

Step 1: Install the Required Libraries

installed the libraries folium and json for geospatial analysis. Using Anaconda

```

Anaconda Prompt

(base) C:\Users\Asus>conda install -c conda-forge folium
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Asus\anaconda3\newanaconda3

  added / updated specs:
    - folium

The following packages will be downloaded:

  package----- build-----
branca-0.7.2 pyhd8ed1ab_0 28 KB conda-forge
ca-certificates-2024.8.30 h56e8100_0 155 KB conda-forge
certifi-2024.8.30 pyhd8ed1ab_0 160 KB conda-forge
conda-24.7.1 py312h2e8e12_0 1.2 MB conda-forge
folium-0.17.0 pyhd8ed1ab_0 77 KB conda-forge
libexpat-2.6.2 h63175ca_0 136 KB conda-forge
libsqlite-3.46.1 h2466b09_0 856 KB conda-forge
libzlib-1.2.13 h2466b09_6 55 KB conda-forge
openssl-3.2.2 h2466b09_0 8.0 MB conda-forge
python-3.12.3 h2628c8c_0_cppython 15.4 MB conda-forge
python_abi-3.12 5_cp312 7 KB conda-forge
ucrt-10.0.22621.0 h57928b3_0 1.2 MB conda-forge
vc14_runtime-14.40.33810 hcc2c482_20 733 KB conda-forge
vs2015_runtime-14.40.33810 h3bf8584_20 17 KB conda-forge
zlib-1.2.13 h2466b09_6 105 KB conda-forge
-----
Total: 28.1 MB

The following NEW packages will be INSTALLED:

branca conda-forge/noarch::branca-0.7.2-pyhd8ed1ab_0
folium conda-forge/noarch::folium-0.17.0-pyhd8ed1ab_0
libexpat conda-forge/win-64::libexpat-2.6.2-h63175ca_0
libsqlite conda-forge/win-64::libsqlite-3.46.1-h2466b09_0
libzlib conda-forge/win-64::libzlib-1.2.13-h2466b09_6
python_abi conda-forge/win-64::python_abi-3.12.5_cp312
ucrt conda-forge/win-64::ucrt-10.0.22621.0-h57928b3_0
vc14_runtime conda-forge/win-64::vc14_runtime-14.40.33810-hcc2c482_20
```

Step 3: Import the Necessary Libraries

```

In [20]: # Import libraries
import folium # For creating interactive maps
import json # For working with JSON/GeoJSON data
import os # For handling file paths

# Print confirmation
print("Libraries imported successfully.")
```

Libraries imported successfully.

Step 4: Load and Explore our GeoJSON Data

```

In [23]: # Define the path to your dataset
path = r'C:\Users\Asus\Music\achievement 6 project'

# Load the GeoJSON file
with open(os.path.join(path, 'Data', 'world-countries.json')) as f:
    data = json.load(f)

# Print the first feature to understand the structure of the data
print(data['features'][0])

{'type': 'Feature', 'properties': {'name': 'Afghanistan'}, 'geometry': {'type': 'Polygon', 'coordinates': [[[61.210817, 35.650072], [62.230651, 35.270664], [62.984662, 35.404041], [63.193538, 35.857166], [63.982896, 36.007957], [64.546479, 36.312073], [64.746105, 37.111818], [65.588948, 37.305217], [65.745631, 37.661164], [66.217385, 37.393791], [66.518607, 37.362784], [67.075782, 37.356144], [67.83, 37.144994], [68.135562, 37.023115], [68.859446, 37.344336], [69.196273, 37.151144], [69.518785, 37.608997], [70.116578, 37.588223], [70.270574, 37.735165], [70.376304, 38.138396], [70.806821, 38.486282], [71.348131, 38.258905], [71.239404, 37.953265], [71.541918, 37.905774], [71.448693, 37.065645], [71.844638, 36.738171], [72.193041, 36.948288], [72.63689, 37.047558], [73.260056, 37.495257], [73.948696, 37.421566], [74.980002, 37.41999], [75.158028, 37.133031], [74.575893, 37.020841], [74.067552, 36.836176], [72.920025, 36.720007], [71.846292, 36.509942], [71.262348, 36.074388], [71.498768, 35.650563], [71.613076, 35.153203], [71.115019, 34.733126], [71.156773, 34.348911], [70.881803, 33.988856], [69.930543, 34.02012], [70.323594, 33.358533], [69.687147, 33.105499], [69.262522, 32.501944], [69.317764, 31.901412], [68.926677, 31.620189], [68.556932, 31.71331], [67.792689, 31.58293], [67.683394, 31.303154], [66.938891, 31.304911], [66.381458, 30.738899], [66.346473, 29.887943], [65.046862, 29.472181], [64.350419, 29.560031], [64.148002, 29.340819], [63.550261, 29.468331], [62.549857, 29.318572], [60.874248, 29.829239], [61.781222, 30.73585], [61.699314, 31.379506], [60.941945, 31.548075], [60.863655, 32.18292], [60.536078, 32.981269], [60.9637, 33.528832], [60.52843, 33.676446], [60.803193, 34.404102], [61.210817, 35.650072]]]]], 'id': 'AFG'}
```

Step 5: Explore and Wrangle Our Data

```

In [27]: # Extract and display country names from the GeoJSON data
country_names = [feature['properties']['name'] for feature in data['features']]

# Print out the first 10 country names to check
print("First 10 countries in the dataset:")
print(country_names[:10])
```

First 10 countries in the dataset:

['Afghanistan', 'Angola', 'Albania', 'United Arab Emirates', 'Argentina', 'Armenia', 'Antarctica', 'French Southern and Antarctic Lands', 'Australia', 'Austria']

Step 6: Clean our Data

```

In [30]: # Check for any missing country names
missing_countries = [country for country in country_names if not country]

# Print a message if any missing countries are found
if missing_countries:
    print(f"Missing countries: {missing_countries}")
else:
    print("No missing country names found.")
```

No missing country names found.

Step 7: Create a Basic Map with Folium

```

In [42]: # Create a basic map centered around a location (latitude, longitude)
simple_map = folium.Map(location=[0, 0], zoom_start=2)

# Add the map title
folium.Marker([0, 0], popup="World Map").add_to(simple_map)

# Display the map
simple_map
```



Step 7: Create a Choropleth Map

```

In [47]: # Initialize the map centered around a specific location
choropleth_map = folium.Map(location=[0, 0], zoom_start=2)

# Example: Setting up a blank choropleth (you'll add data in the next step)
folium.Choropleth(
    geo_data=data,
    name='choropleth',
    data=None, # Replace with your actual data
    columns=['Country Name', 'Variable'], # Replace with actual columns
    key_on='feature.properties.name', # Ensure the country names match
    fill_color='YlGn',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Variable Name' # Customize this based on your data
).add_to(choropleth_map)

# Save the map as an HTML file
choropleth_map.save(os.path.join(path, 'choropleth_map.html'))

# Display the map
choropleth_map
```



file:///C:/Users/Asus/Music/achievement%206%20project/choropleth_map.html

Step 9: Discuss the Results in Markdown

Discussion of Results

The choropleth map visually represents the distribution of the chosen variable (e.g., population density, GDP) across different countries. The varying shades of color indicate different levels of the variable, providing a clear geographical overview.

This map helps to answer the research question about the global distribution of [Your Variable]. It also raises new questions, such as why certain regions have lower or higher values, and what factors might contribute to these differences.

In []: