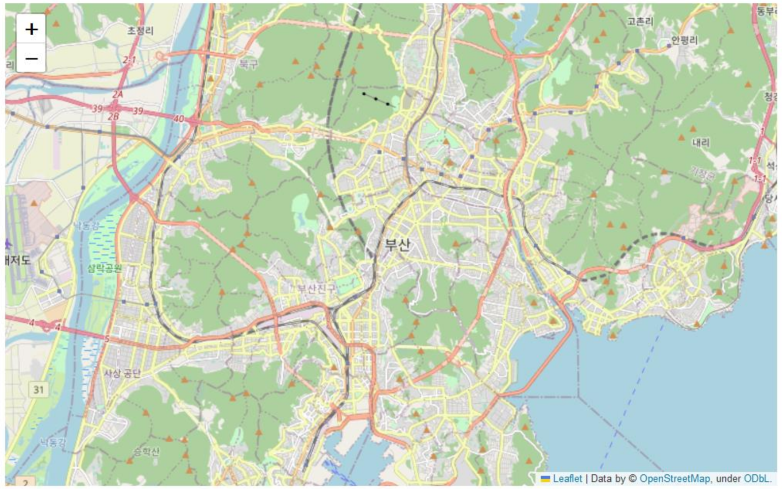
Map Visualization

Create a Busan map

- Import folium and draw a map

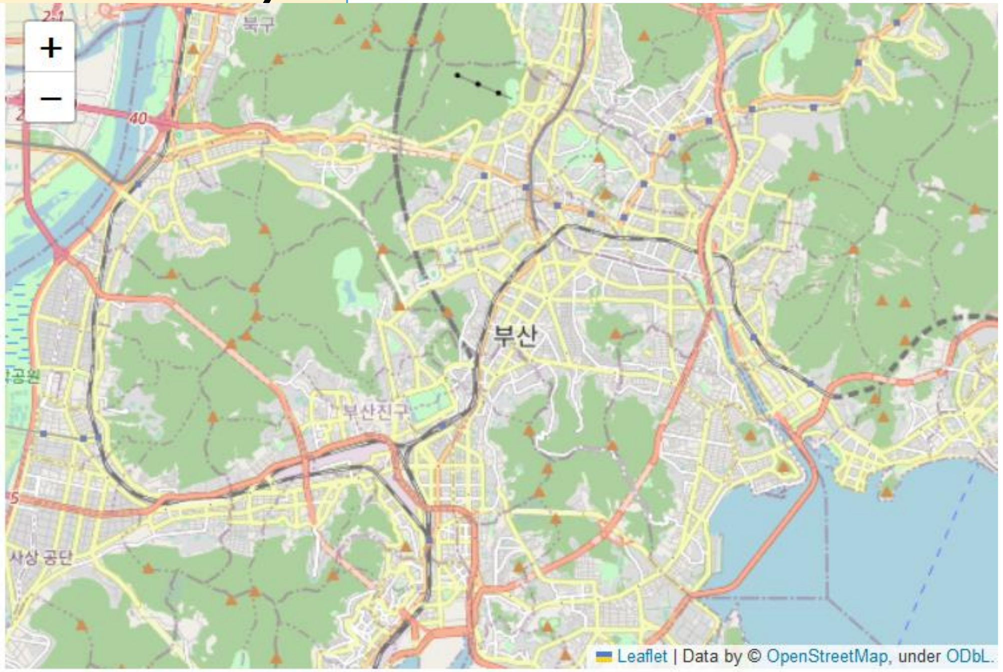
```
import folium

busan =[35.1797957, 129.0727983]
m = folium.Map(location = busan, \
      zoom_start = 12, width=800,height=500)
m
```



Create a Busan map

- Use branca.element

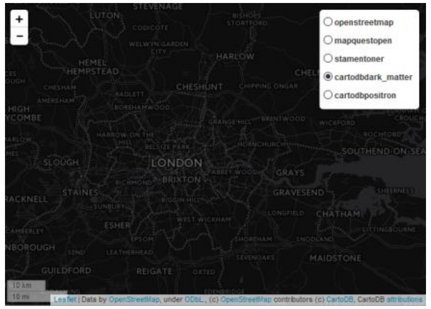
```
from branca.element import Figure
fig = Figure(width=600, height=400)
busan =[35.1797957, 129.0727983]
m = folium.Map(location = busan, zoom_start = 12 )
fig.add_child(m)
```

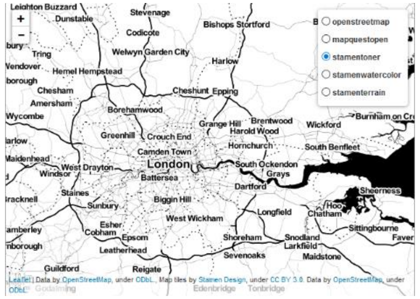


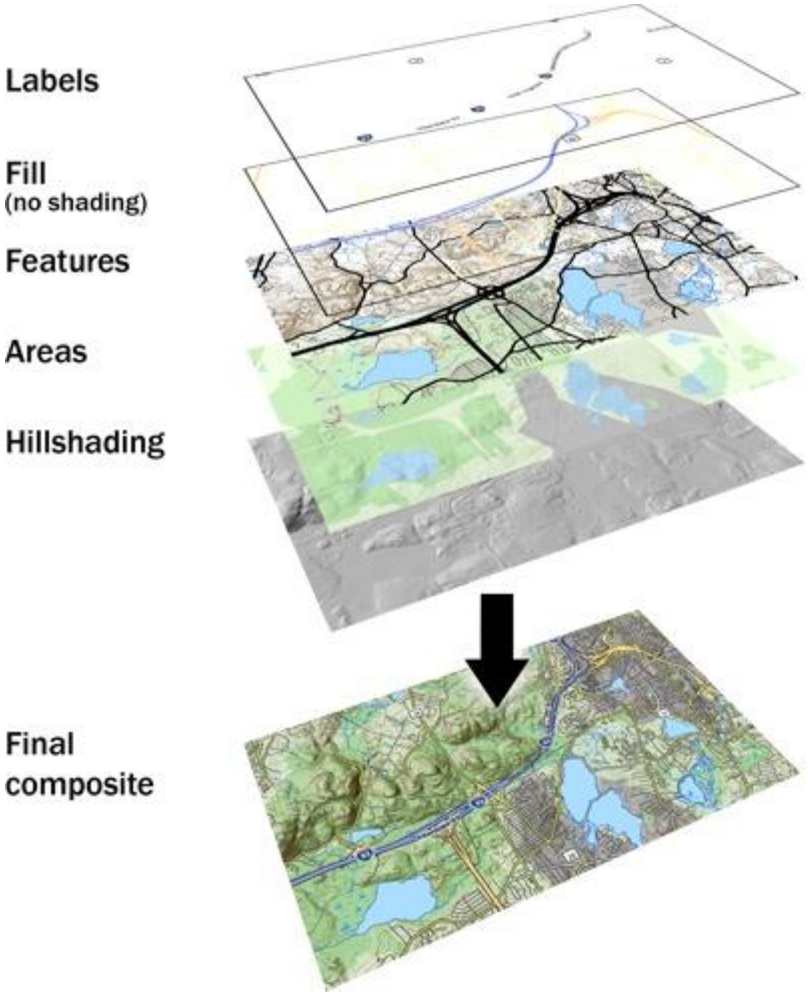
Layers and Tiles in Folium

- Default: OpenStreetMap

https://python-visualization.github.io/folium/latest/user_guide/raster_layers/tiles.html







Labels

Fill
(no shading)

