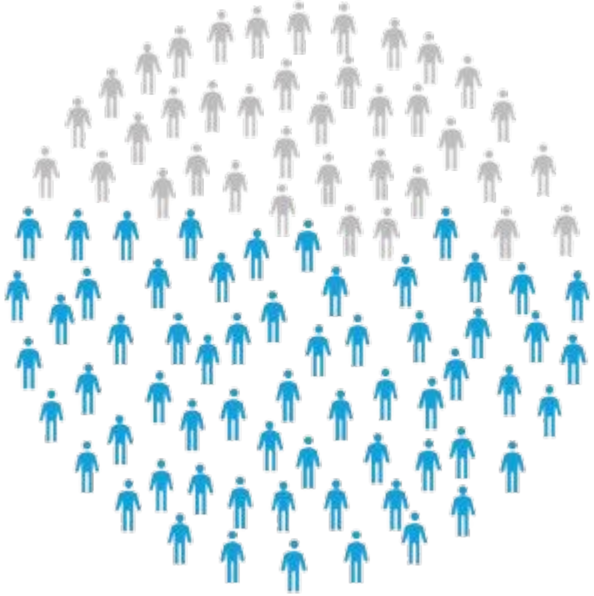
Group 3

Waffle Charts And Word Cloud



Explore waffle charts with the help of an illustration

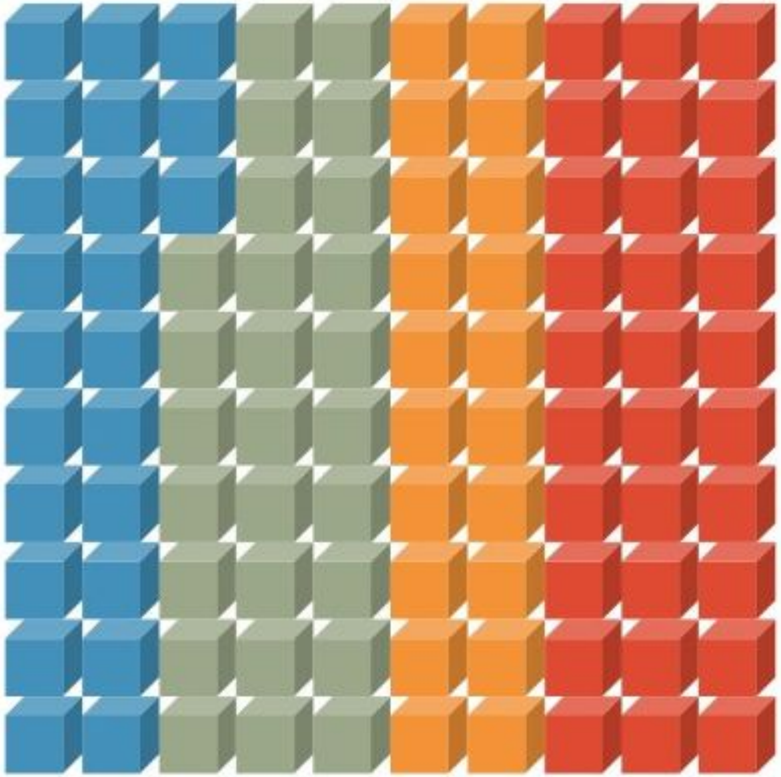
Identify the use cases of waffle charts

Explore word cloud with the help of an illustration

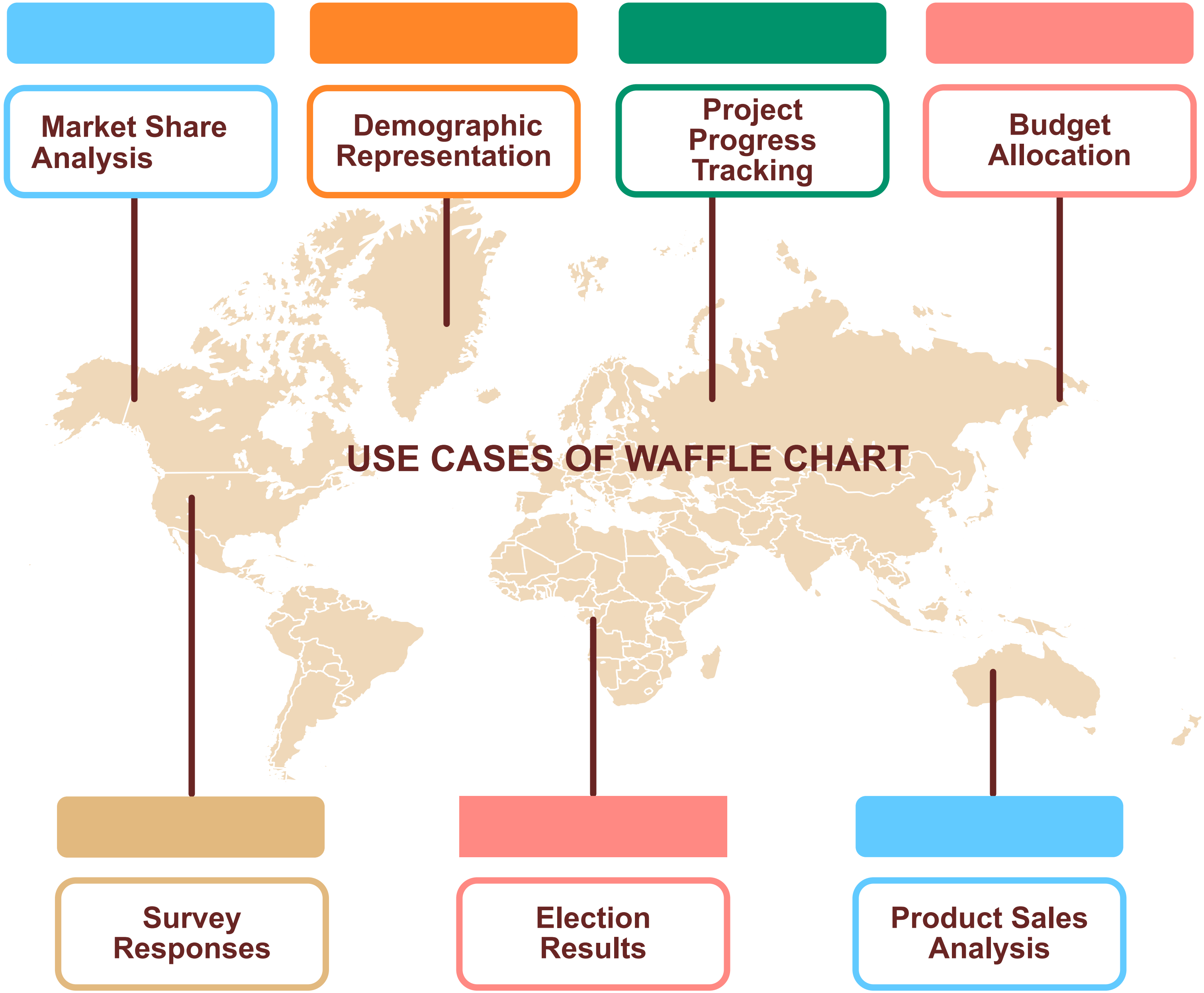
Identify the use cases of Word Cloud

Understanding Waffle Charts

- WHAT ARE WAFFLE CHARTS?
- ADVANTAGES AND LIMITATIONS OF WAFFLE CHART



USE CASES OF WAFFLE CHART



Market Share Analysis

Demographic Representation

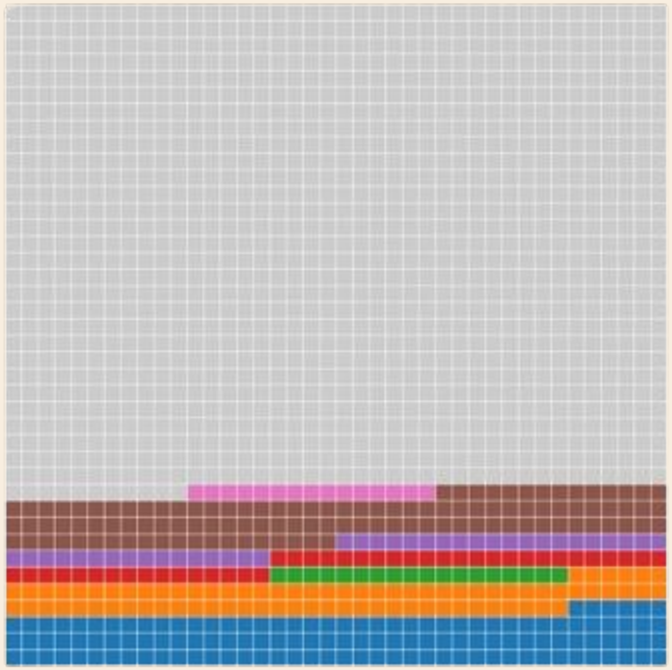
Project Progress Tracking

Budget Allocation

Survey Responses

Election Results

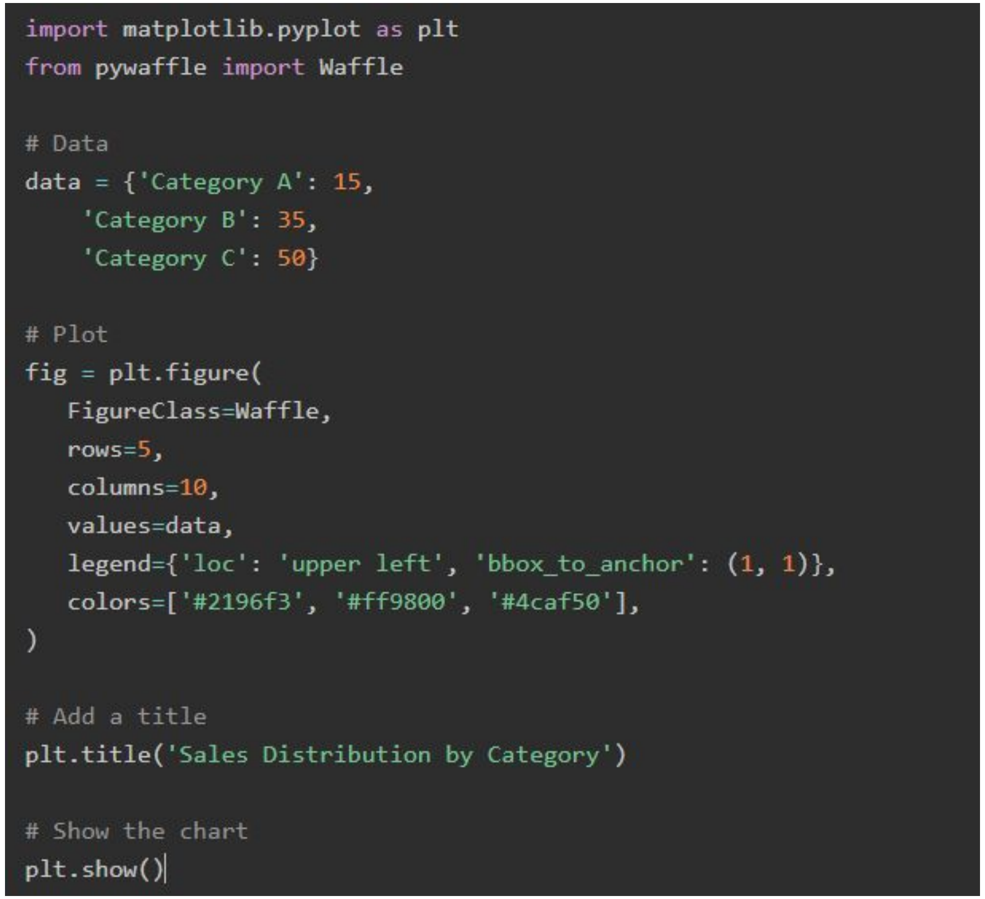
Product Sales Analysis



pywaffle LIBRARY

- WHAT IS pywaffle?
- KEY FEATURES OF PyWaffle
- EXAMPLE CODE
- LIMITATIONS

https://pywaffle.readthedocs.io/en/latest/



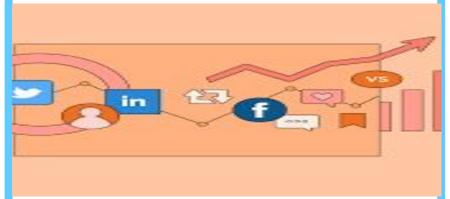
```
import matplotlib.pyplot as plt
from pywaffle import Waffle

# Data
data = {'Category A': 15,
    'Category B': 35,
    'Category C': 50}

# Plot
fig = plt.figure(
   FigureClass=Waffle,
   rows=5,
   columns=10,
   values=data,
   legend={'loc': 'upper left', 'bbox_to_anchor': (1, 1)},
   colors=['#2196f3', '#ff9800', '#4caf50'],
)

# Add a title
plt.title('Sales Distribution by Category')

# Show the chart
plt.show()
```

Understanding Word cloud

- WHAT ARE WORD CLOUD?
- ADVANTAGES AND LIMITATIONS OF WORD CLOUD
- USE CASES OF WORD CLOUD

knowledge
learning
education
development
student
value
wisdom
school
proficiency
comprehension
motivation
faculty
awareness
notion
doctrine
science
semantics
academics
reading
discernment
theory
principles
potential
facts
consciousness
accomplishments
book
sense
brilliance
global
attitude
attention
discipline
observation
enlightenment
graduate
philosophy
acquaintance
light
ken
dope
insight
power
practice
intelligence
substance
apprehension
coaching
vision
dogma
scoop
familiarity
genius
recognition
judgment
cognition
scholarship
study
idea
curriculum
tolerance
university
expertise
career
tuition
degree
attainments
experience
goods
schooling
instruction
perception
concept
erudition
know-how
picture
lore
grasp
ability
teacher

USE CASES OF WORD CLOUD



Social Media Analysis

Customer Feedback Analysis

Content Analysis

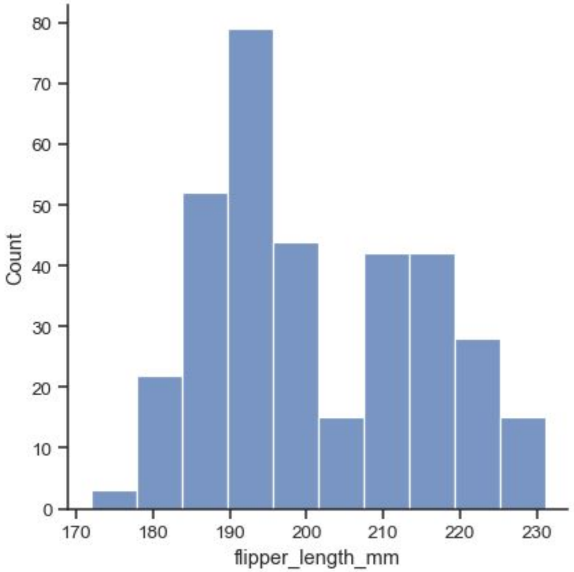
Market Research

Resume or Job Description Analysis

seaborn

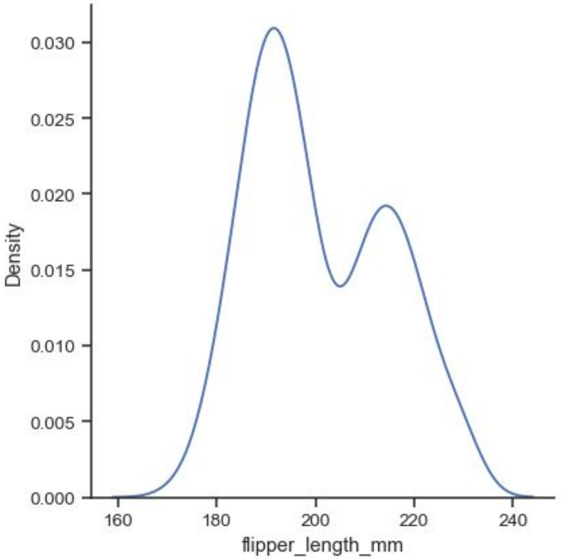
an overview

Histogram



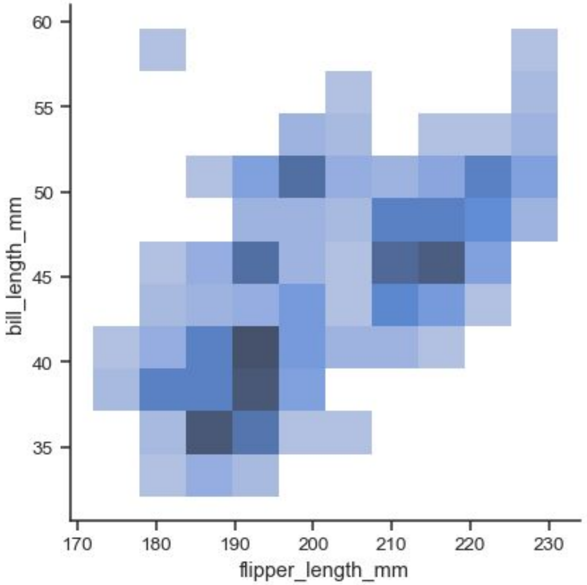
```
sns.displot(
    data=penguins,
    x="flipper_length_mm"
)
```

Kernel Density Estimation



```
sns.displot(
    data=penguins,
    x="flipper_length_mm",
    kind="kde",
)
```

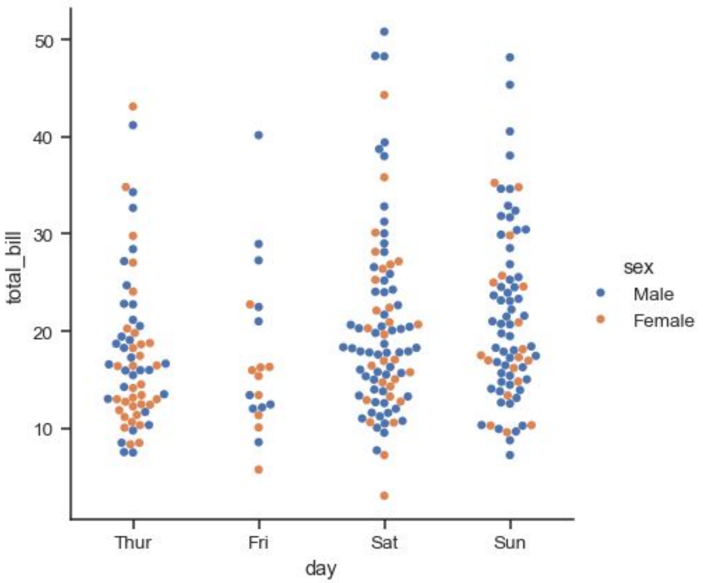
Bivariate



```
sns.displot(
    data=penguins,
    x="flipper_length_mm",
    y="bill_length_mm",
    kind="kde"
)
```

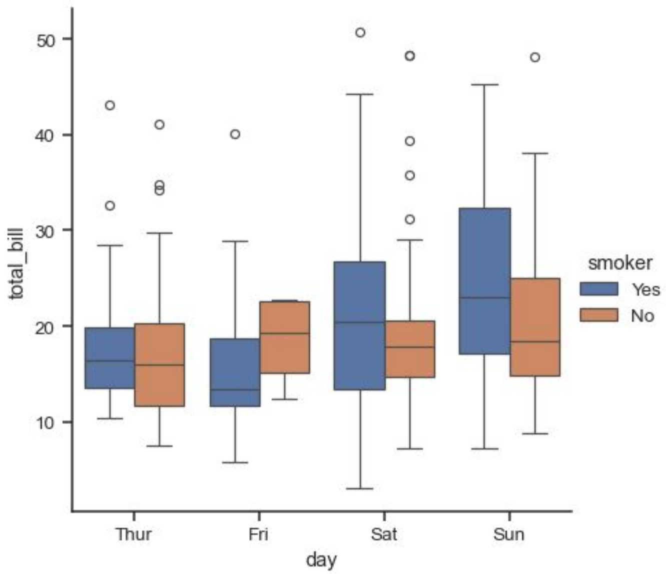
distribution

Categorical Scatter Plot



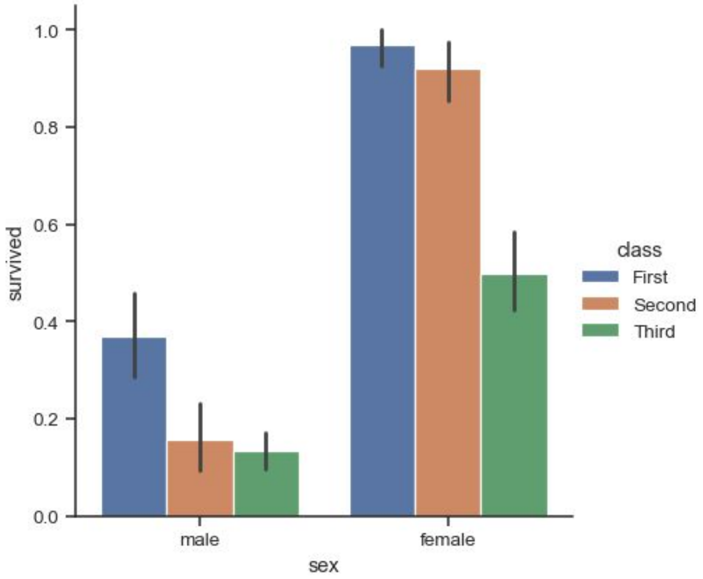
```
sns.catplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="sex",
    kind="swarm"
)
```

Box Plot



```
sns.catplot(
    data=tips,
    x="day",
    y="total_bill",
    hue="smoker",
    kind="box"
)
```

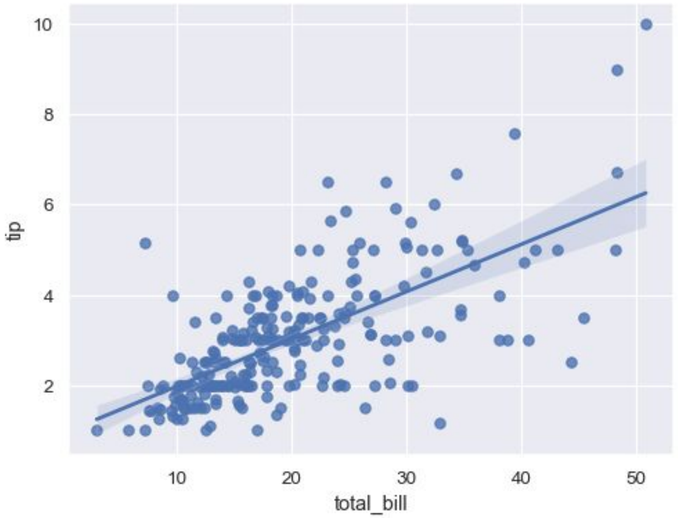
Bar Plot



```
sns.catplot(
    data=titanic,
    x="sex",
    y="survived",
    hue="class",
    kind="bar"
)
```

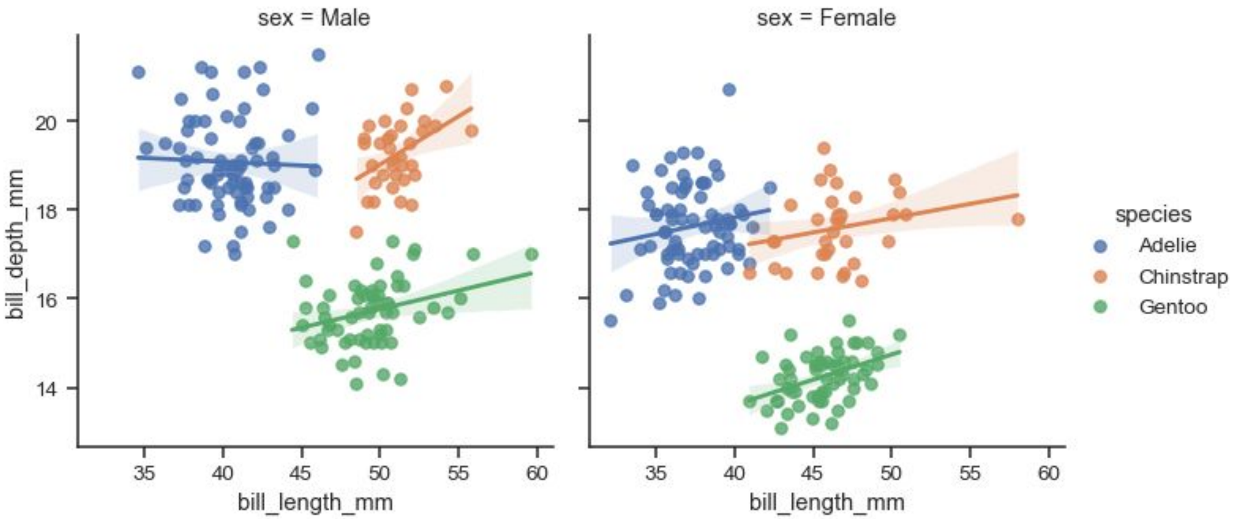
categorical

Linear Regression
via `regplot`



```
sns.regplot(
    data=tips
    x="total_bill",
    y="tip"
);
```

