**Advanced Visualizations and Geospatial Data**

Advanced visualization tools are sophisticated platforms that provide a wide range of advanced features and capabilities. These tools provide an extensive set of options that help create visually appealing and interactive visualizations. In this module, you will learn about waffle charts and word cloud including their application. You will explore Seaborn, a new visualization library in Python, and learn how to create regression plots using it. In addition, you will learn about folium, a data visualization library that visualizes geospatial data. Furthermore, you will explore the process of creating maps using Folium and superimposing them with markers to make them interesting. Finally, you will learn how to create a Choropleth map using Folium.

**Learning Objectives**

* Describe Choropleth maps with the help of an illustration
* Explore the process of superimposing markers on maps using Foilum
* Describe Folium and explore the process of creating maps
* Describe Seaborn and explore the process of generating attractive regression plots
* Explore waffle charts and word cloud along with their application
* Explore the process of creating a Choropleth map using Folium

# **Advanced Visualization Tools**

## **Waffle Charts and Word Cloud**

**Waffle Charts and Word Clouds**:

Waffle Charts

* **Definition**: A visualization technique representing categorical data using square tiles or cells.
* **Structure**:
  + Resembles a grid of equal-sized squares.
  + Each square represents a specific value or category.
  + Size or color indicates the magnitude or proportion of each category.
* **Use Cases**:
  + **Market Share Analysis**: Visualize the proportion of companies/products in a market.
  + **Demographic Representation**: Display age groups or ethnicities within a population.
  + **Project Progress Tracking**: Represent completion status of tasks or milestones.
  + **Budget Allocation**: Show allocation of resources across categories.
  + **Survey Responses**: Summarize distribution of answers to multiple-choice questions.
  + **Election Results**: Visualize voting outcomes for candidates or parties.
  + **Product Sales Analysis**: Illustrate sales distribution across categories or regions.

Creating Waffle Charts in Python

* **Library**: Use the pywaffle library.
* **Implementation**:
  + Import the Waffle class.
  + Use the values parameter for data (e.g., total immigrants from Canada).
  + Set rows and columns to determine the grid size.
  + Customize the chart with a title and legend positioning.

Word Clouds

* **Definition**: A data visualization method to present textual data visually.
* **Functionality**:
  + Displays the importance of words based on frequency.
  + Words appear larger and bolder based on how often they appear in the text.
* **Use Cases**:
  + **Social Media Analysis**: Extract and visualize popular topics or sentiments.
  + **Customer Feedback Analysis**: Summarize reviews to identify recurring themes.
  + **Content Analysis**: Analyze articles or blogs to uncover prevalent keywords.
  + **Market Research**: Analyze survey responses or interviews for key insights.
  + **Resume/Job Description Analysis**: Highlight important skills or keywords.

Summary

* Waffle charts effectively show proportions of categories, making data easy to interpret.
* Word clouds provide a visual summary of textual data, highlighting key terms based on frequency.

## **Seaborn and Regression Plots**

Seaborn and Regression Plots

* **Seaborn Overview**:
  + Seaborn is a data visualization library based on Matplotlib.
  + It offers built-in themes and color palettes to enhance the visual appeal of plots.
  + Seaborn allows for efficient plotting, often requiring less code than Matplotlib.
* **Integration with Statistical Libraries**:
  + Works well with libraries like NumPy and SciPy for statistical analysis.
  + Provides specialized plot types such as regression plots, distribution plots, and categorical plots.
* **Creating Regression Plots**:
  + Example DataFrame: df\_total (columns: year, total immigration).
  + **Code to create a regression plot**:
  + import seaborn as sns

sns.regplot(x='year', y='total', data=df\_total)

* + This code generates a scatter plot with a regression line and a 95% confidence interval.
* **Customization Options**:
  + You can customize the plot using parameters like color and marker.
  + **Example of changing color and marker**:

sns.regplot(x='year', y='total', data=df\_total, color='green', marker='+')

* **Creating Categorical Plots**:
  + Example of plotting counts of categorical data (e.g., continents).
  + **Code for a bar plot**:

sns.countplot(x='continent', data=df\_Canada)

* + This code creates a bar plot showing the count of records for each continent.

Explanation of the Code

1. **Importing Seaborn**:
   * import seaborn as sns: This line imports the Seaborn library and allows you to use it with the alias sns.
2. **Creating a Regression Plot**:
   * sns.regplot(x='year', y='total', data=df\_total):
     + x='year': Specifies the column for the x-axis.
     + y='total': Specifies the column for the y-axis.
     + data=df\_total: Indicates the DataFrame to use for the plot.
3. **Customizing the Plot**:
   * color='green': Changes the color of the regression line and points to green.
   * marker='+': Changes the shape of the markers to a plus sign.
4. **Creating a Count Plot**:
   * sns.countplot(x='continent', data=df\_Canada):
     + x='continent': Specifies the categorical variable for the x-axis.
     + data=df\_Canada: Indicates the DataFrame containing the categorical data.

Conclusion

Seaborn is a powerful tool for creating visually appealing statistical graphics with minimal code. By understanding how to use its functions and customize plots, you can effectively visualize data and uncover insights.

# **Visualizing Geospatial Data**

## **Introduction to Folium**

* **Folium** is a powerful data visualization library in Python for visualizing geospatial data.
* You can create interactive maps using latitude and longitude values.
* Different map styles can be created using the tiles parameter.
* The default map style is OpenStreetMap.
* You can set the initial zoom level using the zoom\_start parameter.