Features

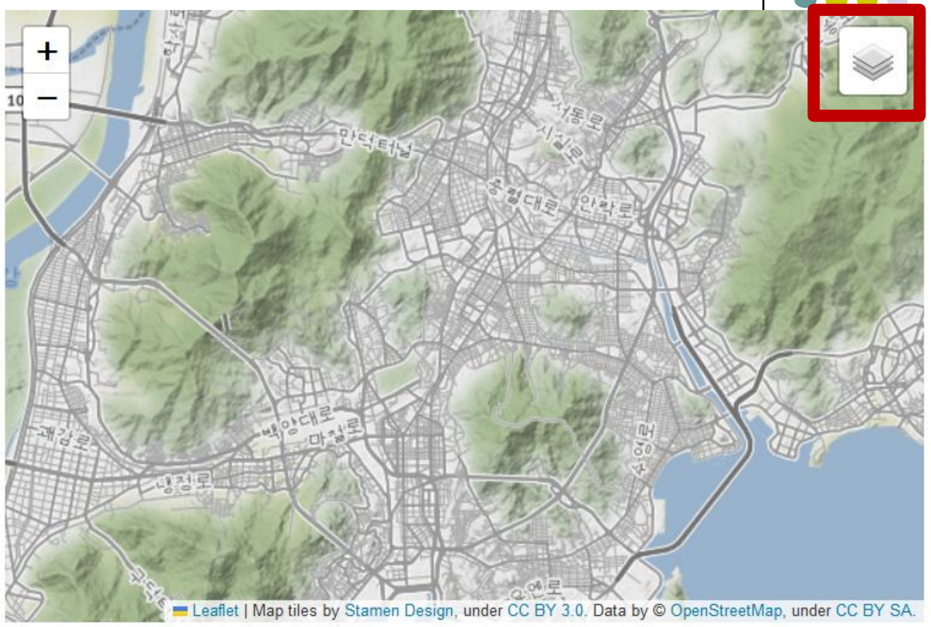
Areas

Hillshading

Final
composite

Layers and Tiles in Folium

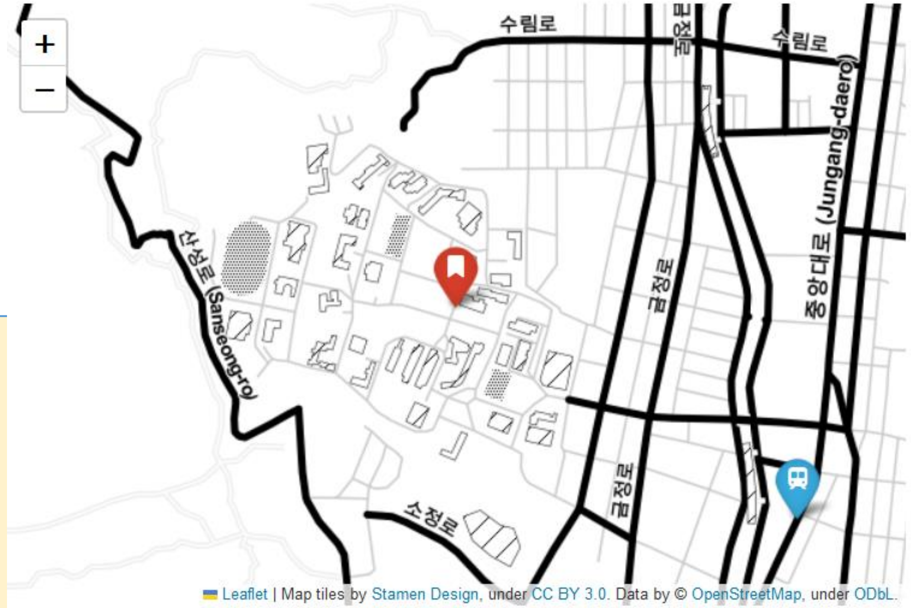
- Default: OpenStreetMap


+
−
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under CC BY SA.

```
fig2 = Figure(width=600, height=400)
busan =[35.1797957, 129.0727983]
m2 = folium.Map(location = busan, zoom_start = 12)
folium.TileLayer('Stamen Terrain').add_to(m2)
folium.TileLayer('Stamen Toner').add_to(m2)
folium.TileLayer('Stamen Water Color').add_to(m2)
folium.TileLayer('cartodbpositron').add_to(m2)
folium.TileLayer('cartodbdark_matter').add_to(m2)
folium.LayerControl().add_to(m2)
fig2.add_child(m2)
```

Plotting Marker on the map

- Use Marker()
  - Options: Icon, popup, tooltip


+
−
수림로
수림로
산성로 (Sanseong-ro)
중앙대로 (Jungang-daero)
금정로
금정로
소정로
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by © OpenStreetMap, under ODbL.

```
fig3 = Figure(width=600, height=400)
# Infos
pnu = [35.23379098528912, 129.08090997889803]
pnu_station=[35.22884833731382, 129.09063225747852]
# map
m3 = folium.Map( pnu, zoom_start=15, tiles='Stamen Toner')

# Marker for PNU
folium.Marker(pnu, popup='PNU', \
              tooltip='Click here to see Popup', \
              icon=folium.Icon(icon='bookmark', color='red', prefix='fa')).add_to(m3)

# Makrer for PNU subway
folium.Marker(pnu_station, popup='PNU Subwary', \
              icon=folium.Icon(icon='subway', color='blue', prefix='fa')).add_to(m3)

fig3.add_child(m3)
```

Marker Cluster (1/2)

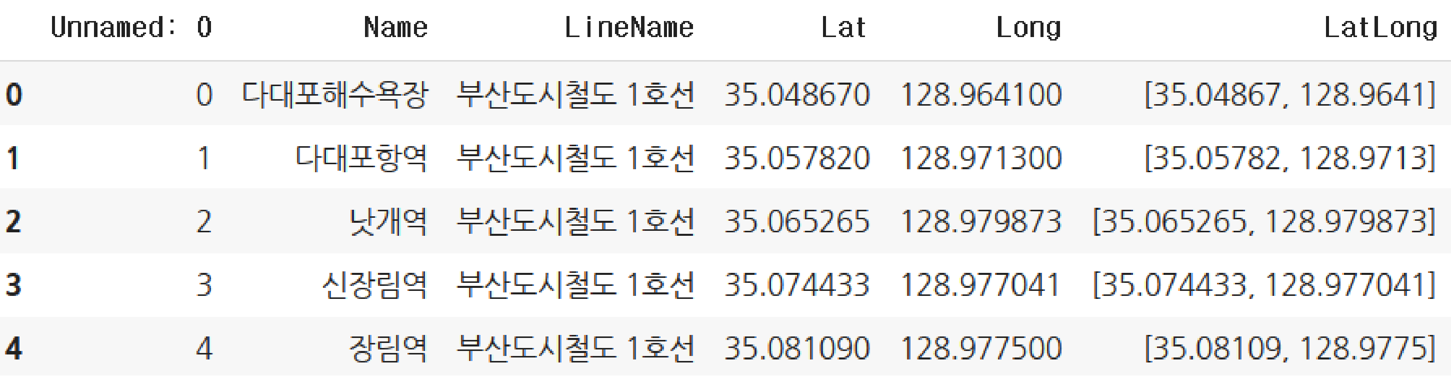
- Import MarkerCluster

```
import pandas as pd
import folium
from folium import Marker
from folium.plugins import MarkerCluster
```

- Load data

```
from google.colab import files
file_uploaded = files.upload()
```

