

Countries_December_2023_Boundaries_UK_B...
File

 HoC-GE2019-results-by-candidate.csv
Spreadsheet

 HoC-GE2024-results-by-candidate.csv
Spreadsheet

Using this geoJson file plot results comparison votes for both 2019 and 2024 where map should have number of votes and county name in legend

Columns of GeoJSON are " "Index(['FID', 'CTRY23CD', 'CTRY23NM', 'CTRY23NMW', 'BNG_E', 'BNG_N', 'LONG', 'LAT', 'GlobalID', 'geometry'],"

 To plot the results comparison of votes for both 2019 and 2024 based on the provided GeoJSON data and CSV results, I'll need to process the data in several steps:

1. Load and inspect the GeoJSON file for geographical boundaries and county information.
2. Load the 2019 and 2024 election results CSV files and filter out relevant information (e.g., votes, county).
3. Merge the election results with the GeoJSON data based on the common column (county name).
4. Generate a map with both 2019 and 2024 vote counts displayed, using the county names in the legend.

Let me start by loading and inspecting the data.

Analysis errored ^

```
python

import geopandas as gpd import pandas as pd # Load GeoJSON data and CSV files
geojson_file =
"/mnt/data/Countries_December_2023_Boundaries_UK_BUC_1594442383739926403.geojson"
csv_2019_file = "/mnt/data/HoC-GE2019-results-by-candidate.csv" csv_2024_file =
"/mnt/data/HoC-GE2024-results-by-candidate.csv" # Load the GeoJSON file geo_df =
gpd.read_file(geojson_file) # Load the election results CSVs for 2019 and 2024
results_2019 = pd.read_csv(csv_2019_file) results_2024 = pd.read_csv(csv_2024_file) #
Display the first few rows of each to understand the structure geo_df_head =
geo_df.head() results_2019_head = results_2019.head() results_2024_head =
results_2024.head() geo_df_head, results_2019_head, results_2024_head

    1>
--> 151     return GeometryArray(vectorized.from_shapely(data), crs=crs)

File ~/.local/lib/python3.11/site-packages/geopandas/_vectorized.py:145, in from_shapely(data)
    143 aout = np.empty(len(data), dtype=object)
    144 with compat.ignore_shapely2_warnings():
--> 145     aout[:] = out
    146 return aout

File ~/.local/lib/python3.11/site-packages/shapely/geometry/base.py:854, in BaseMultipartGeometry.__array__
    851 @property
    852 def __array_interface__(self):
    853     """Provide the Numpy array protocol."""
--> 854     raise NotImplementedError("Multi-part geometries do not themselves "
    855                               "provide the array interface")

NotImplementedError: Multi-part geometries do not themselves provide the array interface
```

It seems the GeoJSON file contains multi-part geometries that are not directly compatible with the current tools. I'll modify the approach to ensure compatibility with the data format, process the geometries, and then move forward with integrating the election results.

Let me handle the geometry issue and reprocess the data. [→]

Analysis errored ^

```
python

# Convert the GeoJSON file into a more standard format by exploding multi-part geometries to individual geometries geo_df = gpd.read_file(geojson_file).explode(index_parts=False) # Check again after exploding the geometries geo_df_head_fixed = geo_df.head() geo_df_head_fixed

--> 151     return GeometryArray(vectorized.from_shapely(data), crs=crs)

File ~/local/lib/python3.11/site-packages/geopandas/_vectorized.py:145, in from_shapely(data)
    143 aout = np.empty(len(data), dtype=object)
    144 with compat.ignore_shapely2_warnings():
--> 145     aout[:] = out
    146 return aout

File ~/local/lib/python3.11/site-packages/shapely/geometry/base.py:854, in BaseMultipartGeometry.__array_interface__(self):
    851 @property
    852 def __array_interface__(self):
    853     """Provide the Numpy array protocol."""
--> 854     raise NotImplementedError("Multi-part geometries do not themselves "
    855                           "provide the array interface")

NotImplementedError: Multi-part geometries do not themselves provide the array interface
```

It seems the multi-part geometries in the GeoJSON are causing some issues, and I'll need to handle them in a more customized manner. I'll approach the data differently by isolating the properties needed (county names and coordinates) and bypass the geometry issues for now to focus on merging the election data.