Code Example:

Here’s a simple example of how to create a map centered around Canada using Folium:

import folium

# Create a map centered around Canada

canada\_map = folium.Map(location=[56.1304, -106.3468], zoom\_start=4)

# Display the map

canada\_map

Explanation:

* **Importing Folium**: The first line imports the Folium library.
* **Creating the Map**: The folium.Map() function creates a map object. The location parameter takes a list of latitude and longitude values for Canada, and zoom\_start sets the initial zoom level.
* **Displaying the Map**: Simply calling the map object will display it in a Jupyter Notebook or similar environment.

## **Maps with Markers**

* **Folium**: A Python library used for creating interactive maps.
* **Creating a Map**:
  + Import Folium.
  + Create a map object centered around specific coordinates (latitude and longitude).
  + Use zoom\_start to set the initial zoom level.
* import folium
* # Create a map centered around Canada

canada\_map = folium.Map(location=[56.1304, -106.3468], zoom\_start=4)

* **Adding Markers**:
  + Use folium.Marker to add markers to the map.
  + Specify the location (latitude and longitude) and pop-up text for the marker.
* # Add a marker for Ontario
* ontario\_marker = folium.Marker(location=[51.2538, -85.3232], popup='Ontario')

ontario\_marker.add\_to(canada\_map)

* **Feature Groups**:
  + Create a FeatureGroup to organize markers.
  + Add children (markers) to the feature group and then add the feature group to the map.
* # Create a feature group
* ontario\_group = folium.FeatureGroup(name='Ontario')
* # Add a circular marker
* folium.CircleMarker(location=[51.2538, -85.3232], radius=10, color='red', fill=True, fill\_color='red', popup='Ontario').add\_to(ontario\_group)
* # Add feature group to the map

ontario\_group.add\_to(canada\_map)

* **Displaying Multiple Markers**:
  + Create a list of locations and loop through it to add multiple markers.
  + Use MarkerCluster to group markers that are close together.
* from folium.plugins import MarkerCluster
* # Create a marker cluster
* marker\_cluster = MarkerCluster().add\_to(canada\_map)
* # List of locations
* locations = [[51.2538, -85.3232], [45.4215, -75.6972]] # Example coordinates
* # Add markers to the cluster
* for loc in locations:

folium.Marker(location=loc).add\_to(marker\_cluster)

Summary:

* **Folium** allows for easy creation of interactive maps with markers.
* Markers enhance interactivity and provide context.
* Feature groups help organize markers, and clustering prevents overcrowding.

## **Choropleth Maps**

* **Definition**: A **choropleth map** is a thematic map where areas are shaded or patterned based on the measurement of a statistical variable (e.g., population density, income).
* **Color Intensity**: The darker the color, the higher the measurement of the variable being represented.
* **Examples**:
  + **Infant Mortality Rate**: A choropleth map showing higher rates in darker shades, particularly in African countries.
  + **Population Density in the US**: States in the eastern US are generally more populous, with California as an exception.

Creating Choropleth Maps with Folium

* **Folium Library**: A Python library for creating interactive maps.
* **GeoJSON Requirement**: To create a choropleth map, you need a GeoJSON file that contains geospatial data for the region of interest.

Example Code to Create a Choropleth Map

Here’s a simplified example of how to create a choropleth map using Folium:

import folium

import pandas as pd

# Sample DataFrame with immigration data

data = {

'Country': ['Country1', 'Country2', 'Country3'],

'Immigration': [1000, 2000, 3000]

}

df\_canada = pd.DataFrame(data)

# Create a base map

m = folium.Map(location=[56.1304, -106.3468], zoom\_start=4)

# Load GeoJSON data (replace 'path\_to\_geojson' with your GeoJSON file path)

geojson\_data = 'path\_to\_geojson'

# Create a choropleth map

folium.Choropleth(

geo\_data=geojson\_data,

data=df\_canada,

columns=['Country', 'Immigration'],

key\_on='feature.properties.name', # Adjust based on your GeoJSON structure

fill\_color='YlGn', # Color scheme

fill\_opacity=0.7,

line\_opacity=0.2,

).add\_to(m)

# Save the map to an HTML file

m.save('choropleth\_map.html')

Explanation of the Code

* **Import Libraries**: Import folium for mapping and pandas for data manipulation.
* **DataFrame Creation**: Create a DataFrame containing country names and immigration numbers.
* **Base Map**: Initialize a Folium map centered on Canada.
* **GeoJSON Data**: Load your GeoJSON file that contains the geographical boundaries.
* **Choropleth Function**: Use folium.Choropleth to create the choropleth map, specifying the data and how it relates to the GeoJSON.
* **Save Map**: Save the generated map as an HTML file for viewing.

Summary: Advanced Visualizations and Geospatial Data

Congratulations! You have completed this module. At this point in the course, you know:

* Folium is a data visualization library in Python that helps people visualize geospatial data.
* With Folium, you can create maps of different styles, such as street-level maps, stamen maps, and more.
* A feature of Folium is that you can create different map styles using the tiles parameter.
* With Folium, you can easily add markers on maps.
* The ‘location’ parameter specifies the latitude and longitude coordinates of the center point of the map.
* Markers play a vital role in enhancing interactivity and adding context to maps.
* The folium.Marker() function specifies location parameters.
* The popup parameter provides a label upon being clicked.
* Markers can be created using “feature group.”
* A choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable.
* When creating a choropleth map, Folium requires a GeoJson file that includes geospatial data of the region.
* The Mapbox Bright Tileset displays the name of every country when used on a map.

Mark as completed

Like

Dislike

Report an issue