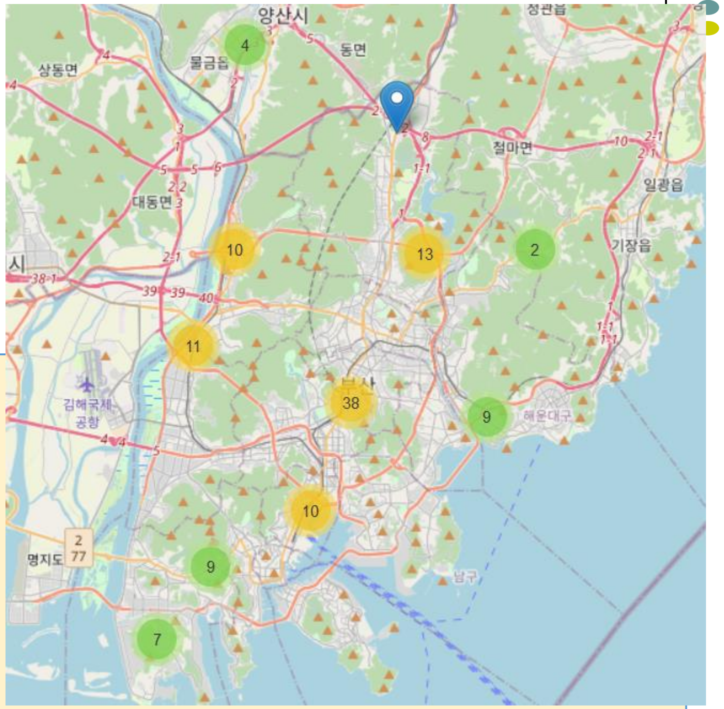
```
df = pd.read_csv('subway.csv')
df.head()
```



	Unnamed: 0	Name	LineName	Lat	Long	LatLong
0	0	다대포해수욕장	부산도시철도 1호선	35.048670	128.964100	[35.04867, 128.9641]
1	1	다대포항역	부산도시철도 1호선	35.057820	128.971300	[35.05782, 128.9713]
2	2	낫개역	부산도시철도 1호선	35.065265	128.979873	[35.065265, 128.979873]
3	3	신장림역	부산도시철도 1호선	35.074433	128.977041	[35.074433, 128.977041]
4	4	장림역	부산도시철도 1호선	35.081090	128.977500	[35.08109, 128.9775]

Marker Cluster (2/2)

- Draw a map with MarkerCluster



```
fig = Figure(width=1024, height=768)
submap = folium.Map(location = busan, \
                zoom_start = 13)

cluster = MarkerCluster()
for _, i in df.iterrows():
    cluster.add_child(
        Marker(location = [i['Lat'], i['Long']],
               popup =folium.Popup("<pre>" + "LineName: " + str(i['LineName']) \
                  + "<br>" + "StationName: " + str(i['Name']) + " <br>" + "</pre>", \
                  max_width=300,min_width=300))
    ).add_to(submap)

fig.add_child(submap)
```

Add a GeoJSON file for boundaries (1/3)

- Import packages

```
import pandas as pd
import folium
from folium import Marker
from folium.plugins import MarkerCluster
import json
```

- Load a GeoJson file

```
from google.colab import files
file_uploaded = files.upload()
```

```
busan_geojson = json.load(open('busan_gu.json', encoding='utf-8'))
```

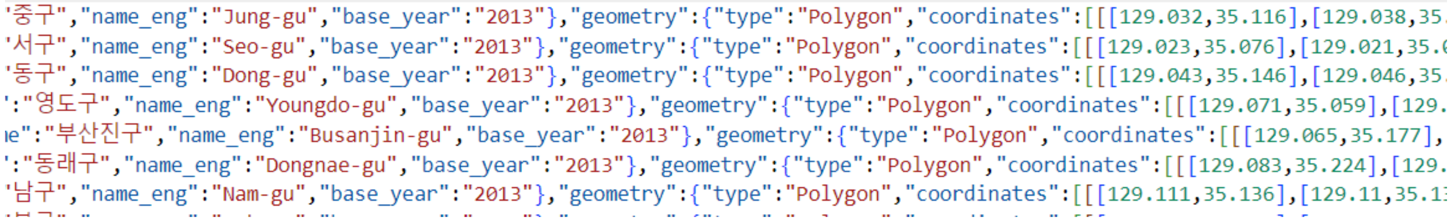
Busan gu Geojson file is downloaded from https://mwna40000.tistory.com/34

Add a GeoJSON file for boundaries (2/3)

- GeJson example

```
busan_geojson["features"][0]
```

```
{'type': 'Feature',
 'id': '중구',
 'properties': {'code': '21010',
  'name': '중구',
  'name_eng': 'Jung-gu',
  'base_year': '2013'},
 'geometry': {'type': 'Polygon',
  'coordinates': [[[129.032, 35.116],
    [129.038, 35.112],    [129.042, 35.111],
    [129.041, 35.108],    [129.038, 35.104],
    [129.038, 35.098],    [129.037, 35.097],
    [129.029, 35.096],    [129.026, 35.096],
    [129.024, 35.1],    [129.022, 35.102],
    [129.021, 35.106],    [129.023, 35.109],
    [129.024, 35.109],    [129.026, 35.111],
    [129.028, 35.11],    [129.028, 35.115],
    [129.032, 35.116]]]}}
```

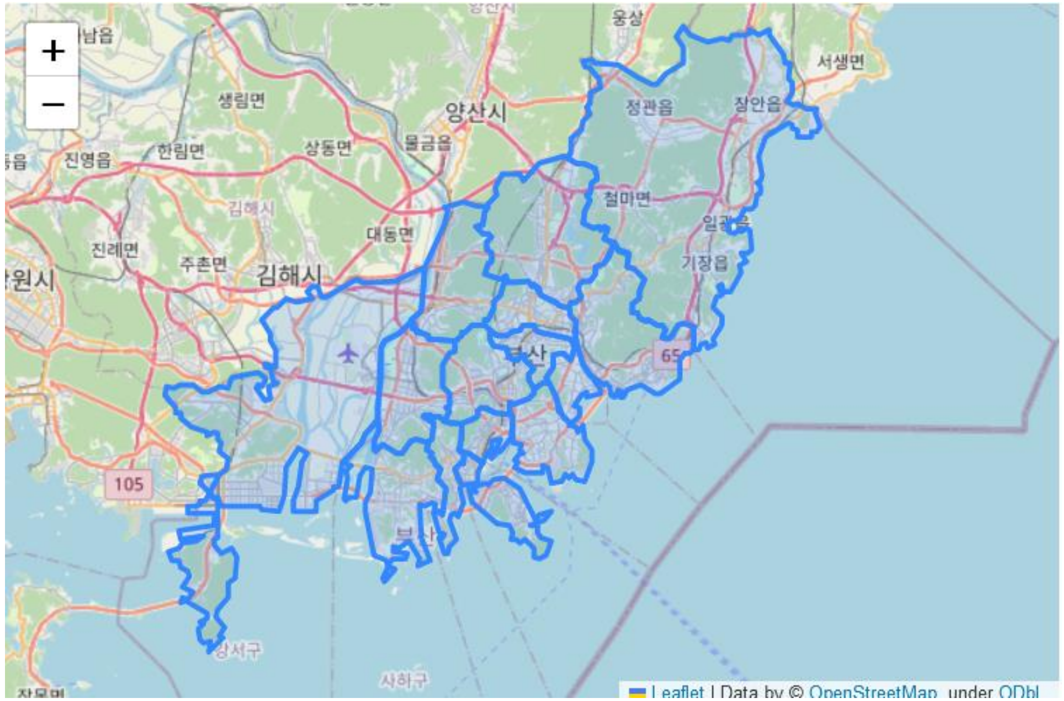


```
"중구","name_eng":"Jung-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.032,35.116],[129.038,35.
"서구","name_eng":"Seo-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.023,35.076],[129.021,35.
"동구","name_eng":"Dong-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.043,35.146],[129.046,35.
":"영도구","name_eng":"Youngdo-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.071,35.059],[129.
ie":"부산진구","name_eng":"Busanjin-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.065,35.177],
":"동래구","name_eng":"Dongnae-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.083,35.224],[129.
"남구","name_eng":"Nam-gu","base_year":"2013"},"geometry":{"type":"Polygon","coordinates":[[[129.111,35.136],[129.11,35.1
```