Let me extract the county and coordinate information and move forward. [→]

Analyzed ^

```
python

# Extract only relevant columns from the GeoJSON file, skipping the problematic geometries geo_df_simplified = gpd.read_file(geojson_file, ignore_geometry=True) # Focus on the relevant county information (CTRY23NM) and other details geo_df_simplified = geo_df_simplified[['CTRY23CD', 'CTRY23NM', 'LONG', 'LAT']] # Display the simplified GeoJSON structure geo_df_simplified.head()

Result
```

| | CTRY23CD | CTRY23NM | LONG | LAT |
|---|----------|------------------|----------|----------|
| 0 | E9200001 | England | -2.07811 | 53.23497 |
| 1 | N9200002 | Northern Ireland | -6.85570 | 54.61501 |
| 2 | S9200003 | Scotland | -3.97094 | 56.17743 |
| 3 | W9200004 | Wales | -3.99417 | 52.06741 |

I have successfully extracted the relevant county information (names and coordinates) from the GeoJSON file. The next step is to merge this data with the election results for both 2019 and 2024 to compare the votes by county.

I'll now proceed with merging the election data with the county information and create a map displaying the results. [→]

Analysis errored ^

```
python

# Inspect the structure of the election results to identify relevant columns
results_2019.columns, results_2024.columns
```

It seems like I can't do more advanced data analysis right now. Please try again later. If you'd like, I can help guide you through the steps to analyze the data manually or assist with another task! [→]

Give code

Using this geoJson file give python code for plotting results comparision votes for both 2019 and 2024 where map should has number of votes and county name

Columns of GeoJSON are " "Index(['FID', 'CTRY23CD', 'CTRY23NM', 'CTRY23NMW', 'BNG_E', 'BNG_N', 'LONG', 'LAT', 'GlobalID', 'geometry'], "

 Here's a Python code snippet to help you compare the 2019 and 2024 election results on a map using GeoPandas and Matplotlib:

Requirements:

1. geopandas
2. matplotlib
3. pandas
4. contextily (for basemap)

Install them using:

```
bash
pip install geopandas matplotlib pandas contextily
```

Copy code
Python Code:

```
python
import geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import
contextily as ctx # Load GeoJSON data geojson_file = "path_to_your_geojson_file.geojson"
geo_df = gpd.read_file(geojson_file) # Load 2019 and 2024 election results results_2019 =
pd.read_csv('path_to_HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('path_to_HoC-GE2024-results-by-candidate.csv') # Group votes by county
# assuming county column name is 'CTRY23NM' in geo_df and 'county' in results) votes_2019 =
results_2019.groupby('county')[['votes']].sum().reset_index() votes_2024 =
results_2024.groupby('county')[['votes']].sum().reset_index() # Merge votes with
geographical data geo_df = geo_df.merge(votes_2019, left_on='CTRY23NM',
right_on='county', how='left', suffixes=('_2019', '_2024')) geo_df =
geo_df.merge(votes_2024, left_on='CTRY23NM', right_on='county', how='left') # Plot the
map fig, ax = plt.subplots(1, 1, figsize=(12, 12)) # Plot geometries
geo_df.plot(column='votes_2019', cmap='Blues', legend=True, ax=ax, legend_kwds={'label':
"2019 Votes"}) # Add basemap ctx.add_basemap(ax, crs=geo_df.crs.to_string(),
source=ctx.providers.Stamen.TonerLite) # Add labels (county name and vote count) for x,
y, label in zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y,
geo_df['CTRY23NM']): ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison
of 2019 and 2024 Election Votes by County') plt.show() # Repeat for 2024 fig, ax =
plt.subplots(1, 1, figsize=(12, 12)) geo_df.plot(column='votes_2024', cmap='Reds',
legend=True, ax=ax, legend_kwds={'label': "2024 Votes"}) ctx.add_basemap(ax,
crs=geo_df.crs.to_string(), source=ctx.providers.Stamen.TonerLite) for x, y, label in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2024 Election
Votes by County') plt.show()
```