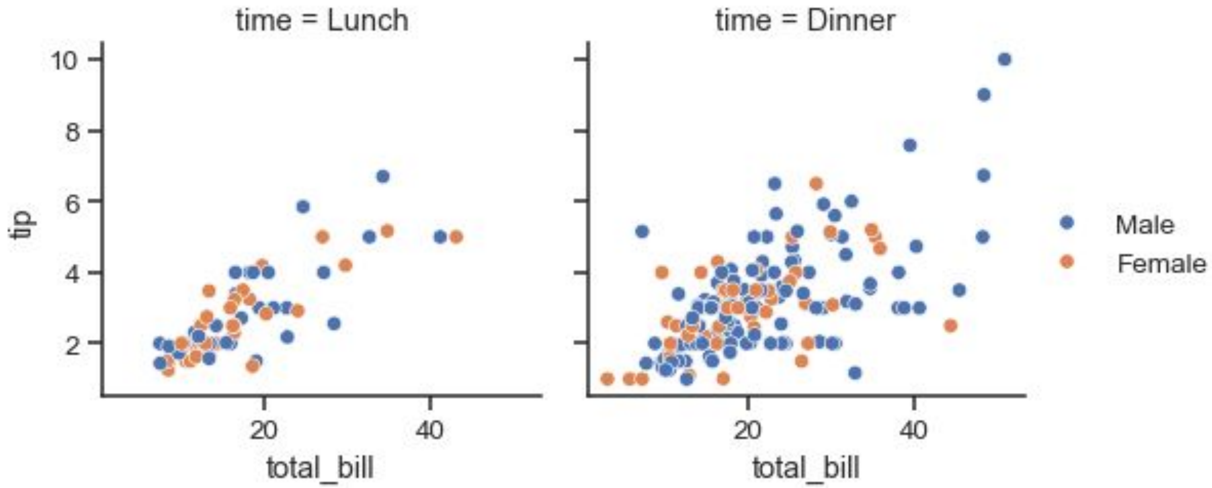
Regression
via `lmplot`



```
sns.lmplot(
    data=penguins,
    x="bill_length_mm",
    y="bill_depth_mm",
    hue="species",
    col="sex",
    height=4,
)
```

regression

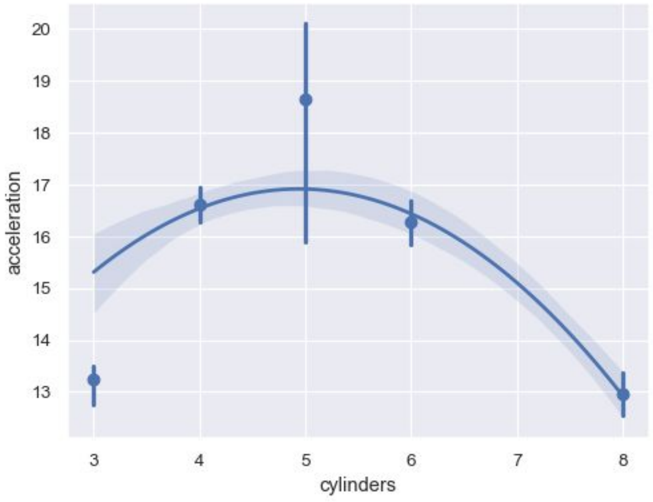
FacetGrid



```
g = sns.FacetGrid(tips, col="time")
g.map_dataframe(
    sns.scatterplot,
    x="total_bill",
    y="tip",
    hue="sex"
)
g.add_legend()
```

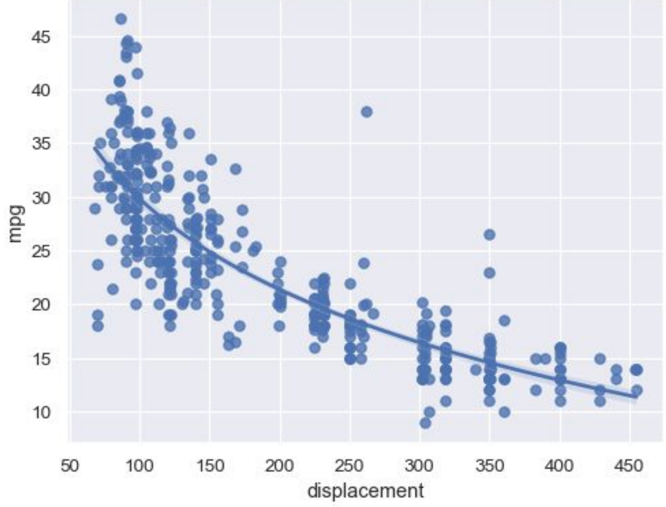
The aforementioned `lmplot` method in the “regression” section is simply a wrapper for a quick way to render a FacetGrid.

Polynomial (Binomial)



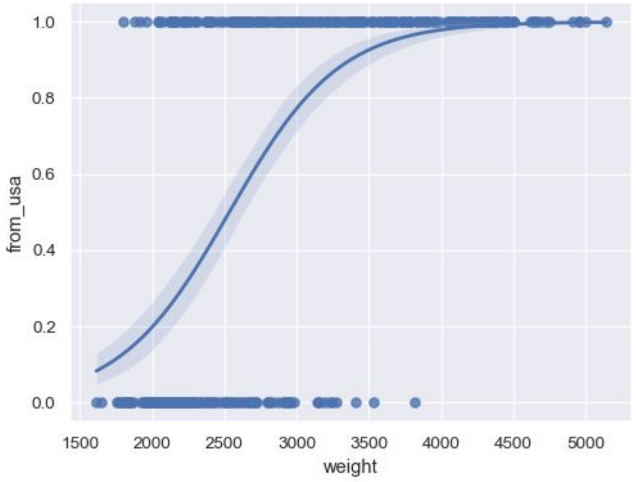
```
sns.regplot(
    data=mpg,
    x="cylinders",
    y="acceleration",
    x_estimator=np.mean,
    order=2
)
```

Logarithmic



```
sns.regplot(
    data=mpg,
    x="displacement",
    y="mpg",
    logx=True
)
```

Logistic



```
sns.regplot(
    x=mpg["weight"],
    y=mpg["origin"]
        .eq("usa")
        .rename("from_usa"),
    logistic=True
)
```

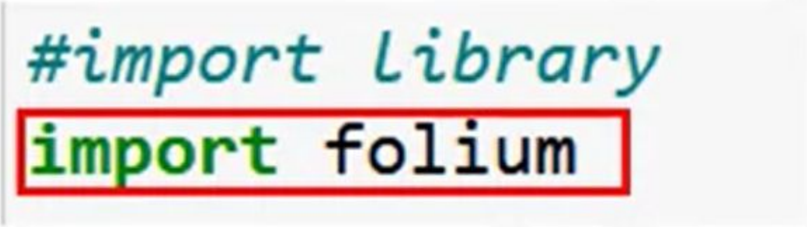
nonlinear regression

Folium

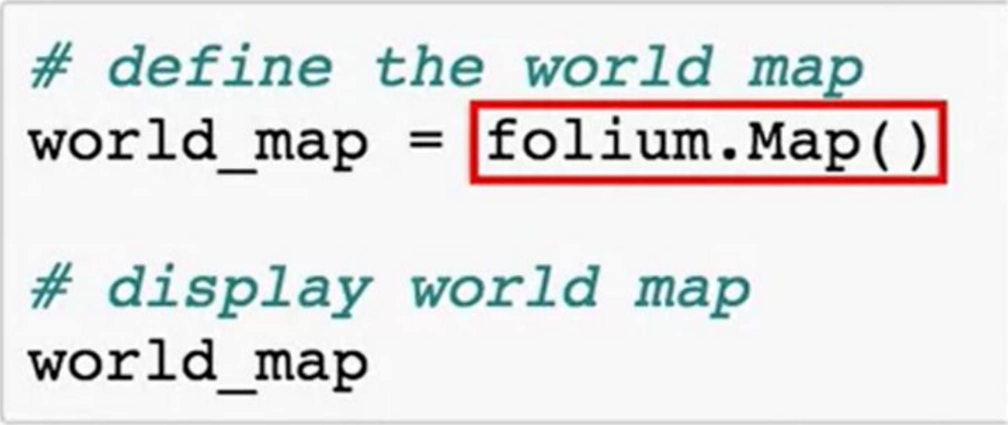
Folium Introduction & Use Cases

- Powerful Data Visualization Library in Python
- Primarily used to help people visualize geospatial data
- Create a map for any location in the world using Folium
- Parameters : Longitude and Latitude values
- Superimpose Markers and clusters for interesting Visualizations
- Street-Level & Stamen Maps can also be created