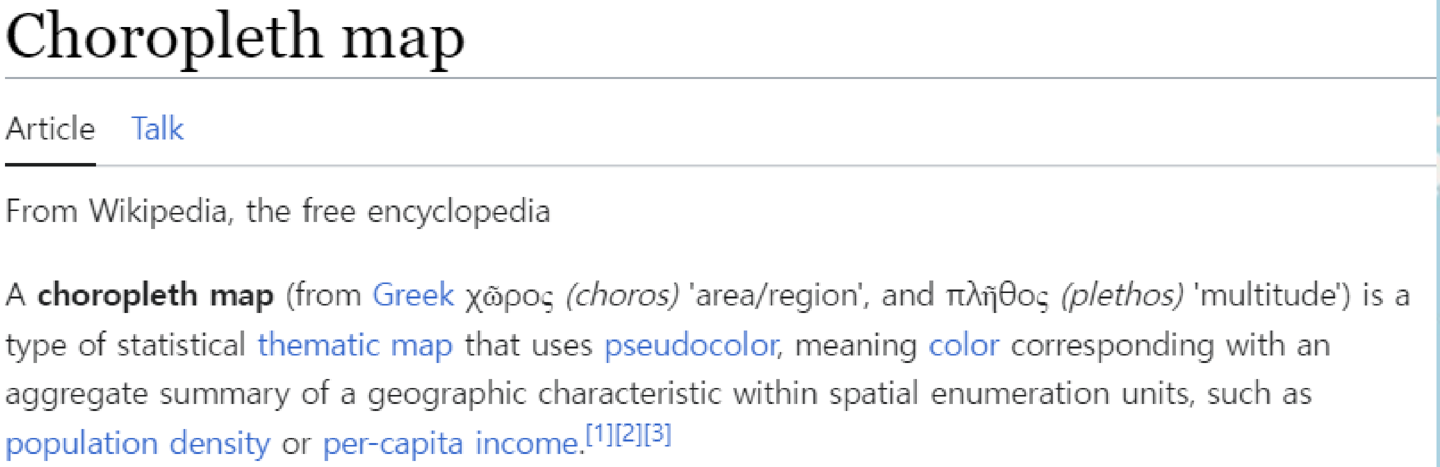
Add a GeoJSON file for boundaries (3/3)

- Draw a map

```
fig3 = Figure(width=600, height=400)
busan =[35.1797957, 129.0727983]
m = folium.Map(location=busan, zoom_start=10)
folium.GeoJson(busan_geojson).add_to(m)

fig3.add_child(m)
```



Draw a choropleth map (1/3)

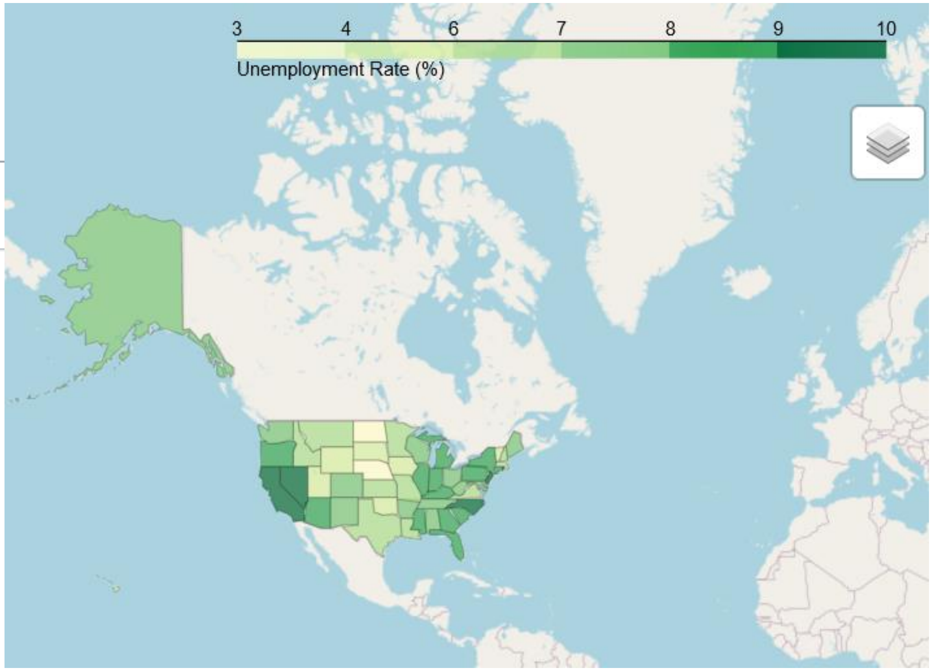


Choropleth map

Article Talk

From Wikipedia, the free encyclopedia

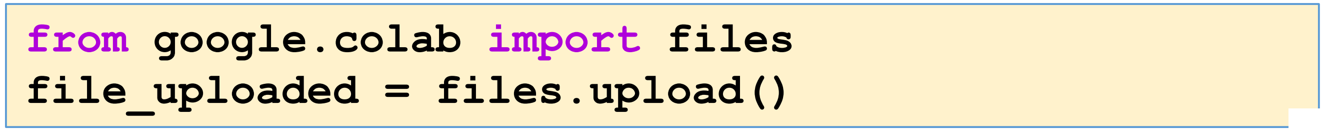
A **choropleth map** (from Greek χῶρος *(choros)* 'area/region', and πλῆθος *(plethos)* 'multitude') is a type of statistical thematic map that uses pseudocolor, meaning color corresponding with an aggregate summary of a geographic characteristic within spatial enumeration units, such as population density or per-capita income.[1][2][3]



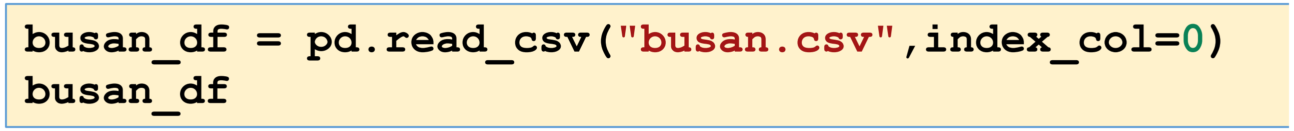
- provide an easy way to visualize how a variable varies across a geographic area or show the level of variability within a region

Draw a choropleth map (2/3)

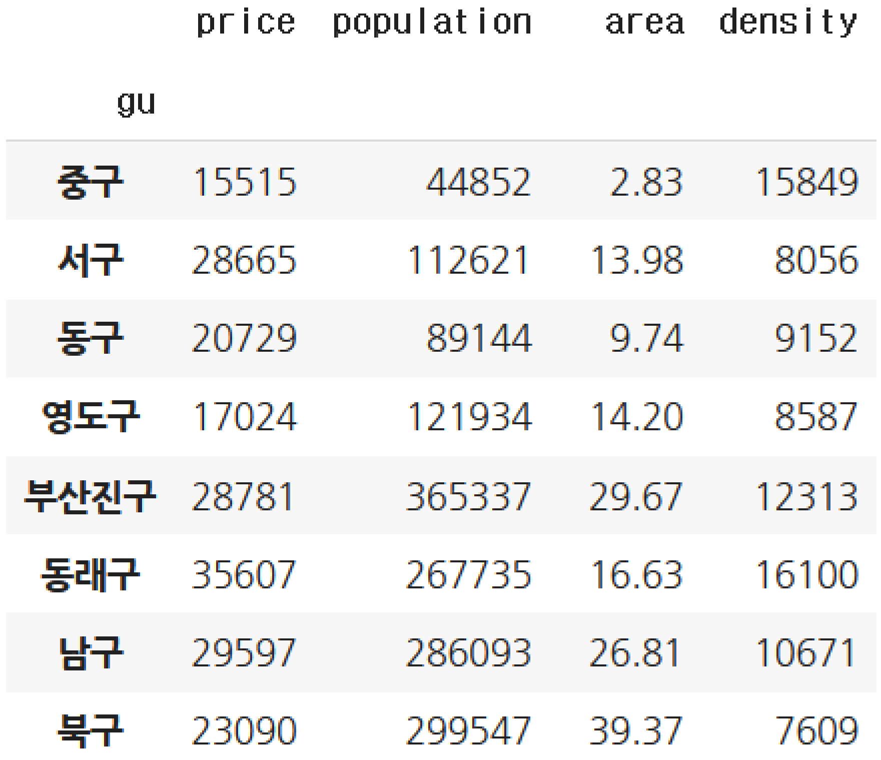
- 1. Prepare Data



```
from google.colab import files
file_uploaded = files.upload()
```



```
busan_df = pd.read_csv("busan.csv",index_col=0)
busan_df
```