Copy code
Explanation:

1. **Loading Data:** The GeoJSON file is loaded using GeoPandas, and the election data is loaded as CSVs.
2. **Grouping by County:** Both 2019 and 2024 vote counts are grouped by county, assuming the column name for the county is 'county' .
3. **Merging Data:** The election results are merged with the GeoJSON to map the votes to counties.

4. **Plotting:** The number of votes for both years is visualized on separate maps, with counties labeled.

You can adjust paths and column names as needed.

```
KeyError           Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py
in get_loc(self, key)
    3790     try:
-> 3791         return self._engine.get_loc(casted_key)
    3792     except KeyError as err:
```

```
index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

```
index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

```
pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

```
pandas/_libs/hashtable_class_helper.pxi in
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

```
KeyError: 'Votes'
```

The above exception was the direct cause of the following exception:

```
KeyError           Traceback (most recent call last)
5 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py
in get_loc(self, key)
    3796     ):
-> 3797         raise InvalidIndexError(key)
    3798     raise KeyError(key) from err
    3799     except TypeError:
    3800         # If we have a listlike key, _check_indexing_error will
```

```
raise
```

```
KeyError: 'Votes'
```



The error you're seeing suggests that the column 'Votes' is not present in your election results CSV file. The actual name of the column containing the vote counts might be different.

Steps to Fix:

- Inspect Column Names:** First, check what the actual column names are in your CSV files for 2019 and 2024.

```
python
print(results_2019.columns) print(results_2024.columns)
```

Copy code

- Update Column Names in the Code:** Replace 'votes' in the code with the actual name of the column that contains the vote counts.

For example, if the column name is 'total_votes' or something similar, you should update this line:

```
python
votes_2019 = results_2019.groupby('county')['votes'].sum().reset_index() votes_2024 =
results_2024.groupby('county')['votes'].sum().reset_index()
```

Copy code

To:

```
python

votes_2019 = results_2019.groupby('county')['total_votes'].sum().reset_index()
votes_2024 = results_2024.groupby('county')['total_votes'].sum().reset_index()
```

Modified Code After Correction:

```
python

# Example: if 'total_votes' is the correct column name votes_2019 =
results_2019.groupby('county')['total_votes'].sum().reset_index() votes_2024 =
results_2024.groupby('county')['total_votes'].sum().reset_index()
```

Once you inspect the correct column names and update the code, it should work without the `KeyError`.

```
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py
in get_loc(self, key)
    3796      ):
    3797          raise InvalidIndexError(key)
-> 3798          raise KeyError(key) from err
    3799      except TypeError:
    3800          # If we have a listlike key, _check_indexing_error will
raise

KeyError: 'votes_2019'
```



The `KeyError: 'votes_2019'` means that the column '`'votes_2019'`' does not exist after the merge step. This likely indicates an issue with how the columns are named during the merge process.

Let's review the steps that might have led to this and fix the code:

Problem:

The column `votes_2019` doesn't exist because the merge didn't add the `votes_2019` or `votes_2024` columns.

Fix:

We need to carefully ensure the columns are being named correctly during the merge. Also, confirm the correct column names are used from both the election results and the GeoJSON file.