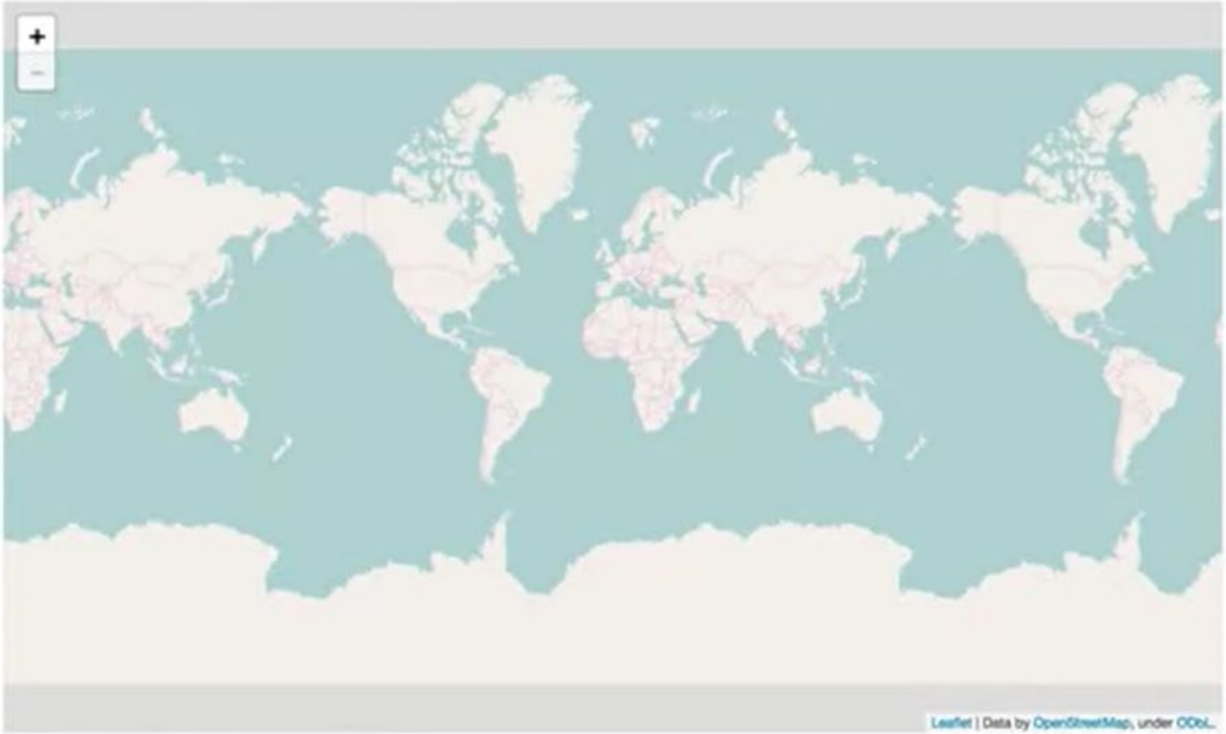
Creating World Map



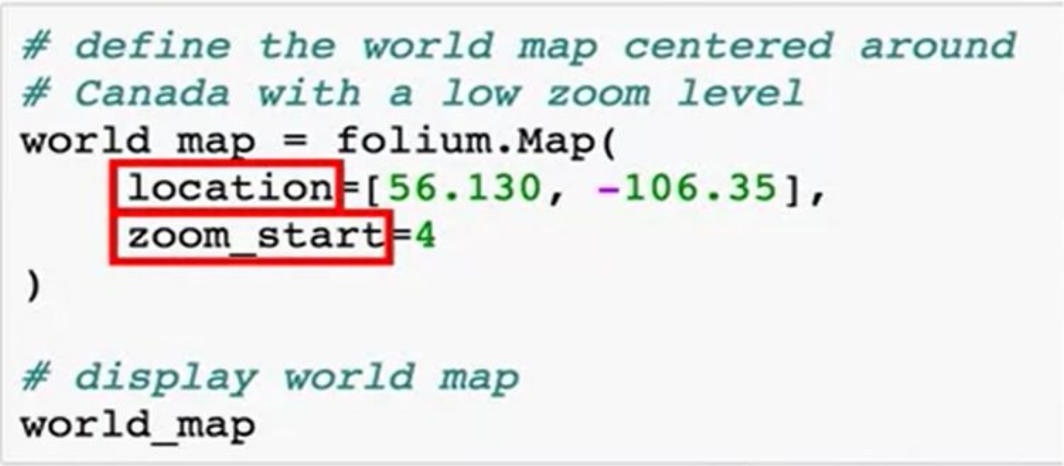
```
#import library
import folium
```



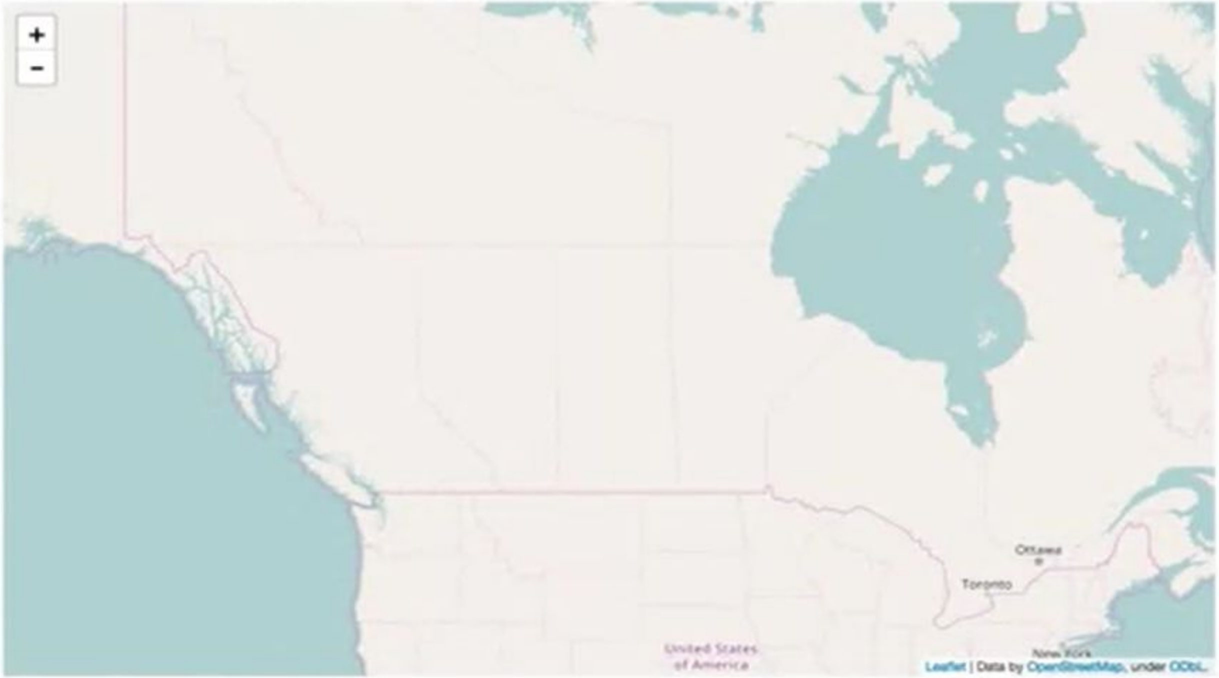
```
# define the world map
world_map = folium.Map()

# display world map
world_map
```



Creating a map of Canada

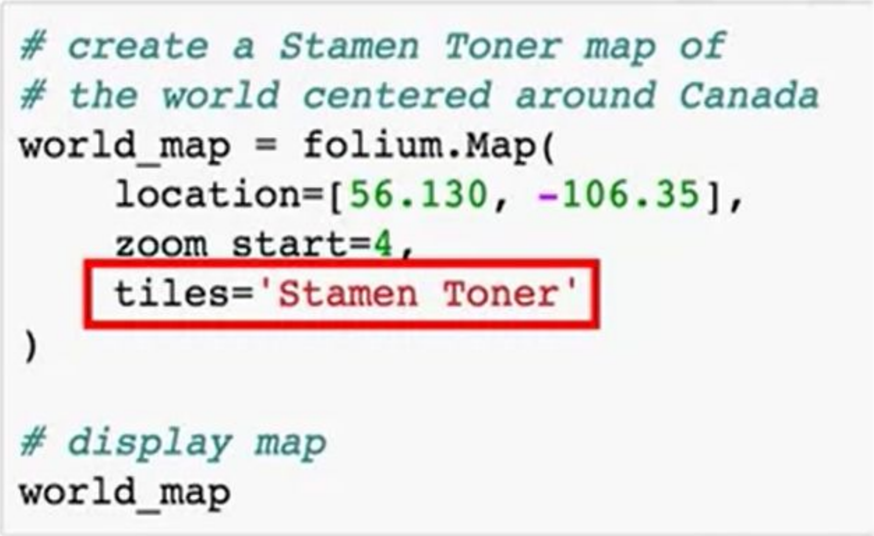


```
# define the world map centered around
# Canada with a low zoom level
world_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)

# display world map
world_map
```

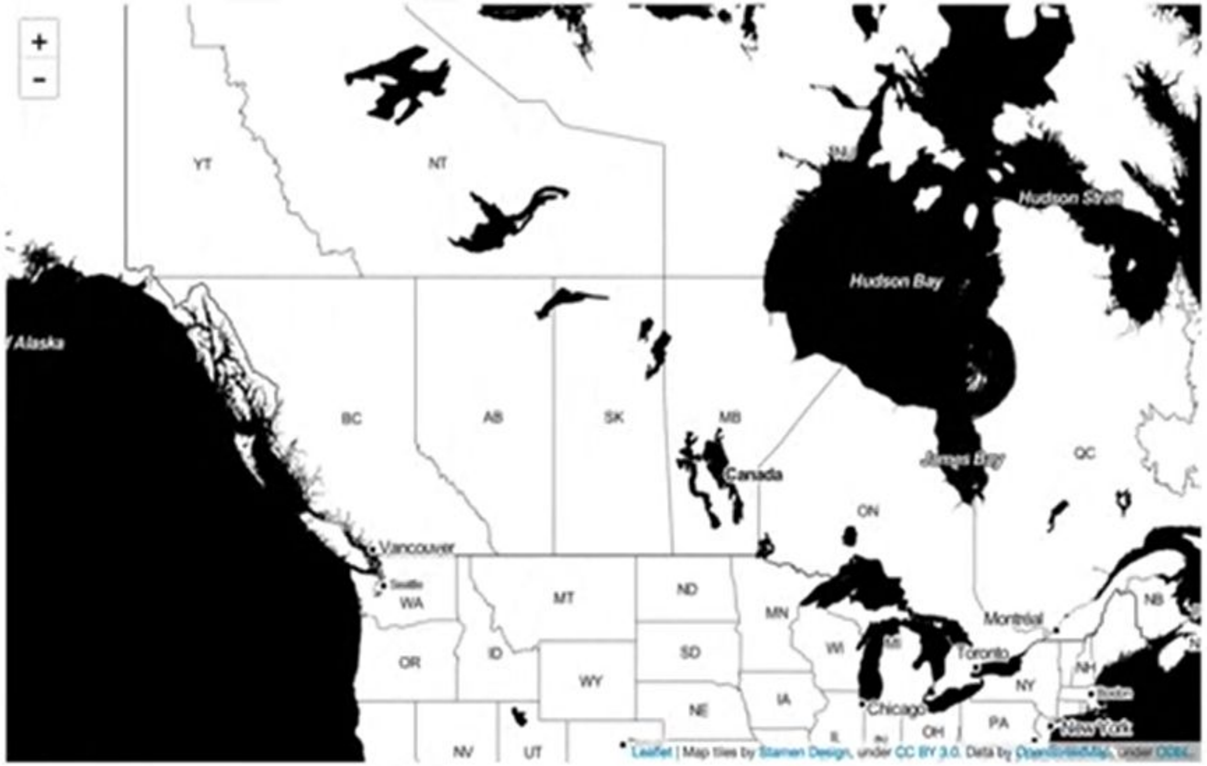


Map Style - ‘Stamen Tomer’

This style is great for visualizing and exploring rivers and coastal zones.

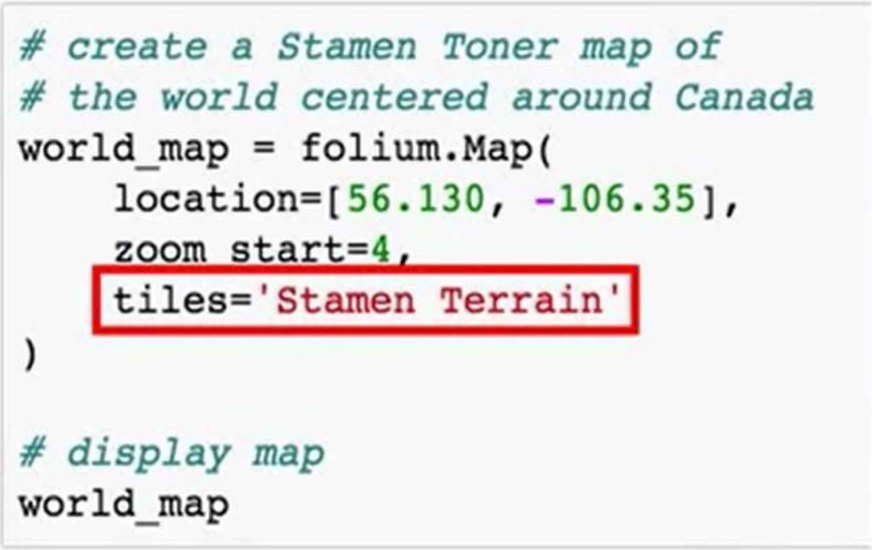


```
# create a Stamen Toner map of
# the world centered around Canada
world_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4,
    tiles='Stamen Toner'
)

# display map
world_map
```

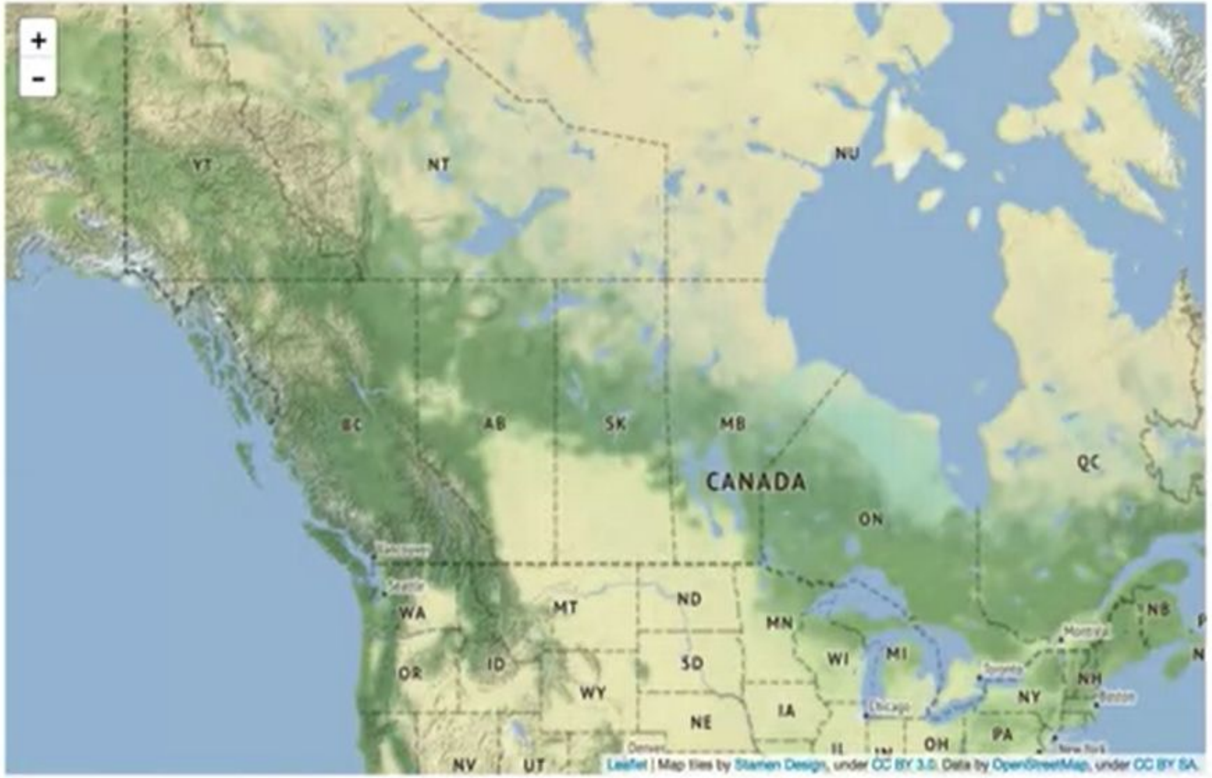


Map Style - ‘Stamen Terrain’

This style is excellent for visualizing hill shading and natural vegetation colors.