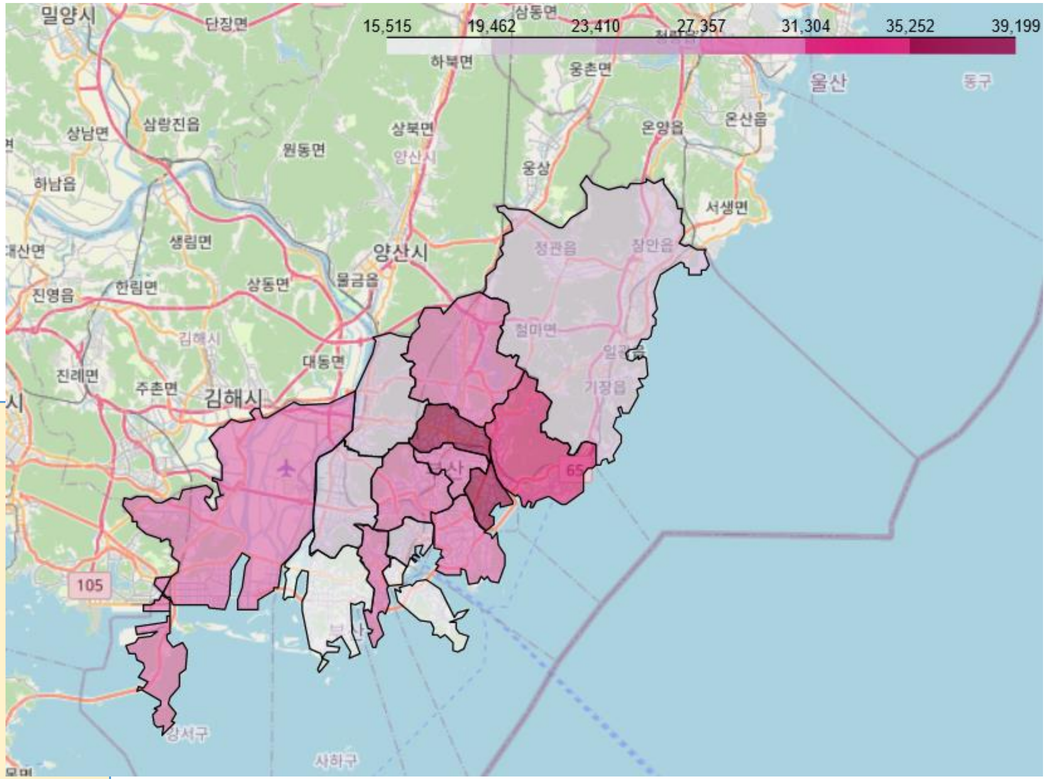


	price	population	area	density
gu				
중구	15515	44852	2.83	15849
서구	28665	112621	13.98	8056
동구	20729	89144	9.74	9152
영도구	17024	121934	14.20	8587
부산진구	28781	365337	29.67	12313
동래구	35607	267735	16.63	16100
남구	29597	286093	26.81	10671
북구	23090	299547	39.37	7609

Draw a choropleth map (3/3)

- 2. Draw a map

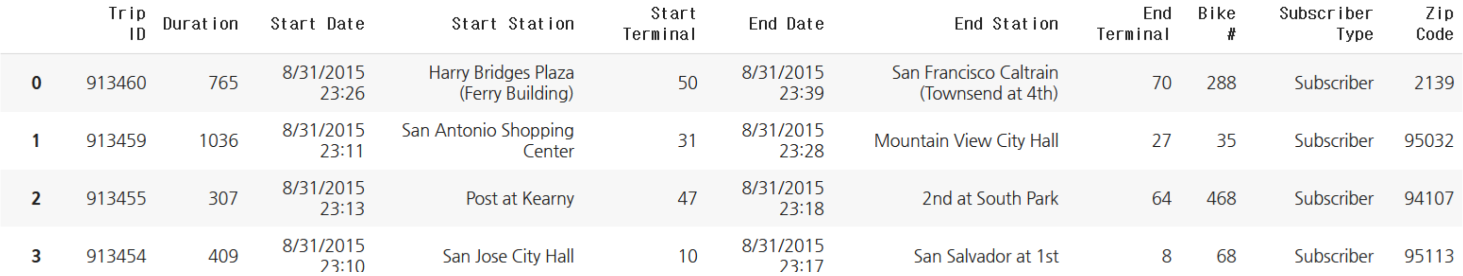
```
fig = Figure(width=800, height=600)
busan =[35.1797957, 129.0727983]
busan_map = folium.Map(location=busan, zoom_start=10)

folium.Choropleth(geo_data=busan_geojson,
                  data=busan_df["price"],
                  columns =[df.index, df["price"]],
                  fill_color="PuRd",
                  key_on="feature.id").add_to(busan_map)

fig.add_child(busan_map)
```



Bike Sharing simulation

Bay Area Bike Share (1/2)

- Attributes
  - An ID for the rental
  - Duration of the rental, in seconds
  - Start date
  - Name of the Start Station and code for Start Terminal
  - Name of the End Station and code for End Terminal
  - A serial number for the bike
  - Subscriber type and zip code

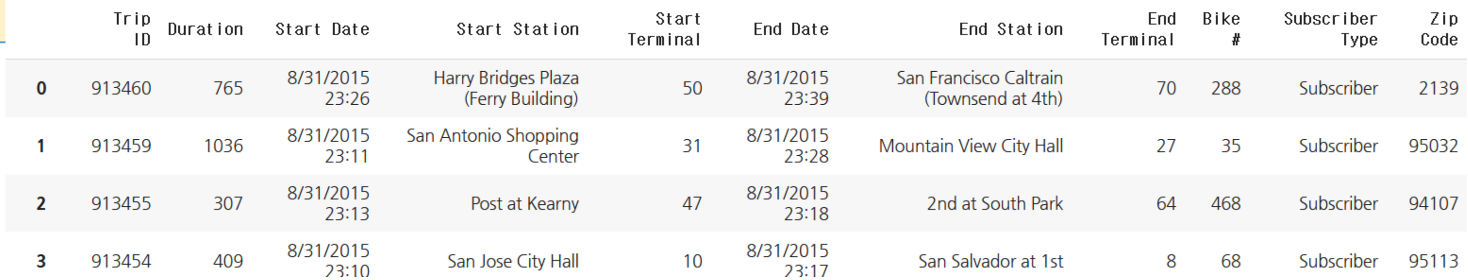


	Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
0	913460	765	8/31/2015 23:26	Harry Bridges Plaza (Ferry Building)	50	8/31/2015 23:39	San Francisco Caltrain (Townsend at 4th)	70	288	Subscriber	2139
1	913459	1036	8/31/2015 23:11	San Antonio Shopping Center	31	8/31/2015 23:28	Mountain View City Hall	27	35	Subscriber	95032
2	913455	307	8/31/2015 23:13	Post at Kearny	47	8/31/2015 23:18	2nd at South Park	64	468	Subscriber	94107
3	913454	409	8/31/2015 23:10	San Jose City Hall	10	8/31/2015 23:17	San Salvador at 1st	8	68	Subscriber	95113