Updated Code with Detailed Merge:

```
python

import geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import
contextily as ctx # Load GeoJSON data geojson_file = "path_to_your_geojson_file.geojson"
geo_df = gpd.read_file(geojson_file) # Load 2019 and 2024 election results results_2019 =
pd.read_csv('path_to_HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('path_to_HoC-GE2024-results-by-candidate.csv') # Inspect columns in election
results print(results_2019.columns) print(results_2024.columns) # Assuming 'votes' is
the column in both CSVs and 'county' corresponds to 'CTRY23NM' in GeoJSON votes_2019 =
results_2019.groupby('county')['votes'].sum().reset_index() votes_2024 =
results_2024.groupby('county')['votes'].sum().reset_index() # Merge votes with
geographical data # Ensure column names match between geo_df and results data geo_df =
geo_df.merge(votes_2019, left_on='CTRY23NM', right_on='county', how='left', suffixes=
('_2019', '_2024')) geo_df = geo_df.merge(votes_2024, left_on='CTRY23NM',
right_on='county', how='left') # Plot the map for 2019 fig, ax = plt.subplots(1, 1,
figsize=(12, 12)) # Plot geometries geo_df.plot(column='votes_2019', cmap='Blues',
legend=True, ax=ax, legend_kwds={'label': '2019 Votes'}) # Add basemap
ctx.add_basemap(ax, crs=geo_df.crs.to_string(), source=ctx.providers.Stamen.TonerLite) #
Add labels (county name and vote count) for x, y, label in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2019 Election
```

```
'Votes by County') plt.show() # Repeat for 2024 fig, ax = plt.subplots(1, 1, figsize=(12, 12))
geo_df.plot(column='votes_2024', cmap='Reds', legend=True, ax=ax, legend_kwds={'label': "2024 Votes"})
ctx.add_basemap(ax, crs=geo_df.crs.to_string(), source=ctx.providers.Stamen.TonerLite)
for x, y, label in zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
    ax.text(x, y, label, fontsize=8, ha='center')
plt.title('Comparison of 2024 Election Votes by County')
plt.show()
```

Key Updates:

1. **votes_2019 and votes_2024 Grouping:** Make sure you're grouping by the correct column (`county` in the election data, assuming this is how counties are labeled).
2. **Merging:** I added `suffixes=('_2019', '_2024')` to the merge process to ensure the columns are named correctly during the merge.
3. **Plotting:** Each map now uses the appropriate column (`votes_2019` and `votes_2024`).

Make sure to check the actual column names in your data (especially for vote counts and county names) and update the code accordingly.

```
-----
```

| | |
|--|---|
| KeyError /usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key) 3790 try: -> 3791 return self._engine.get_loc(casted_key) 3792 except KeyError as err: index.pyx in pandas._libs.index.IndexEngine.get_loc() index.pyx in pandas._libs.index.IndexEngine.get_loc() pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item() pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'votes_2019' | Traceback (most recent call last) /usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key) 3790 try: -> 3791 return self._engine.get_loc(casted_key) 3792 except KeyError as err: index.pyx in pandas._libs.index.IndexEngine.get_loc() index.pyx in pandas._libs.index.IndexEngine.get_loc() pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item() pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'votes_2019' |
|--|---|

The above exception was the direct cause of the following exception:

```
-----
```

| | |
|--|---|
| KeyError 5 frames /usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key) 3796): 3797 raise InvalidIndexError(key) -> 3798 raise KeyError(key) from err 3799 except TypeError: 3800 # If we have a listlike key, _check_indexing_error will raise | Traceback (most recent call last) /usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key) 3790 try: -> 3791 return self._engine.get_loc(casted_key) 3792 except KeyError as err: index.pyx in pandas._libs.index.IndexEngine.get_loc() index.pyx in pandas._libs.index.IndexEngine.get_loc() pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item() pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item() KeyError: 'votes_2019' |
|--|---|



The `KeyError: 'votes_2019'` indicates that the column `votes_2019` is not being created or correctly referenced in the merged data.

