

```
#Code from ChatGPT

import time

# Function to measure execution time
def measure_response_time(func, *args, **kwargs):
    # Record start time
    start_time = time.time()

    # Run the function
    result = func(*args, **kwargs)

    # Record end time
    end_time = time.time()

    # Calculate total execution time
    execution_time = end_time - start_time

    # Display the execution time
    print(f"Response time: {execution_time:.6f} seconds")

    return result

# Example of a function to test
def example_function(n):
    total = 0
    for i in range(n):
        total += i
    return total

import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import contextily as ctx

# Load GeoJSON data
geojson_file =
"/content/Countries_December_2023_Boundaries_UK_BUC_159444238
3739926403.geojson"
geo_df = gpd.read_file(geojson_file)

# Load 2019 and 2024 election results
results_2019 = pd.read_csv('/content/HoC-GE2019-results-by-
candidate.csv')
results_2024 = pd.read_csv('/content/HoC-GE2024-results-by-
candidate.csv')

# Assuming 'votes' is the column in both CSVs and 'county'
corresponds to 'CTRY23NM' in GeoJSON
votes_2019 = results_2019.groupby('Country name')
['Votes'].sum().reset_index()
votes_2024 = results_2024.groupby('Country name')
['Votes'].sum().reset_index()

# Merge votes with geographical data by country name
geo_df = geo_df.merge(votes_2019, left_on='CTRY23NM',
right_on='Country name', how='left', suffixes=('_', '2019', '_2024'))
geo_df = geo_df.merge(votes_2024, left_on='CTRY23NM',
right_on='Country name', how='left')

# Set up the plot with two subplots
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 12))

# Plot for 2019 (using 'Votes_x' as the 2019 votes column)
geo_df.plot(column='Votes_x', cmap='Blues', legend=True, ax=ax1,
             legend_kwds={'label': "2019 Votes"})
ctx.add_basemap(ax1, crs=geo_df.crs.to_string(),
               source=ctx.providers.OpenStreetMap.Mapnik)

# Add labels (country name and votes) to the 2019 plot
for x, y, label, votes in zip(geo_df.geometry.centroid.x,
                              geo_df.geometry.centroid.y, geo_df['CTRY23NM'], geo_df['Votes_x']):
    ax1.text(x, y, f'{label}\n{int(votes):,} votes', fontsize=8, ha='center')

ax1.set_title('2019 Election Votes by Country')

# Plot for 2024 (using 'Votes_y' as the 2024 votes column)
geo_df.plot(column='Votes_y', cmap='Reds', legend=True, ax=ax2,
             legend_kwds={'label': "2024 Votes"})
ctx.add_basemap(ax2, crs=geo_df.crs.to_string(),
               source=ctx.providers.OpenStreetMap.Mapnik)

# Add labels (country name and votes) to the 2024 plot
for x, y, label, votes in zip(geo_df.geometry.centroid.x,
                              geo_df.geometry.centroid.y, geo_df['CTRY23NM'], geo_df['Votes_y']):
    ax2.text(x, y, f'{label}\n{int(votes):,} votes', fontsize=8, ha='center')

ax2.set_title('2024 Election Votes by Country')

plt.tight_layout()
plt.show()

# Measure the response time of the example function
measure_response_time(example_function, 1000000)
```

Use this code to calculate vote difference and plot in separate maps



To calculate the vote difference between 2019 and 2024 and display it on a separate map, you can modify the code as follows:

1. Compute the vote difference.
2. Add a third subplot for the vote difference.
3. Adjust the color scheme to reflect vote differences (e.g., green for increase, red for decrease).