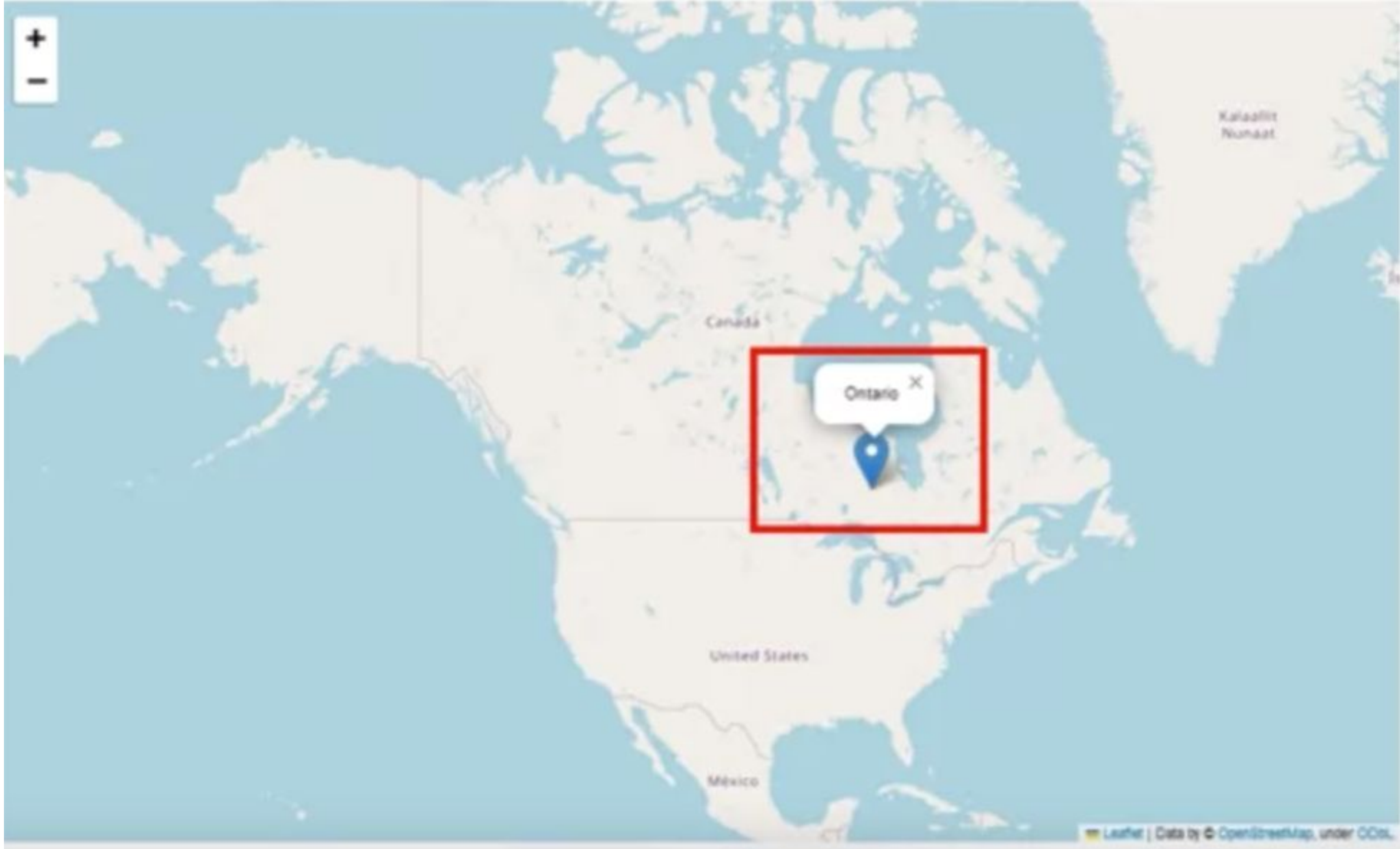
```
# create a Stamen Toner map of
# the world centered around Canada
world_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4,
    tiles='Stamen Terrain'
)

# display map
world_map
```



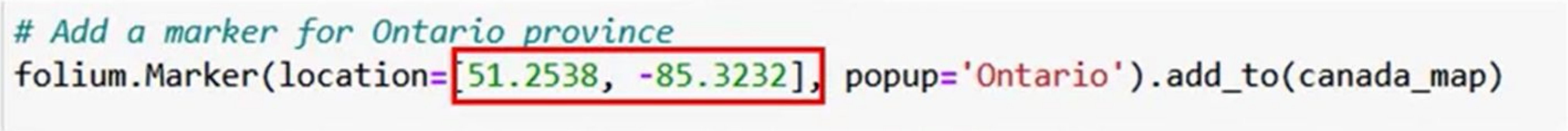
Leaflet | Map tiles by Stamen Design, under CC BY 3.0. Data by OpenStreetMap, under CC BY SA.

Adding Markers & Labels to the Map

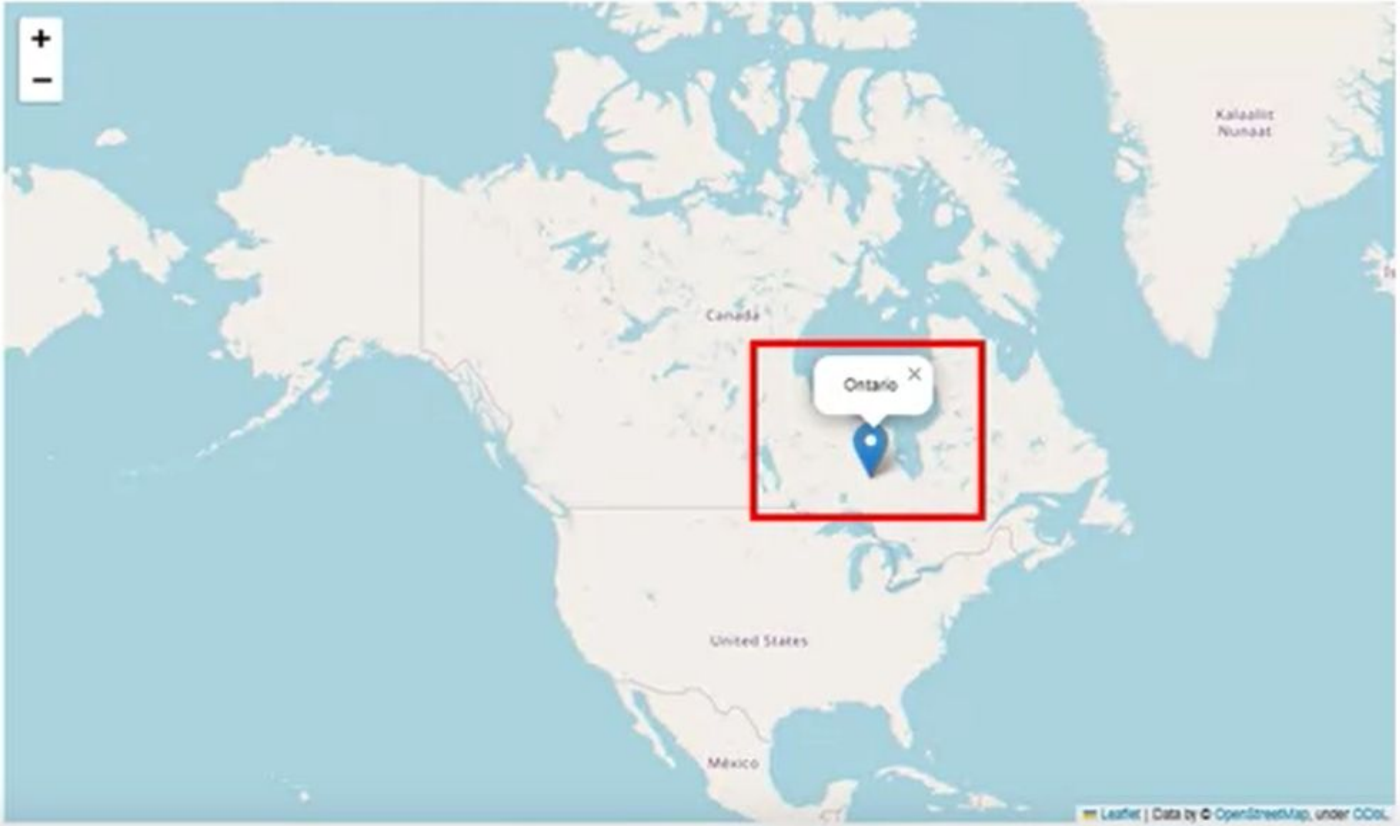
Markers play a vital role in enhancing interactivity and adding context to maps. They represent specific locations or points of interest, providing additional information when clicked.



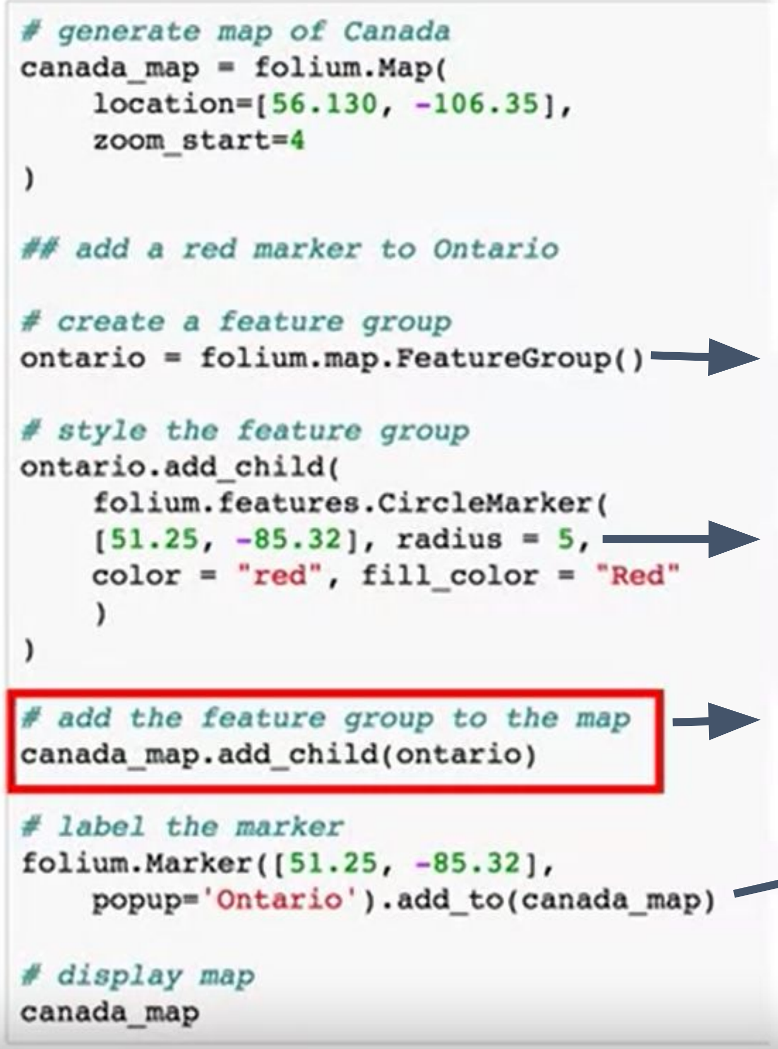
Adding Markers & Labels to the Map



```python
# Add a marker for Ontario province
folium.Marker(location=[51.2538, -85.3232], popup='Ontario').add_to(canada_map)
```



Adding Markers with Feature Group

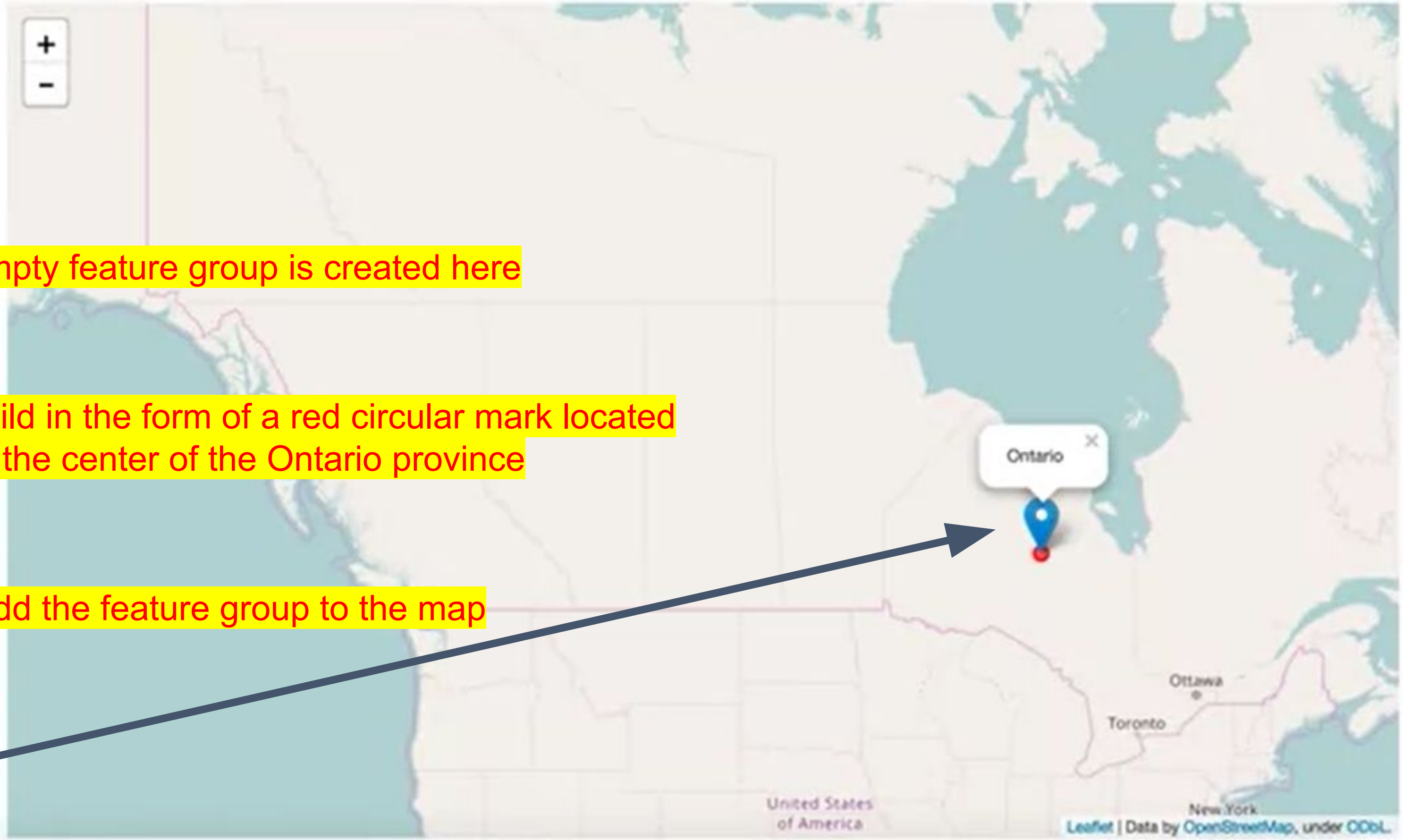


```
# generate map of Canada
canada_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)

## add a red marker to Ontario

# create a feature group
ontario = folium.map.FeatureGroup()

# style the feature group
ontario.add_child(
    folium.features.CircleMarker(
    [51.25, -85.32], radius = 5,
    color = "red", fill_color = "Red"
    )
)

# add the feature group to the map
canada_map.add_child(ontario)

# label the marker
folium.Marker([51.25, -85.32],
    popup='Ontario').add_to(canada_map)

# display map
canada_map
```

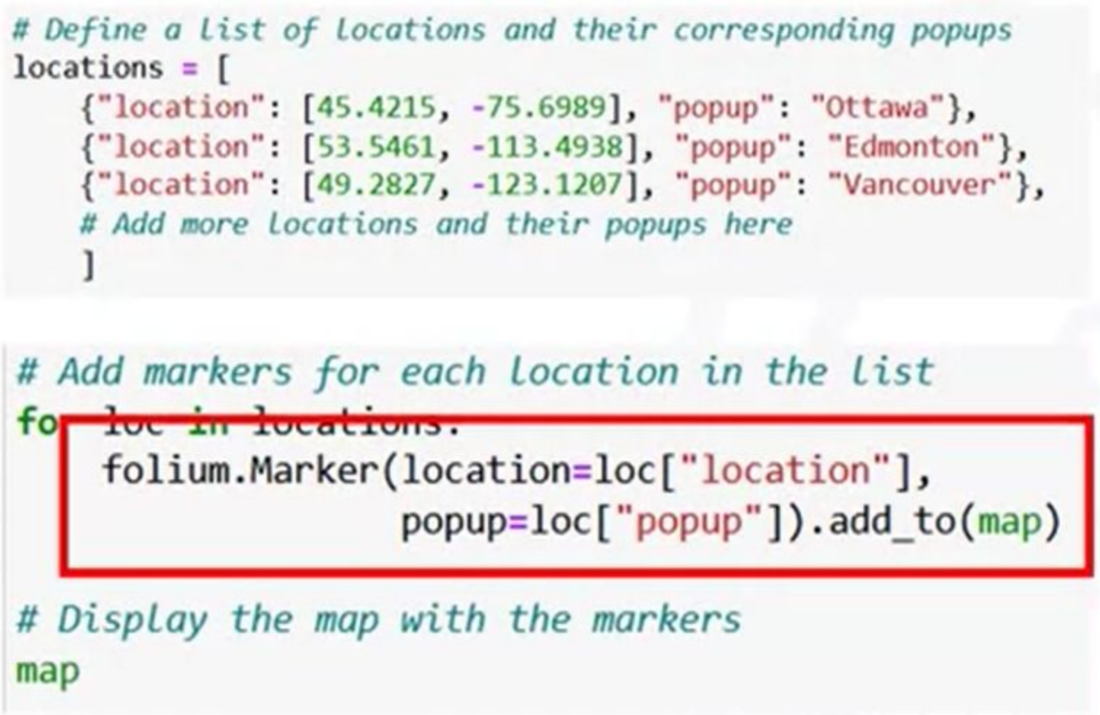
empty feature group is created here

child in the form of a red circular mark located at the center of the Ontario province

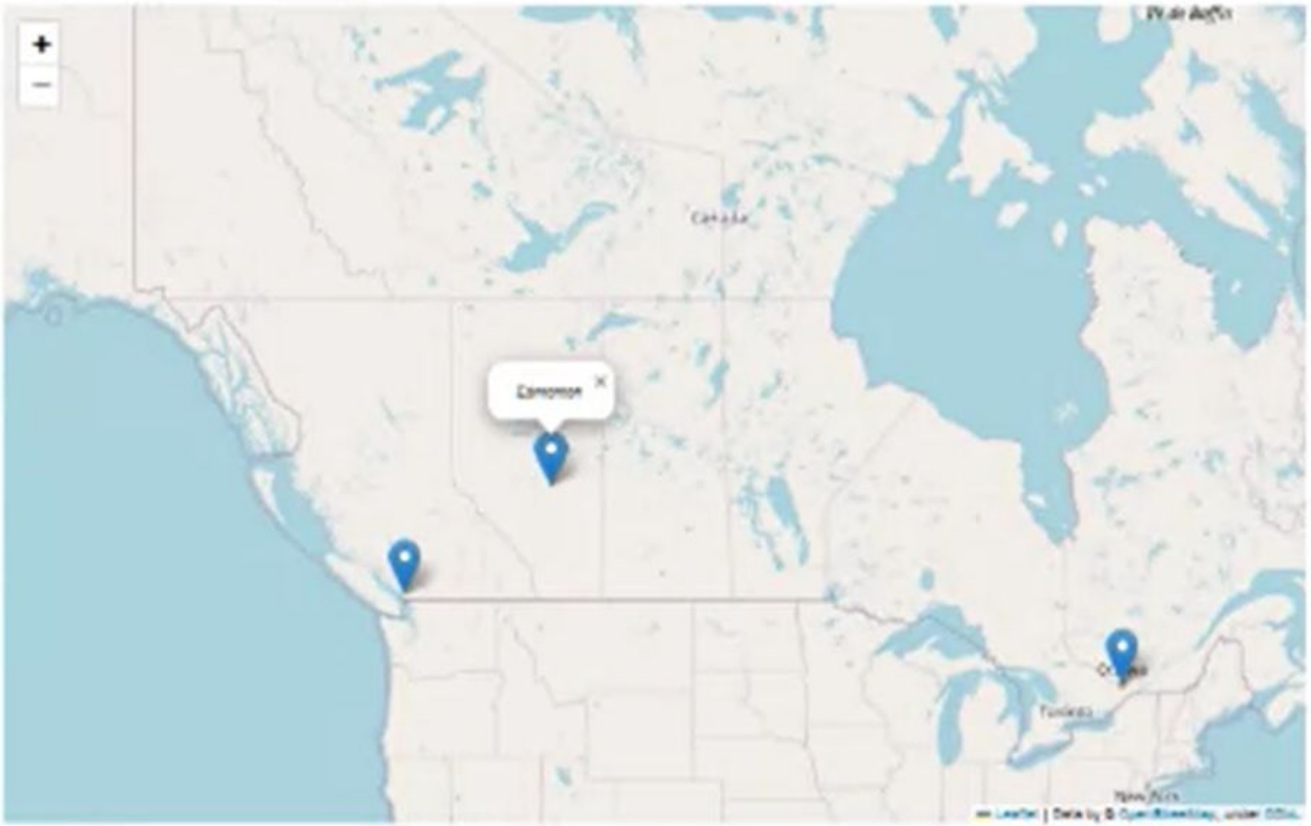
add the feature group to the map



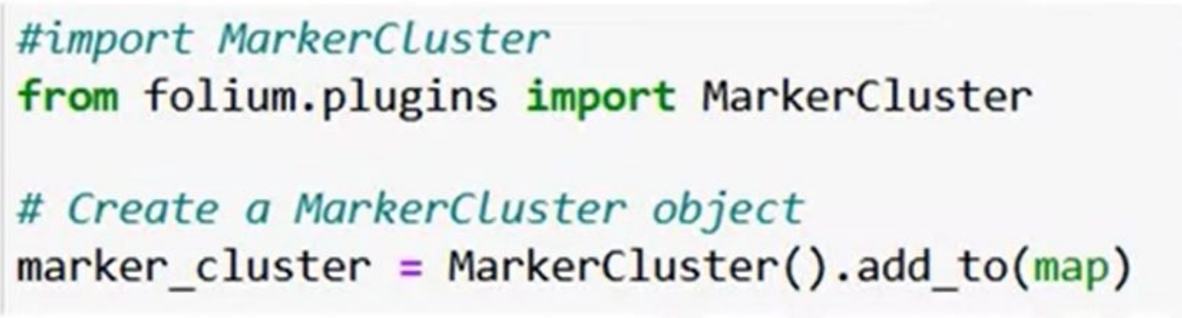
Adding Multiple Markers to the Map



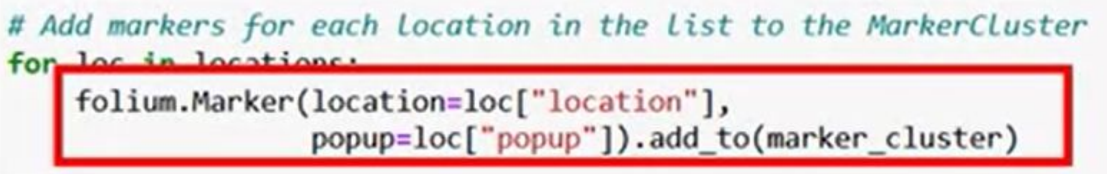
```python
# Define a list of locations and their corresponding popups
locations = [
    {"location": [45.4215, -75.6989], "popup": "Ottawa"},
    {"location": [53.5461, -113.4938], "popup": "Edmonton"},
    {"location": [49.2827, -123.1207], "popup": "Vancouver"},
    # Add more locations and their popups here
    ]

# Add markers for each location in the list
for loc in locations:
    folium.Marker(location=loc["location"],
                  popup=loc["popup"]).add_to(map)

# Display the map with the markers
map
```



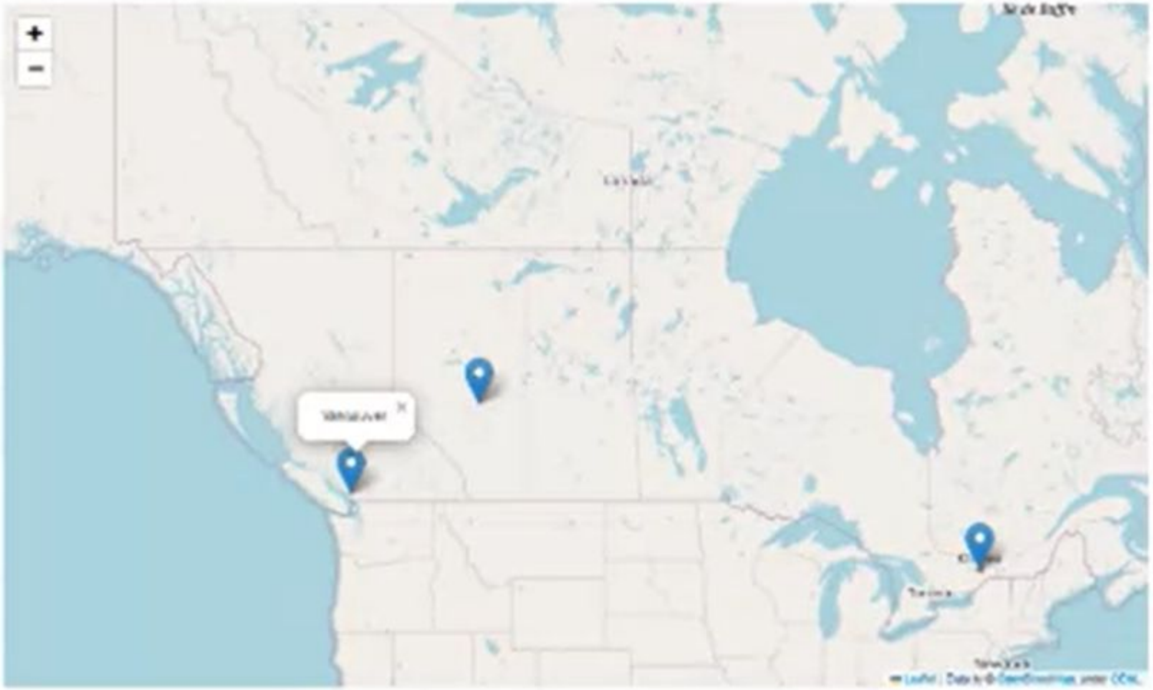
Clustering Feature



```
#import MarkerCluster
from folium.plugins import MarkerCluster

# Create a MarkerCluster object
marker_cluster = MarkerCluster().add_to(map)
```