How to fix:

1. **Check the column names after the merge:** The `merge()` function uses `suffixes` to append `_2019` and `_2024` to the vote columns. However, it may not be applied correctly, or

there might be a mismatch in column names.

- Print the resulting `geo_df` after merging: This will help to verify the column names.

Updated Code with Debugging:

```
python

import geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import contextily as ctx # Load GeoJSON data geojson_file = "path_to_your_geojson_file.geojson"
geo_df = gpd.read_file(geojson_file) # Load 2019 and 2024 election results results_2019 =
= pd.read_csv('path_to_HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('path_to_HoC-GE2024-results-by-candidate.csv') # Inspect columns in election
results print("2019 Election Results Columns:", results_2019.columns) print("2024
Election Results Columns:", results_2024.columns) # Assuming 'votes' is the correct
column in both CSVs and 'county' corresponds to 'CTRY23NM' in GeoJSON votes_2019 =
results_2019.groupby('county')[['votes']].sum().reset_index() votes_2024 =
results_2024.groupby('county')[['votes']].sum().reset_index() # Merge votes with
geographical data (geo_df) by county name geo_df = geo_df.merge(votes_2019,
left_on='CTRY23NM', right_on='county', how='left', suffixes=('_2019', '_2024')) geo_df =
geo_df.merge(votes_2024, left_on='CTRY23NM', right_on='county', how='left') # Print the
first few rows of the merged GeoDataFrame to inspect the column names
print(geo_df.head()) # Ensure the correct column name exists before plotting if
'votes_2019' in geo_df.columns: # Plot the map for 2019 fig, ax = plt.subplots(1, 1,
figsize=(12, 12)) # Plot geometries with 2019 votes data
geo_df.plot(column='votes_2019', cmap='Blues', legend=True, ax=ax, legend_kwds={'label':
"2019 Votes"}) # Add basemap ctx.add_basemap(ax, crs=geo_df.crs.to_string(),
source=ctx.providers.Stamen.TonerLite) # Add labels (county name) for x, y, label in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2019 Election
Votes by County') plt.show() else: print("Column 'votes_2019' not found in merged
GeoDataFrame.") # Repeat for 2024, similarly checking if the column exists if
'votes_2024' in geo_df.columns: fig, ax = plt.subplots(1, 1, figsize=(12, 12))
geo_df.plot(column='votes_2024', cmap='Reds', legend=True, ax=ax, legend_kwds={'label':
"2024 Votes"}) ctx.add_basemap(ax, crs=geo_df.crs.to_string(),
source=ctx.providers.Stamen.TonerLite) for x, y, label in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2024 Election
Votes by County') plt.show() else: print("Column 'votes_2024' not found in merged
GeoDataFrame.")
```

Key Debugging Steps:

- Check the election result columns:** You print out the columns of both the 2019 and 2024 datasets to ensure the `votes` column exists.
- Check the merged GeoDataFrame:** After merging the votes data with the geographic data, you print the first few rows to inspect the column names. This ensures that the vote columns (`votes_2019`, `votes_2024`) exist after merging.
- Conditional Plotting:** If the vote columns exist, the map will plot. If they don't, the script will notify you that the column wasn't found.

By inspecting the output of the printed columns and rows, you can identify the correct column names and ensure the data is being merged properly.

`'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 (geo_df) by county name
`geo_df = geo_df.merge(votes_2019, left_on='CTRY23NM',`
`right_on='Country name', how='left', suffixes=('_2019', '_2024'))`