Here's the updated code:

```
python Copy code

import time # Function to measure execution time def measure_response_time(func, *args,
**kwargs): start_time = time.time() result = func(*args, **kwargs) end_time =
time.time() execution_time = end_time - start_time print(f"Response time:
{execution_time:.6f} seconds") return result # Example function for testing def
example_function(n): total = 0 for i in range(n): total += i return total import
geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import contextily
as ctx # Load GeoJSON data geojson_file =
"/content/Countries_December_2023_Boundaries_UK_BUC_1594442383739926403.geojson" geo_df
= gpd.read_file(geojson_file) # Load 2019 and 2024 election results results_2019 =
pd.read_csv('/content/HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('/content/HoC-GE2024-results-by-candidate.csv') # Assuming 'votes' is the
column in both CSVs and 'country' corresponds to 'CTRY23NM' in GeoJSON votes_2019 =
results_2019.groupby('Country name')['Votes'].sum().reset_index() votes_2024 =
results_2024.groupby('Country name')['Votes'].sum().reset_index() # Merge votes with
geographical data by country name geo_df = geo_df.merge(votes_2019, left_on='CTRY23NM',
right_on='Country name', how='left', suffixes=('_2019', '_2024')) geo_df =
```

```

geo_df.merge(votes_2024, left_on='CTRY23NM', right_on='Country name', how='left') #
Calculate vote difference between 2024 and 2019 geo_df['Vote_Difference'] =
geo_df['Votes_y'] - geo_df['Votes_x'] # Set up the plot with three subplots fig, (ax1,
ax2, ax3) = plt.subplots(1, 3, figsize=(30, 12)) # Plot for 2019 (using 'Votes_x' as the
2019 votes column) geo_df.plot(column='Votes_x', cmap='Blues', legend=True, ax=ax1,
legend_kwds={'label': "2019 Votes"}) ctx.add_basemap(ax1, crs=geo_df.crs.to_string(),
source=ctx.providers.OpenStreetMap.Mapnik) # Add labels (country name and votes) to the
2019 plot for x, y, label, votes in zip(geo_df.geometry.centroid.x,
geo_df.geometry.centroid.y, geo_df['CTRY23NM'], geo_df['Votes_x']): ax1.text(x, y,
f'{label}\n{int(votes):,} votes', fontsize=8, ha='center') ax1.set_title('2019 Election
Votes by Country') # Plot for 2024 (using 'Votes_y' as the 2024 votes column)
geo_df.plot(column='Votes_y', cmap='Reds', legend=True, ax=ax2, legend_kwds={'label':
"2024 Votes"}) ctx.add_basemap(ax2, crs=geo_df.crs.to_string(),
source=ctx.providers.OpenStreetMap.Mapnik) # Add labels (country name and votes) to the
2024 plot for x, y, label, votes in zip(geo_df.geometry.centroid.x,
geo_df.geometry.centroid.y, geo_df['CTRY23NM'], geo_df['Votes_y']): ax2.text(x, y,
f'{label}\n{int(votes):,} votes', fontsize=8, ha='center') ax2.set_title('2024 Election
Votes by Country') # Plot for vote difference geo_df.plot(column='Vote_Difference',
cmap='RdYlGn', legend=True, ax=ax3, legend_kwds={'label': "Vote Difference (2024 -
2019)}) ctx.add_basemap(ax3, crs=geo_df.crs.to_string(),
source=ctx.providers.OpenStreetMap.Mapnik) # Add labels (country name and vote
difference) to the difference plot for x, y, label, votes in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM'],
geo_df['Vote_Difference']): ax3.text(x, y, f'{label}\n{int(votes):,} diff', fontsize=8,
ha='center') ax3.set_title('Vote Difference (2024 vs 2019)') plt.tight_layout()
plt.show() # Measure the response time of the example function
measure_response_time(example_function, 1000000)

```

Key Changes:

1. **Vote Difference Calculation:** `geo_df['Vote_Difference'] = geo_df['Votes_y'] - geo_df['Votes_x']`
2. **Third Subplot:** Added a third map to display the vote difference.
3. **Color Map for Difference:** Used `RdYlGn` (red to green) to represent negative and positive vote changes.

This code will now plot the vote counts for 2019, 2024, and the difference between them on three separate maps.

< 1/2 > 🔊 📄 🗑️ 🔄

```

-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-4-abe2592f4a57> in <cell line: 34>()
      32 import pandas as pd
      33 import matplotlib.pyplot as plt
----> 34 import contextily as ctx
      35
      36 # Load GeoJSON data

ModuleNotFoundError: No module named 'contextily'

```



The error indicates that the `contextily` module is not installed in your environment. To resolve this issue, you need to install the `contextily` package, which is used for adding basemaps to GeoPandas plots.

Solution:

If you're working in a Jupyter notebook or a Python environment, run the following command to install `contextily`:

```
bash
```

Copy code

```
!pip install contextily
```

After installation, you can rerun your code. If you are using an environment where you can't directly install packages (like Google Colab), you can run the same installation command in a code cell.

Let me know if you encounter any other issues!