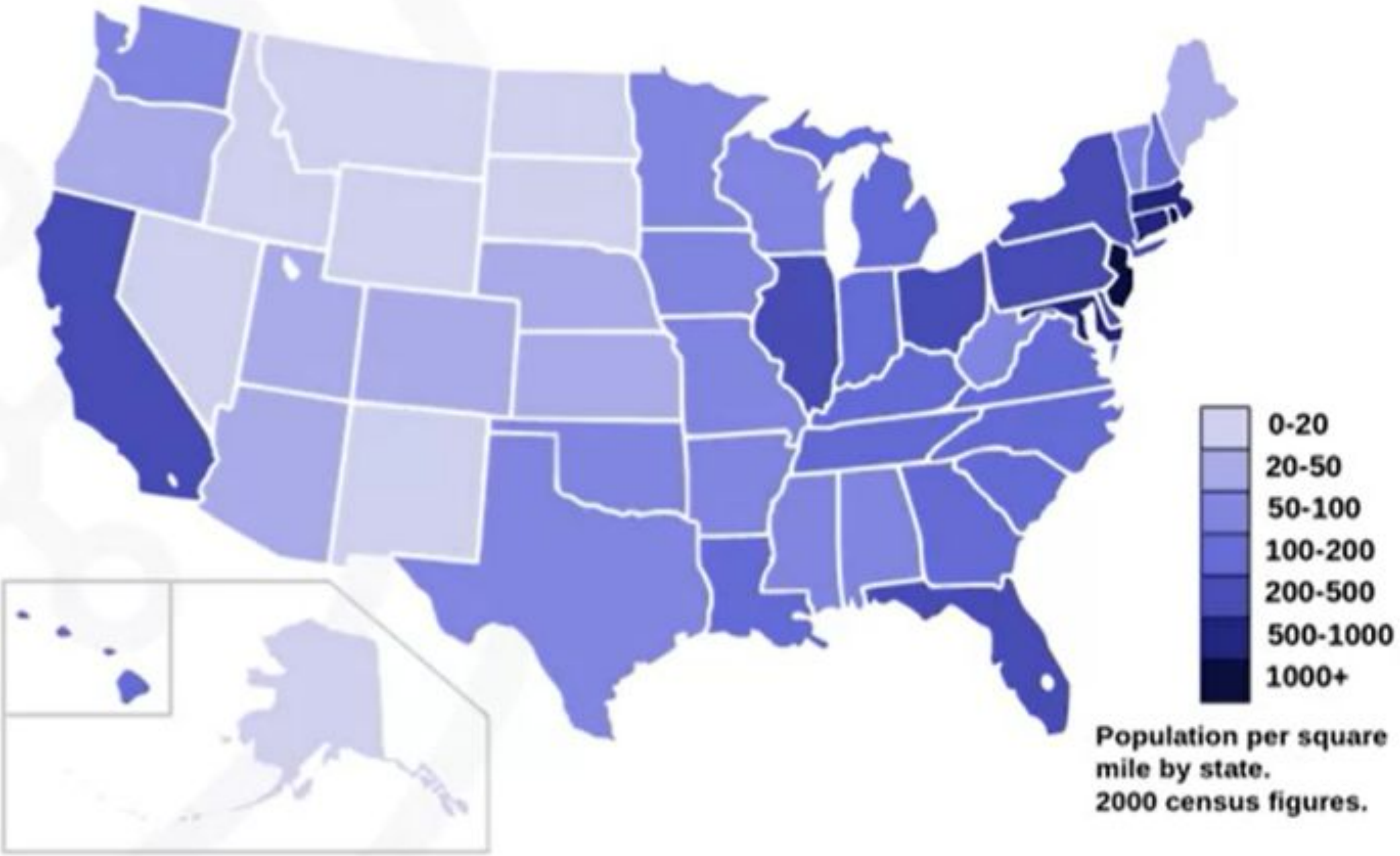


```
# Add markers for each location in the list to the MarkerCluster
for loc in locations:
    folium.Marker(location=loc["location"],
                  popup=loc["popup"]).add_to(marker_cluster)
```

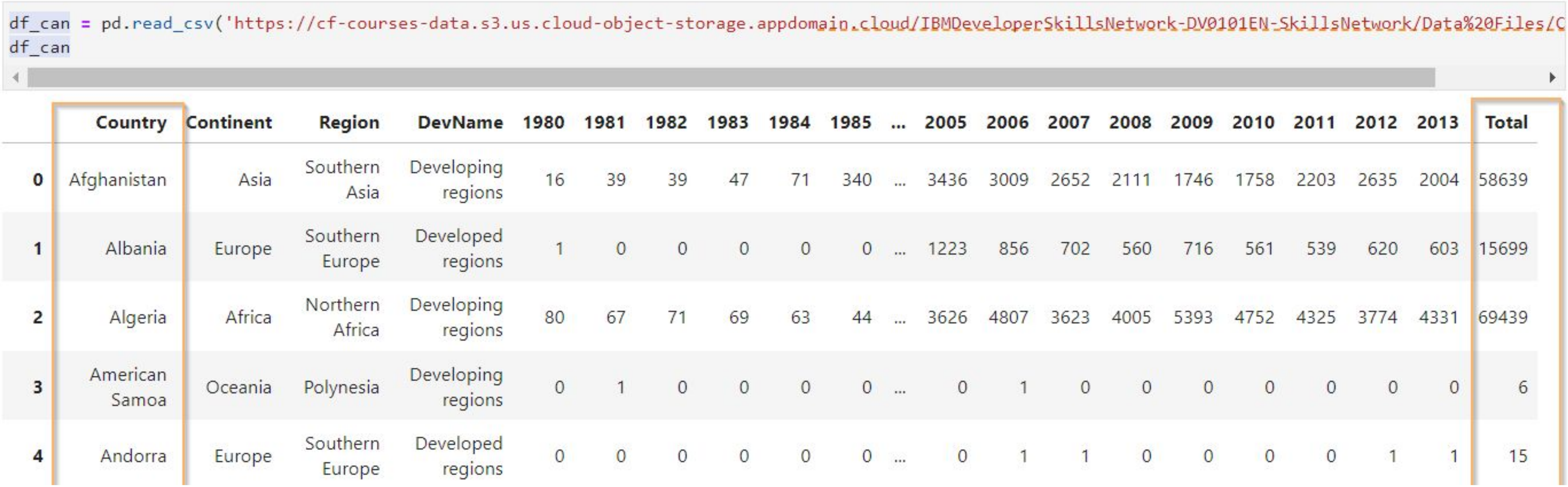


This clustering feature enhances the visual presentation by preventing overcrowding and ensuring a clear representation, primarily when numerous markers are close.

Choropleth maps



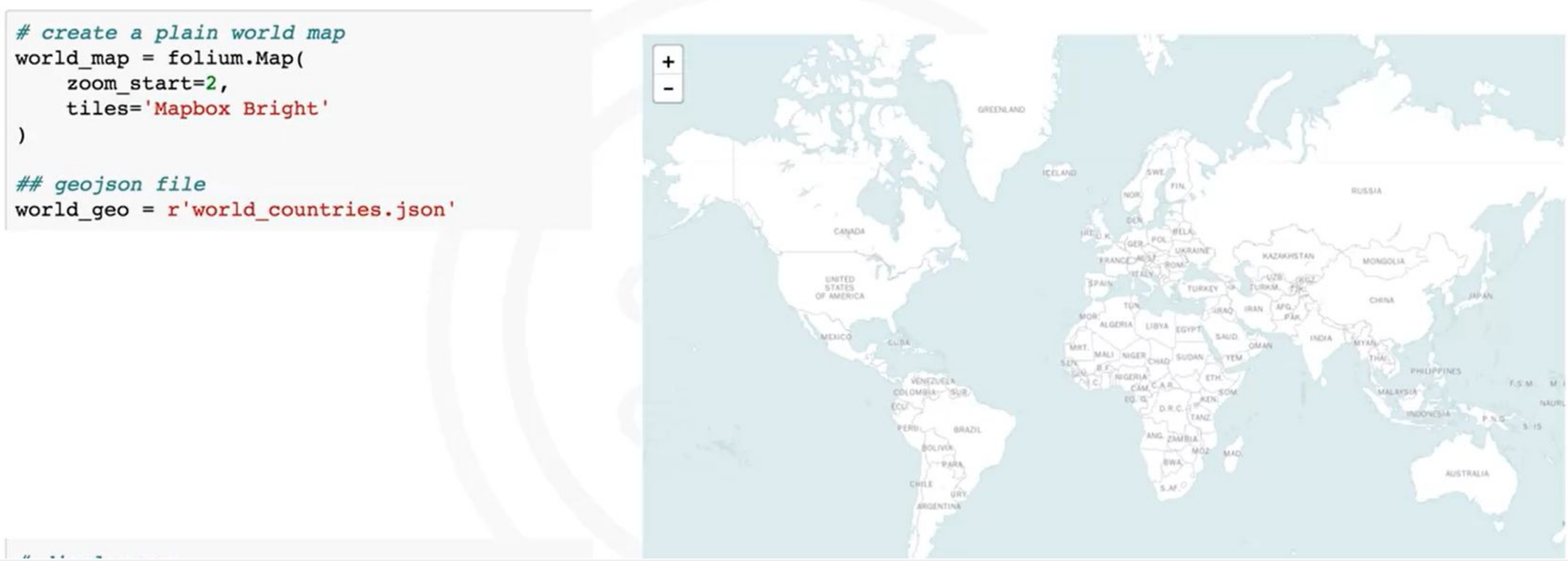
Choropleth maps



```
df_can = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/C
df_can
```

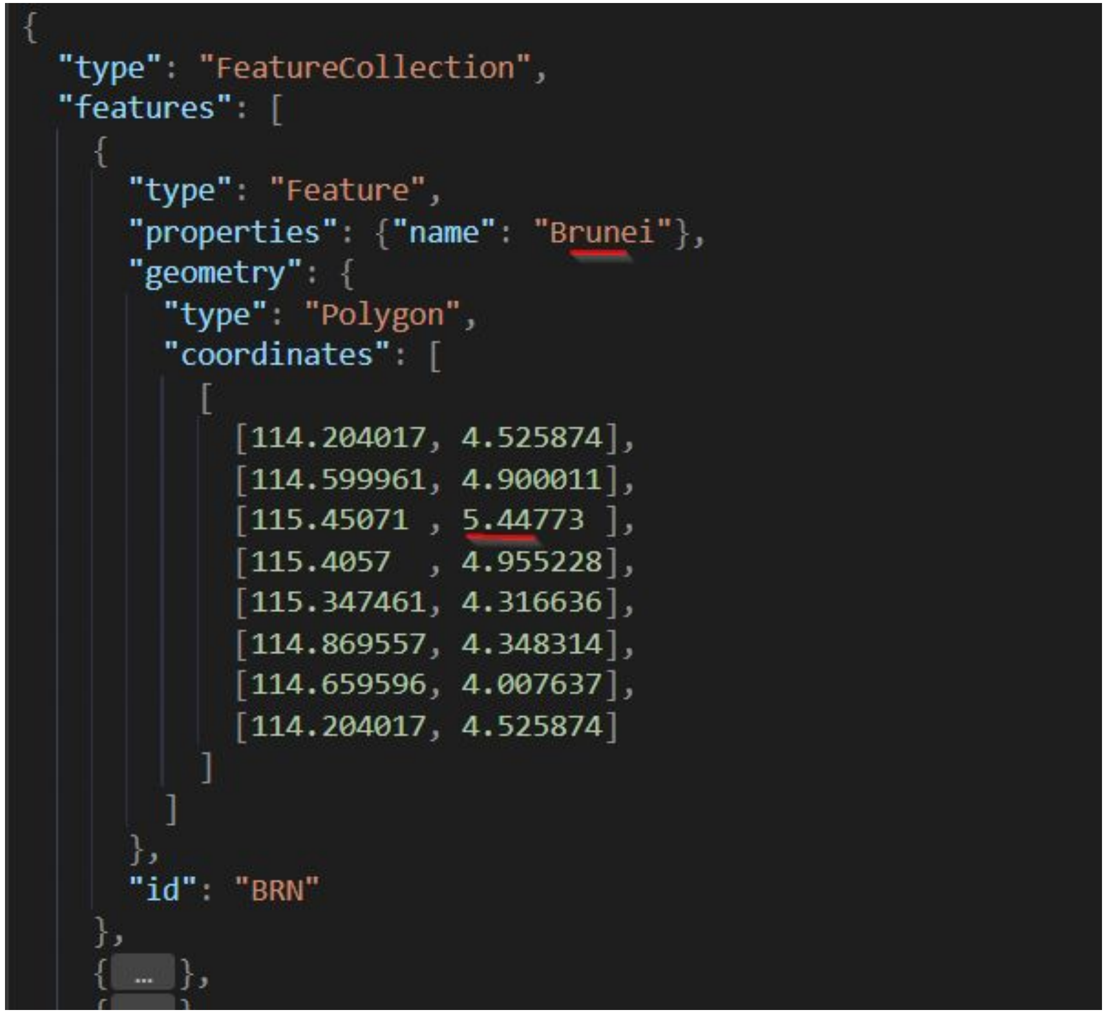
	Country	Continent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
0	Afghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	3436	3009	2652	2111	1746	1758	2203	2635	2004	58639
1	Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1223	856	702	560	716	561	539	620	603	15699
2	Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3626	4807	3623	4005	5393	4752	4325	3774	4331	69439
3	American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	1	0	0	0	0	0	0	0	6
4	Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	0	1	1	0	0	0	0	1	1	15

Choropleth maps

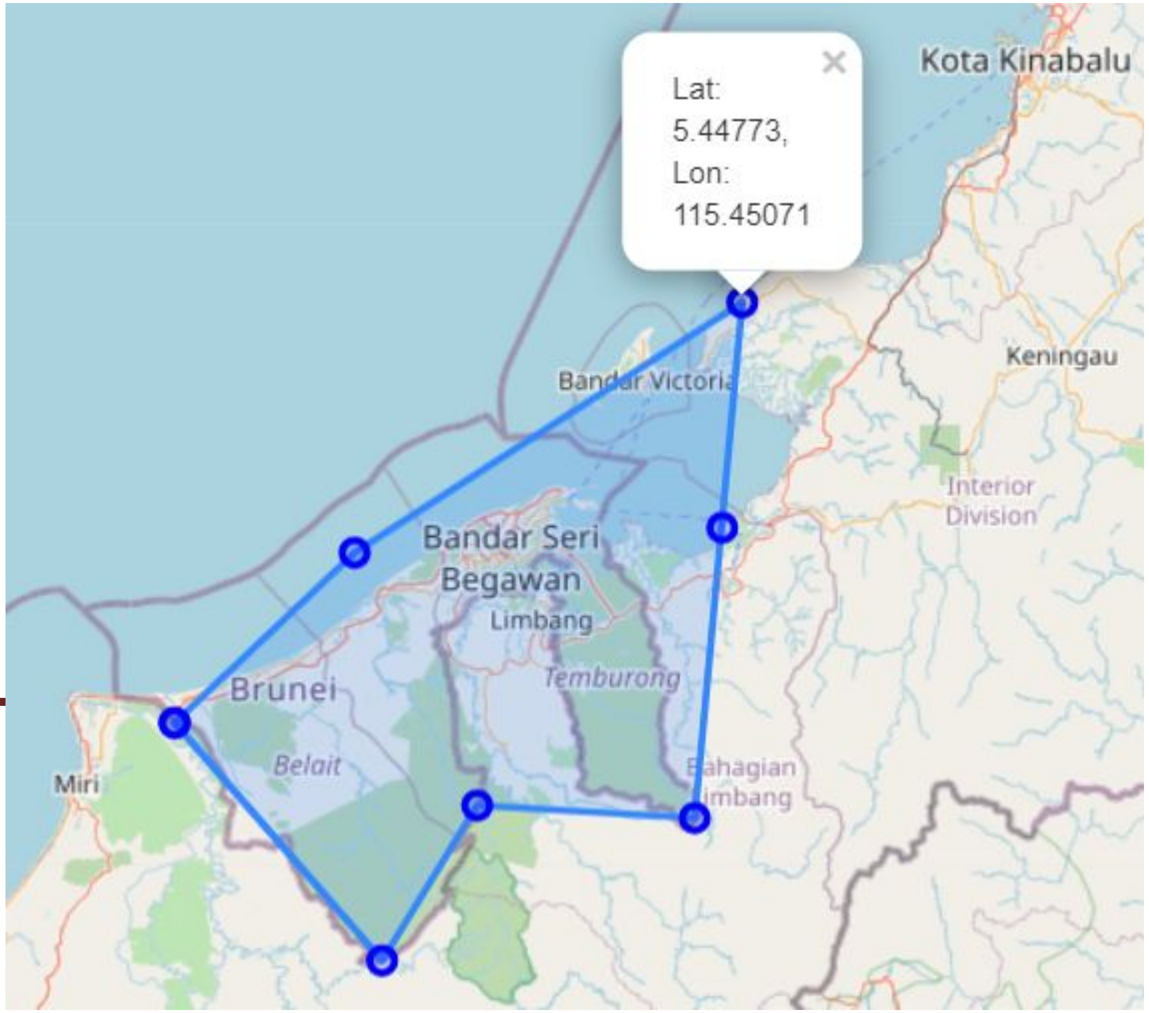


```
# create a plain world map
world_map = folium.Map(
    zoom_start=2,
    tiles='Mapbox Bright'
)

## geojson file
world_geo = r'world_countries.json'
```

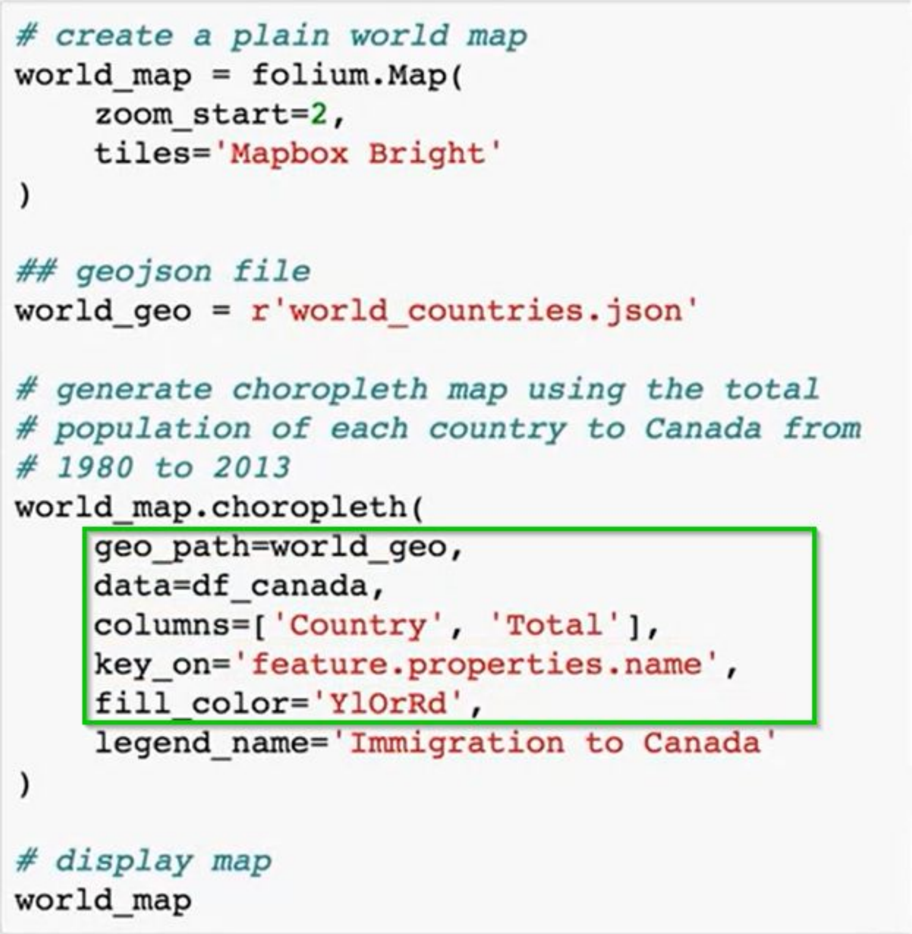
GeoJSON



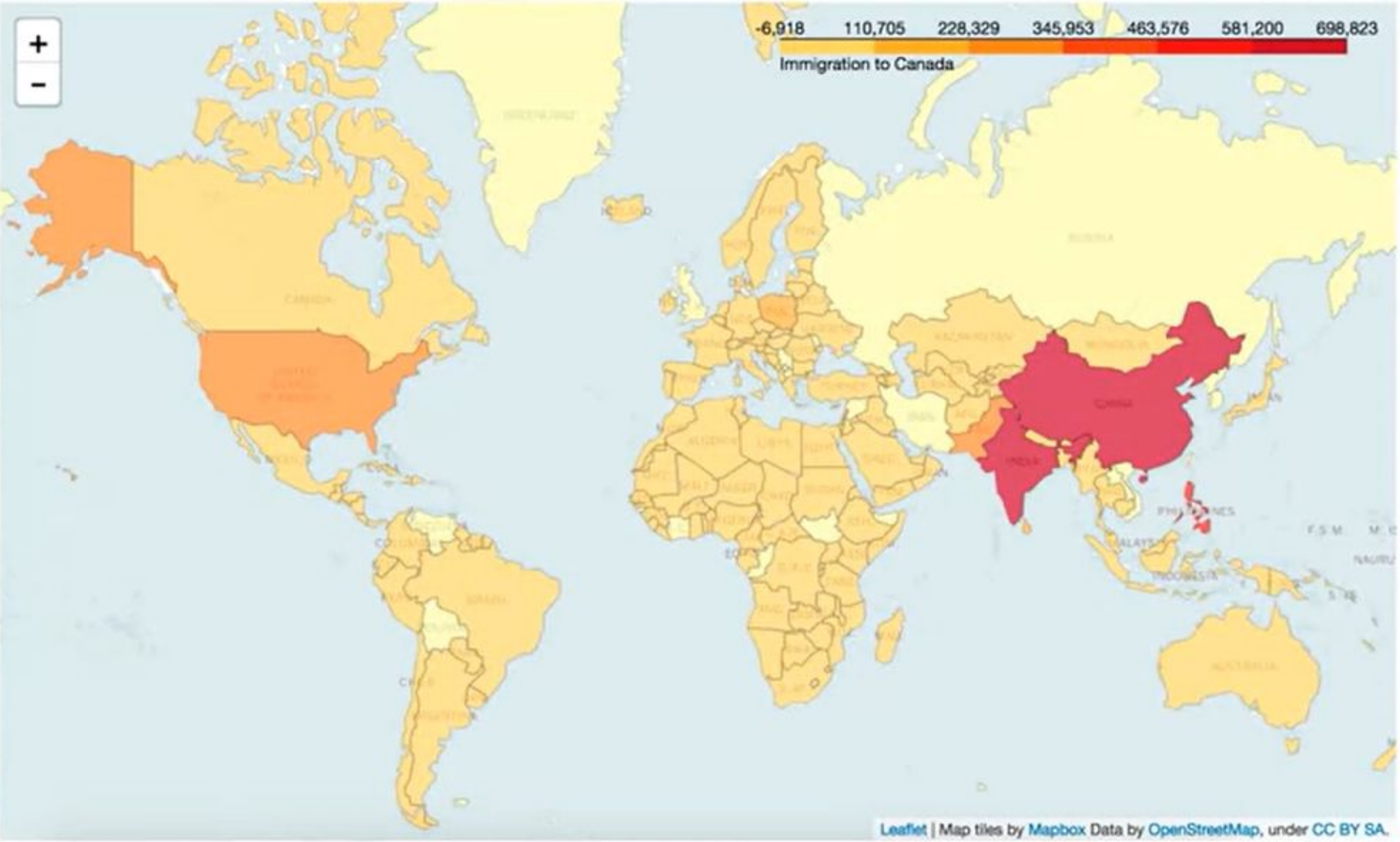
```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {"name": "Brunei"},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [114.204017, 4.525874],
            [114.599961, 4.900011],
            [115.45071 , 5.44773 ],
            [115.4057  , 4.955228],
            [115.347461, 4.316636],
            [114.869557, 4.348314],
            [114.659596, 4.007637],
            [114.204017, 4.525874]
          ]
        ]
      },
      "id": "BRN"
    },
    {…},
```



Choropleth maps



```
# create a plain world map
world_map = folium.Map(
    zoom_start=2,
    tiles='Mapbox Bright'
)

## geojson file
world_geo = r'world_countries.json'

# generate choropleth map using the total
# population of each country to Canada from
# 1980 to 2013
world_map.choropleth(
    geo_path=world_geo,
    data=df_canada,
    columns=['Country', 'Total'],
    key_on='feature.properties.name',
    fill_color='YlOrRd',
    legend_name='Immigration to Canada'
)

# display map
world_map
```



Leaflet | Map tiles by Mapbox Data by OpenStreetMap, under CC BY SA.

Thank You