Plotting Vote Results

```
geo_df = geo_df.merge(votes_2024, left_on='CTRY23NM',
right_on='Country name', how='left')

      FID  CTRY23CD     CTRY23NM   CTRY23NMW  BNG_E
BNG_N \
0   1  E92000001    England    Lloegr  394883  370883
1   2  N92000002  Northern Ireland Gogledd Iwerddon  86544
535337
2   3  S92000003    Scotland   Yr Alban  277744  700060
3   4  W92000004     Wales    Cymru  263405  242881

      LONG     LAT           GlobalID \
0 -2.07811 53.23497 c58f3ff1-92d1-42f1-aec6-f398f31504ff
1 -6.85570 54.61501 1c622ff0-0599-43a3-b6b1-0ff9bcce7e83
2 -3.97094 56.17743 c14596b0-f745-48b3-846a-175d8f347e45
3 -3.99417 52.06741 4abcb9a1-bace-44c6-b7d2-d1152017adbe

      geometry  Country name_x \
0  MULTIPOLYGON (((-6.34905 49.89822, -6.32842 49...    England
1  MULTIPOLYGON (((-5.52389 54.67041, -5.52451 54... Northern
Ireland
2  MULTIPOLYGON (((-3.06033 54.98452, -3.06337 54...    Scotland
3  MULTIPOLYGON (((-4.33394 51.56503, -4.31132 51...     Wales

      Votes_x  Country name_y  Votes_y
0  26911657    England  24294926
1  799035  Northern Ireland  779840
2  2759061    Scotland  2414810
3  1544357     Wales  1319076
Column 'votes_2019' not found in merged GeoDataFrame.
Column 'votes_2024' not found in merged GeoDataFrame.
Response time: 0.074096 seconds
499999500000
```



The error you're seeing occurs because the columns after merging are not named `votes_2019` and `votes_2024`. Instead, the merged columns are named `Votes_x` and `Votes_y`. This happens because the `suffixes` parameter was applied to the wrong columns during the merge.

Fix:

Update the column names to reference `Votes_x` (for 2019) and `Votes_y` (for 2024).

Updated Code:

```
python
import geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import
contextily as ctx # Load GeoJSON data geojson_file = "path_to_your_geojson_file.geojson"
geo_df = gpd.read_file(geojson_file) # Load 2019 and 2024 election results
results_2019 = pd.read_csv('path_to_HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('path_to_HoC-GE2024-results-by-candidate.csv') # Group by 'Country name' and
sum votes 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') # Inspect merged data frame to confirm column names print(geo_df.columns) #
Plot for 2019 (using 'Votes_x' as the 2019 votes column) fig, ax = plt.subplots(1, 1,
figsize=(12, 12)) geo_df.plot(column='Votes_x', cmap='Blues', legend=True, ax=ax,
legend_kwds={'label': "2019 Votes"}) # Add basemap ctx.add_basemap(ax,
crs=geo_df.crs.to_string(), source=ctx.providers.Stamen.TonerLite) # Add labels (county
name) for x, y, label in zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y,
geo_df['CTRY23NM']): ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison
of 2019 Election Votes by County') plt.show() # Plot for 2024 (using 'Votes_y' as the
2024 votes column) fig, ax = plt.subplots(1, 1, figsize=(12, 12))
```

```
geo_df.plot(column='Votes_y', cmap='Reds', legend=True, ax=ax, legend_kwds={'label': '2024 Votes'}) ctx.add_basemap(ax, crs=geo_df.crs.to_string(), source=ctx.providers.Stamen.TonerLite) # Add labels (county name) for x, y, label in zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']): ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2024 Election Votes by County') plt.show()
```

Key Changes:

1. **Column Names:** In the merged data, `Votes_x` represents votes for 2019, and `Votes_y` represents votes for 2024.
2. **Plotting:** The plots now reference `Votes_x` for 2019 and `Votes_y` for 2024.