Bay Area Bike Share (2/2)

```
import pandas as pd
import matplotlib
matplotlib.use('Agg')
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import numpy as np
```

```
url = "https://raw.githubusercontent.com/data-8/textbook/main/assets/data/trip.csv"
trip = pd.read_csv(url)
trip
```

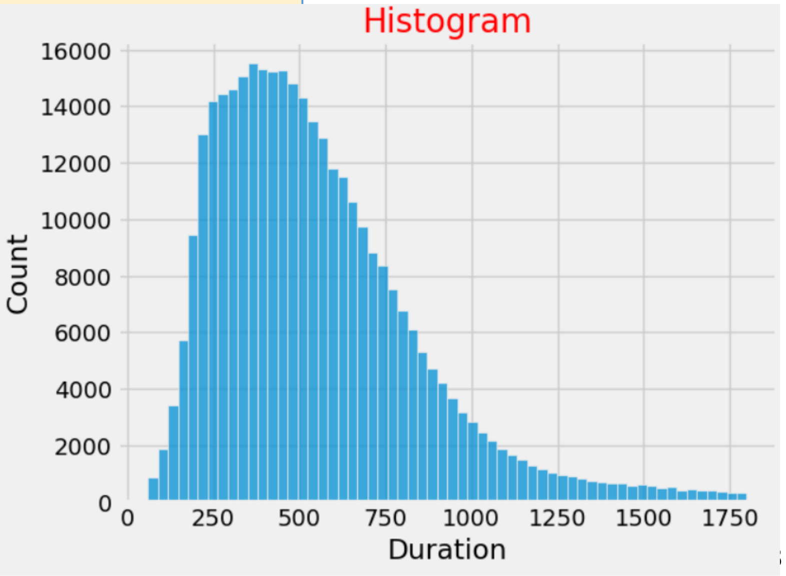


	Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
0	913460	765	8/31/2015 23:26	Harry Bridges Plaza (Ferry Building)	50	8/31/2015 23:39	San Francisco Caltrain (Townsend at 4th)	70	288	Subscriber	2139
1	913459	1036	8/31/2015 23:11	San Antonio Shopping Center	31	8/31/2015 23:28	Mountain View City Hall	27	35	Subscriber	95032
2	913455	307	8/31/2015 23:13	Post at Kearny	47	8/31/2015 23:18	2nd at South Park	64	468	Subscriber	94107
3	913454	409	8/31/2015 23:10	San Jose City Hall	10	8/31/2015 23:17	San Salvador at 1st	8	68	Subscriber	95113

Histogram

- focus only on the free trips
  - trips that last less than 1800 seconds (half an hour)

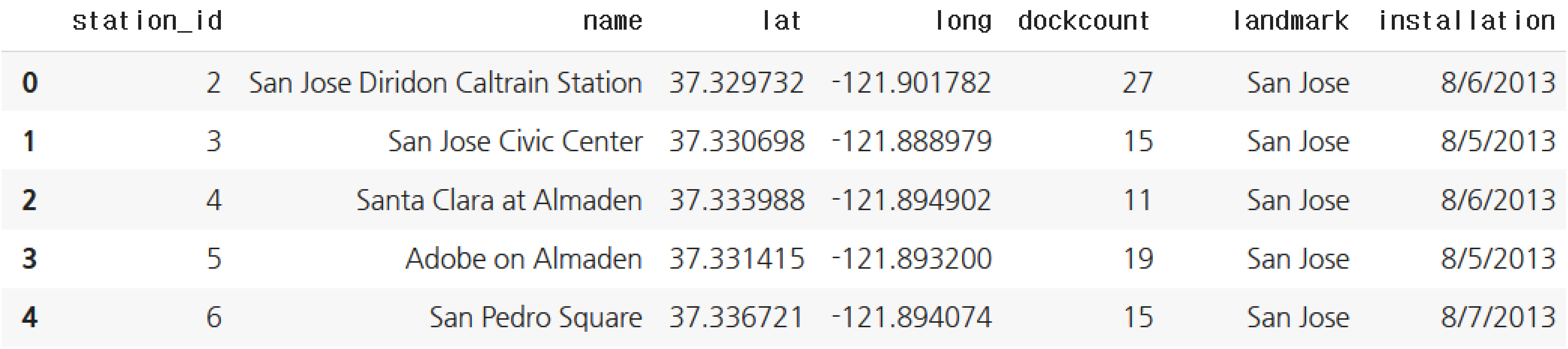
```
commute = trip[trip.Duration < 1800]
fig, ax = plt.subplots()
ax=sns.histplot(data=commute, x = "Duration", bins=60)
ax.set_xlabel("Duration")
ax.set_ylabel("Count")
ax.set_title("Histogram", size=20, color="red")
plt.show()
```



Drawing a station map (1/3)

- 1. Prepare stations’ geographical info

```
url = "https://raw.githubusercontent.com/data-8/textbook/main/assets/data/station.csv"
stations = pd.read_csv(url)
stations
```

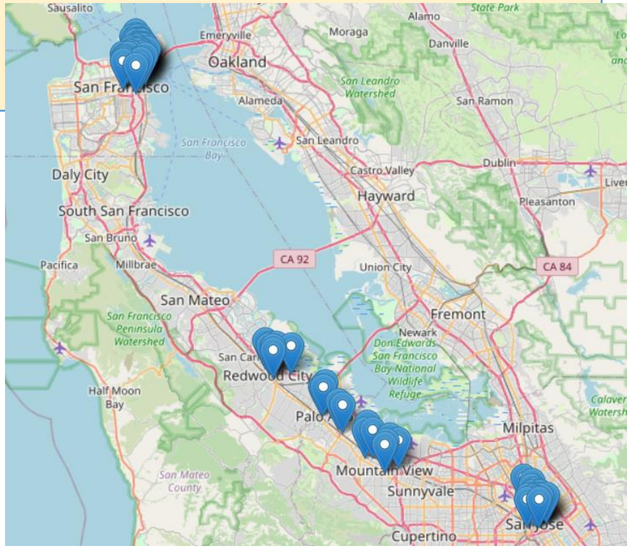


	station_id	name	lat	long	dockcount	landmark	installation
0	2	San Jose Diridon Caltrain Station	37.329732	-121.901782	27	San Jose	8/6/2013
1	3	San Jose Civic Center	37.330698	-121.888979	15	San Jose	8/5/2013
2	4	Santa Clara at Almaden	37.333988	-121.894902	11	San Jose	8/6/2013
3	5	Adobe on Almaden	37.331415	-121.893200	19	San Jose	8/5/2013
4	6	San Pedro Square	37.336721	-121.894074	15	San Jose	8/7/2013

Drawing a station map (2/3)

- 2. Draw a map with markers

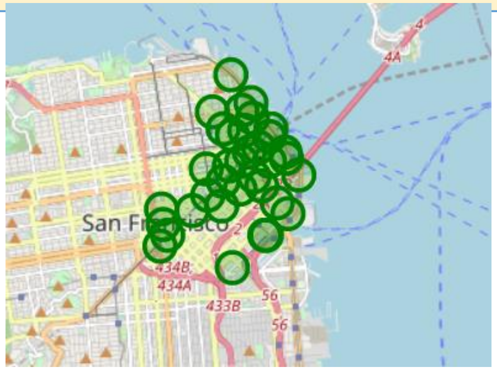
```
import folium

fig = Figure(width=800, height=600)
cityMap = folium.Map(location = [stations['lat'].mean(axis='rows’), \
      stations['long'].mean(axis='rows’)], zoom_start = 10)
makerPositions = stations[['lat', 'long']].values.tolist()
for markerPosition in makerPositions:
    folium.Marker(markerPosition).add_to(cityMap)
fig.add_child(cityMap)
```



Drawing a station map (3/3)

