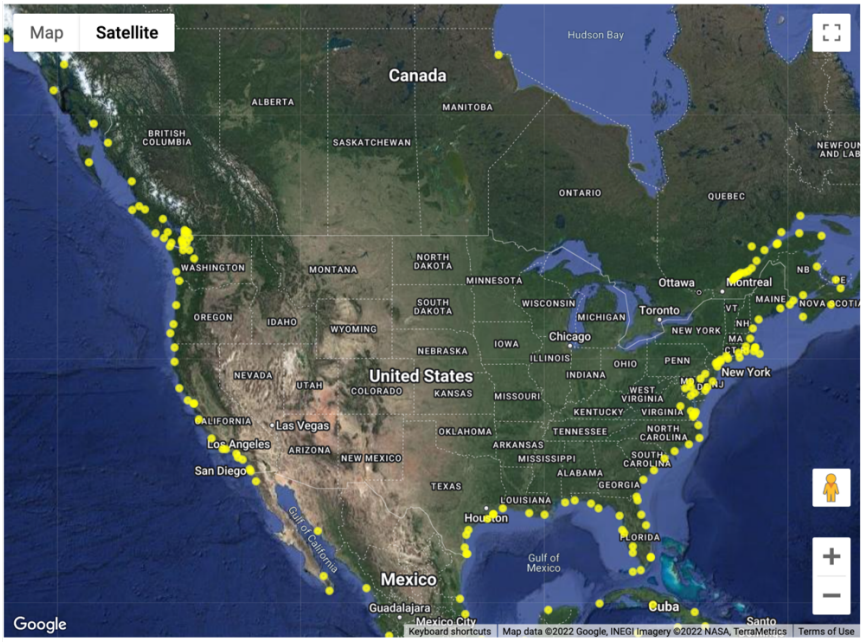

```
p = plotSeaLevelRisk(slr_zip, state="Massachusetts")
```



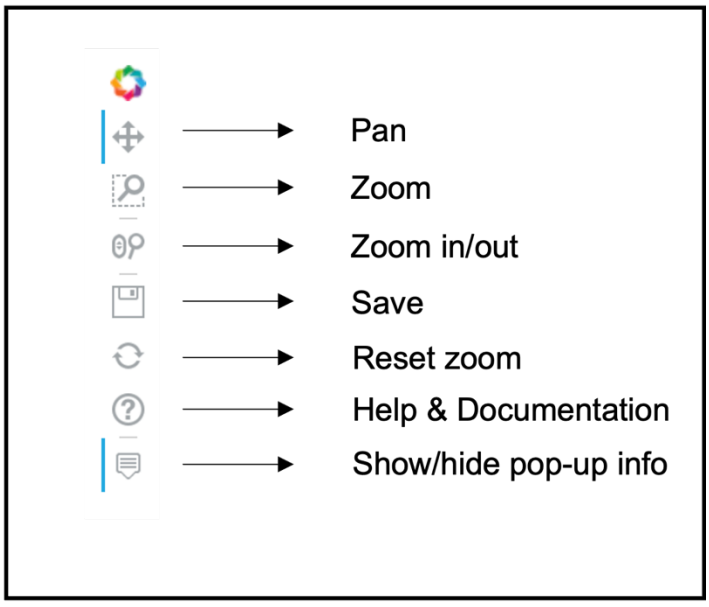
```
p = plotSeaLevelRisk(slr_zip, city="Boston",
    state="Massachusetts")
```




```
fig = plotSeaLevel(noaa, scenario=scenarios[7])
fig
```



The toolbar on the right of the results has the following controls:



Exporting results as HTML. These results can be found under "*Results/*" folder.

```
output_file(filename="FloodRiskSLR.html");
save(p);

output_file(filename="FloodRisk-SLR-Mass.html");
save(p);

output_file(filename="FloodRisk-SLR-Boston.html");
save(p);

embed_minimal_html('SLR.html', views=[fig])
```

**Instructions to open "*SLR.html*"

- Run command *python -m http.server 8080*
- Navigate to *http://0.0.0.0:8080/SLR.html* to view the result.

For other HTML files, no server is required and can be viewed directly in the browser.

References:

- https://thedatafrog.com/en/articles/show-data-google-map-python/
- https://coderzcolumn.com/tutorials/data-science/plotting-maps-using-bokeh
- https://medium.com/future-vision/google-maps-in-python-part-2-393f96196eaf