- Draw a map representing points on a map by colored circles.

```
fig = Figure(width=600, height=400)
sf = stations[stations['landmark'] == 'San Francisco']
sf_map_data = folium.Map(location = [sf['lat'].mean(axis='rows’), \
      sf['long'].mean(axis='rows')],zoom_start = 12)
makerPositions = stations[['lat', 'long']].values.tolist()

for markerPosition in makerPositions:
    folium.CircleMarker(markerPosition, fill = True,
                       color = 'green').add_to(sf_map_data)
fig.add_child(sf_map_data)
```

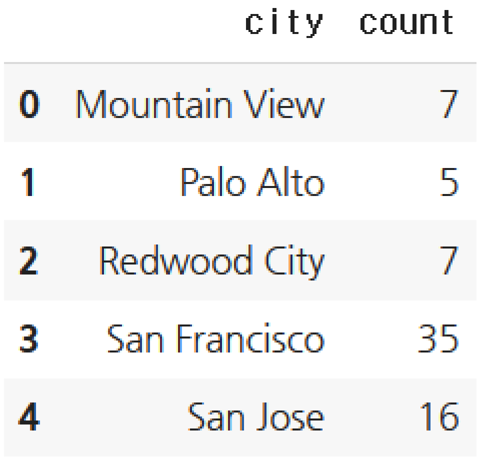


More Informative Maps: An Application of join (1/4)

- Use group to identify all the cities

```
cities = stations.groupby('landmark')['landmark'].count()
cities
```

```
cities = stations.groupby('landmark')['landmark'].count().reset_index
(name='count').rename(columns={'landmark': 'city'})
cities
```

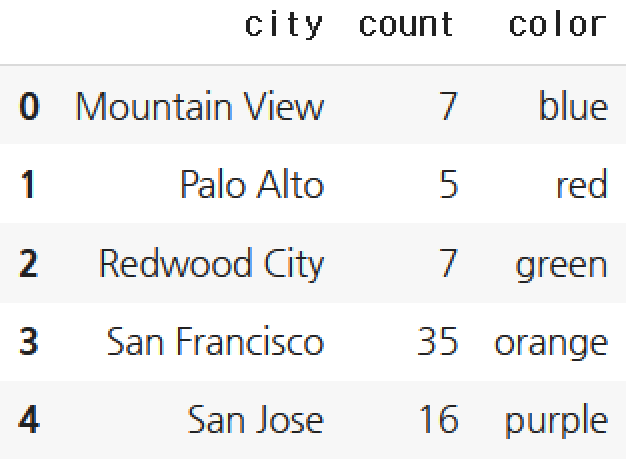


	city	count
0	Mountain View	7
1	Palo Alto	5
2	Redwood City	7
3	San Francisco	35
4	San Jose	16

More Informative Maps: An Application of join (2/4)

- Copy cities to colors then a new columns

```
colors = cities.copy()
colors['color'] = np.array(['blue', 'red', 'green', 'orange', 'purple'])
colors
```

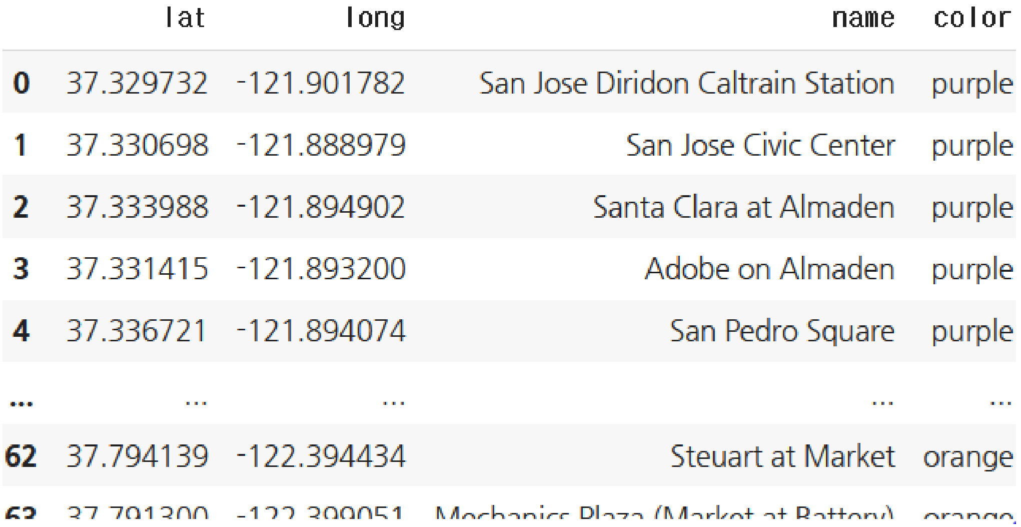


	city	count	color
0	Mountain View	7	blue
1	Palo Alto	5	red
2	Redwood City	7	green
3	San Francisco	35	orange
4	San Jose	16	purple

More Informative Maps: An Application of join (3/4)

- join stations and colors by landmark

```
# JOIN table stations and colors
joined = stations.join(colors.set_index('city'),
                       on='landmark',
                       how='inner')
colored = joined.loc[:, ['lat', 'long', 'name', 'color']]
colored
```

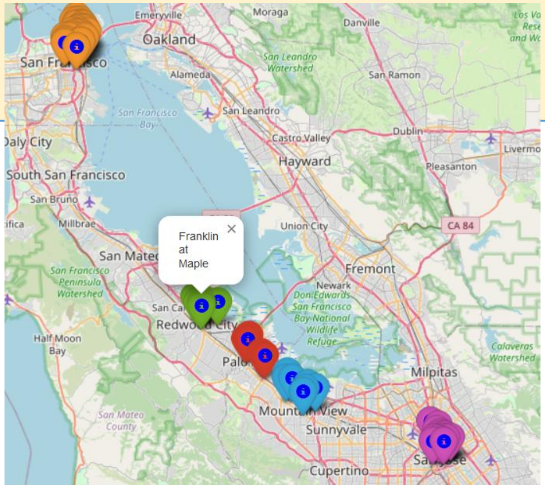


	lat	long	name	color
0	37.329732	-121.901782	San Jose Diridon Caltrain Station	purple
1	37.330698	-121.888979	San Jose Civic Center	purple
2	37.333988	-121.894902	Santa Clara at Almaden	purple
3	37.331415	-121.893200	Adobe on Almaden	purple
4	37.336721	-121.894074	San Pedro Square	purple
...	...	...	...	...
62	37.794139	-122.394434	Steuart at Market	orange
63	37.791300	-122.399051	Mechanics Plaza (Market at Battery)	orange

More Informative Maps: An Application of join (4/4)

- Draw joined data

```
fig = Figure(width=800, height=600)

colored_map = folium.Map(location = [colored['lat'].mean(axis='rows’),\
      colored['long'].mean(axis='rows')],zoom_start = 10)

for markerData in colored.values:
    folium.Marker([markerData[0],markerData[1]] ,
      icon=folium.Icon(color=markerData[3],icon_color='blue'),
      popup=markerData[2]).add_to(colored_map)
fig.add_child(colored_map)
```



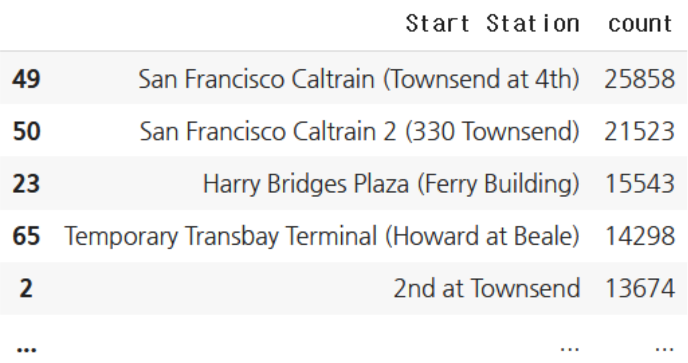
Most popular bike rental station (1/6)

- Make a group

```
starts = commute.groupby('Start Station')['Start Station'].count()
starts
```

- Make a dataframe with group by and sorting

```
starts = commute.groupby('Start Station')['Start Station'].count().
reset_index(name='count').sort_values(by='count', ascending=False)
starts
```

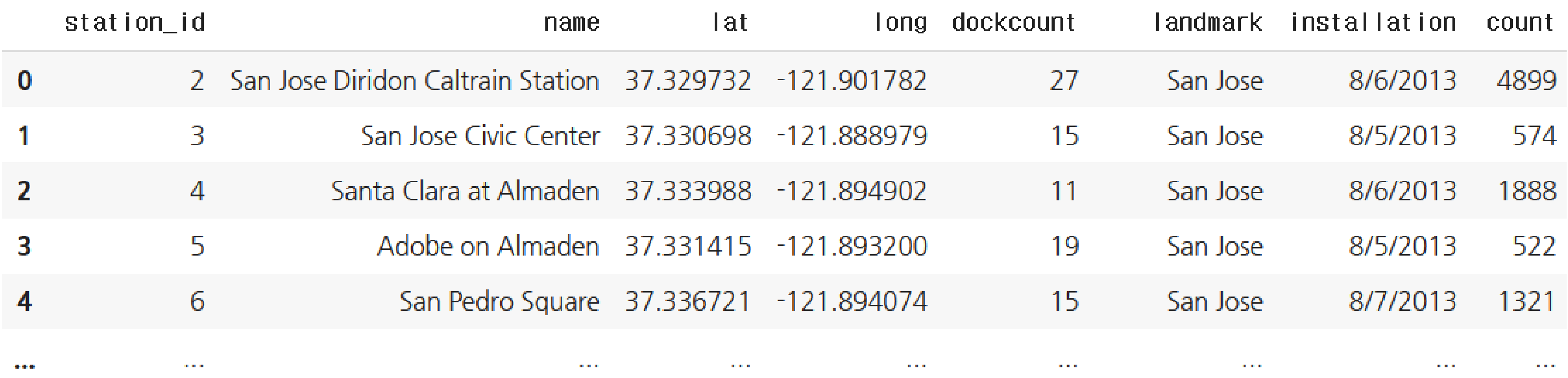


	Start Station	count
49	San Francisco Caltrain (Townsend at 4th)	25858
50	San Francisco Caltrain 2 (330 Townsend)	21523
23	Harry Bridges Plaza (Ferry Building)	15543
65	Temporary Transbay Terminal (Howard at Beale)	14298
2	2nd at Townsend	13674
...	...	...

Most popular bike rental station (2/6)

- Include geo data to start station by join

```
station_starts = stations.join(starts.set_index('Start Station'),
                               on='name',
                               how='inner')

station_starts
```



	station_id	name	lat	long	dockcount	landmark	installation	count
0	2	San Jose Diridon Caltrain Station	37.329732	-121.901782	27	San Jose	8/6/2013	4899
1	3	San Jose Civic Center	37.330698	-121.888979	15	San Jose	8/5/2013	574
2	4	Santa Clara at Almaden	37.333988	-121.894902	11	San Jose	8/6/2013	1888
3	5	Adobe on Almaden	37.331415	-121.893200	19	San Jose	8/5/2013	522
4	6	San Pedro Square	37.336721	-121.894074	15	San Jose	8/7/2013	1321
...	...	...	...	...	...	...	...	...

Most popular bike rental station (3/6)

- Adding a color and an area size
  - will use a circle marker, the computed area size means the size of a circle

```
# Extract columns 'lat', 'long', 'name' from station_starts to starts_map_data
starts_map_data = station_starts.loc[:, ['lat', 'long', 'name']].copy()

# Set color
starts_map_data['colors'] = ['blue'] * 68 # 68 rows

# Set size
starts_map_data['areas'] = station_starts['count'] * 0.3

starts_map_data
```

Most popular bike rental station (4/6)

- Selector a color based on an area size

```
def color_select(areas):
    if areas > 3000:
        return 'red'
    elif areas > 2000:
        return  'yellow'
    elif areas > 1000:
        return 'green'
    else:
        return 'dodgerblue'
```

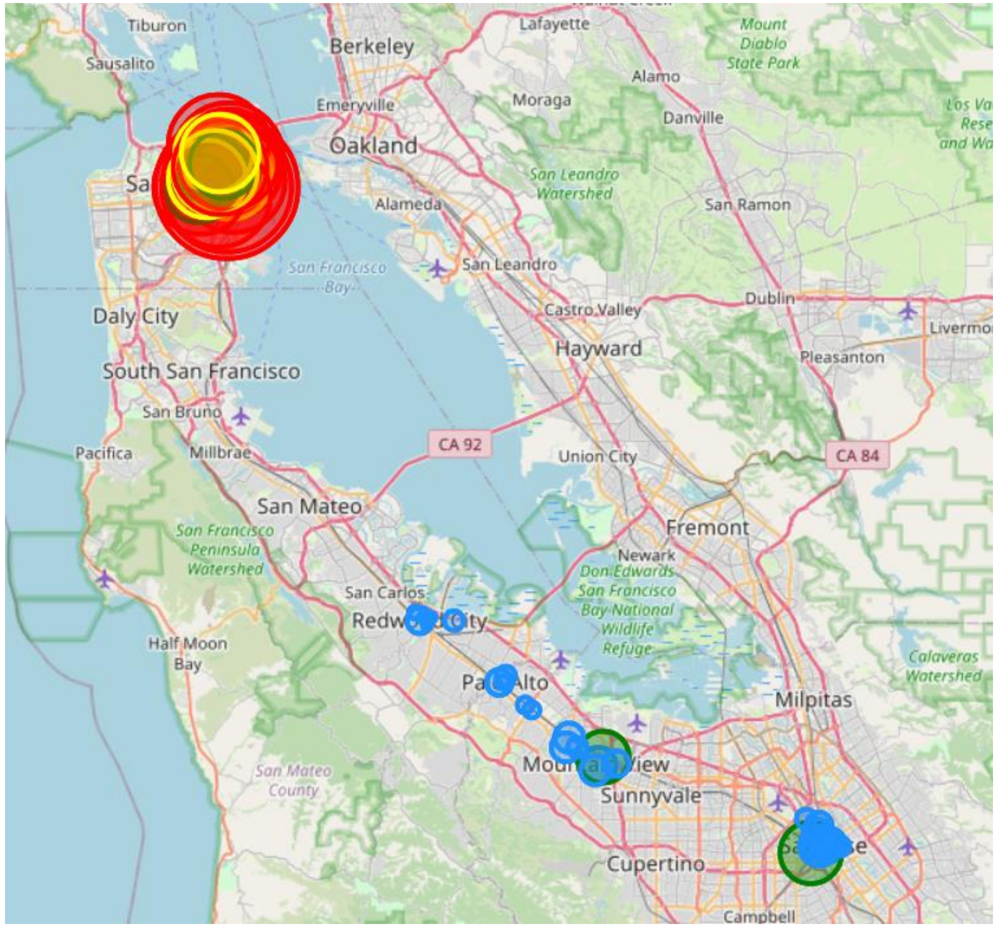
Most popular bike rental station (5/6)

- Draw a circle marker

```
fig = Figure(width=800, height=600)

stataion_starts_map = folium.Map(location = [starts_map_data['lat']
.mean(axis='rows'), starts_map_data['long'].mean(axis='rows')],
              zoom_start = 10)

for markerData in starts_map_data.values:
    folium.CircleMarker([markerData[0],markerData[1]],
     fill = True, color = color_select(markerData[4]),
     radius = (markerData[4]**(1/2))/2).add_to(station_starts_map)

fig.add_child(station_starts_map)
```

Most popular bike rental station (6/6)

- Results



Q&A