This should resolve the issue with column names and allow the plots to render correctly.

```
Index(['FID', 'CTRY23CD', 'CTRY23NM', 'CTRY23NMW', 'BNG_E',
       'BNG_N', 'LONG',
       'LAT', 'GlobalID', 'geometry', 'Country name_x', 'Votes_x',
       'Country name_y', 'Votes_y'],
      dtype='object')
```

```
-----
KeyError          Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/xyzservices/lib.py in
__getattr__(self, key)
    43     try:
--> 44         return self.__getitem__(key)
    45     except KeyError as err:
```

```
KeyError: 'Stamen'
```

The above exception was the direct cause of the following exception:

```
AttributeError          Traceback (most recent call last)
1 frames
/usr/local/lib/python3.10/dist-packages/xyzservices/lib.py in
__getattr__(self, key)
    44     return self.__getitem__(key)
    45     except KeyError as err:
--> 46         raise AttributeError(key) from err
    47
    48     def __dir__(self):
```

```
AttributeError: Stamen
```



The `AttributeError: 'Stamen'` error occurs because the basemap provider `stamen.TonerLite` is not recognized. It seems the `contextily` library doesn't have the `Stamen` provider available.

Fix:

We can replace `stamen.TonerLite` with another valid basemap provider that is available in `contextily`.

Try using `ctx.providers.OpenStreetMap.Mapnik`, which is a commonly available provider.

Updated Code:

```
python
import geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import
contextily as ctx # Load GeoJSON data geojson_file = "path_to_your_geojson_file.geojson"
geo_df = gpd.read_file(geojson_file) # Load 2019 and 2024 election results results_2019 =
= pd.read_csv('path_to_HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('path_to_HoC-GE2024-results-by-candidate.csv') # Group by 'Country name' and
```

```

sum votes 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') # Inspect merged dataframe to confirm column names print(geo_df.columns) #
Plot for 2019 (using 'Votes_x' as the 2019 votes column) fig, ax = plt.subplots(1, 1,
figsize=(12, 12)) geo_df.plot(column='Votes_x', cmap='Blues', legend=True, ax=ax,
legend_kwds={'label': "2019 Votes"}) # Add basemap (using OpenStreetMap instead of
Stamen) ctx.add_basemap(ax, crs=geo_df.crs.to_string(),
source=ctx.providers.OpenStreetMap.Mapnik) # Add labels (county name) for x, y, label in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2019 Election
Votes by County') plt.show() # Plot for 2024 (using 'Votes_y' as the 2024 votes column)
fig, ax = plt.subplots(1, 1, figsize=(12, 12)) geo_df.plot(column='Votes_y',
cmap='Reds', legend=True, ax=ax, legend_kwds={'label': "2024 Votes"})
ctx.add_basemap(ax, crs=geo_df.crs.to_string(),
source=ctx.providers.OpenStreetMap.Mapnik) # Add labels (county name) for x, y, label in
zip(geo_df.geometry.centroid.x, geo_df.geometry.centroid.y, geo_df['CTRY23NM']):
ax.text(x, y, label, fontsize=8, ha='center') plt.title('Comparison of 2024 Election
Votes by County') plt.show()

```

Key Changes:

- Basemap Provider:** Replaced `Stamen.TonerLite` with `OpenStreetMap.Mapnik`, which should work with `contextily`.
- Plotting:** The rest of the code remains the same, using `Votes_x` and `Votes_y` for 2019 and 2024 votes, respectively.

This should resolve the issue with the basemap. Let me know if this works!

Plot side by side with legend having country name and number of votes

- 💡 To display the maps side by side and include a legend with both the country names and the number of votes, we can adjust the plot by creating a figure with two subplots and modify the legend to display the votes.

Here's the updated code to display the maps side by side:

Updated Code:

```

python
import geopandas as gpd import pandas as pd import matplotlib.pyplot as plt import
contextily as ctx # Load GeoJSON data geojson_file = "path_to_your_geojson_file.geojson"
geo_df = gpd.read_file(geojson_file) # Load 2019 and 2024 election results results_2019 =
pd.read_csv('path_to_HoC-GE2019-results-by-candidate.csv') results_2024 =
pd.read_csv('path_to_HoC-GE2024-results-by-candidate.csv') # Group by 'Country name' and
sum votes 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{n{int(votes)}:,} votes', fontsize=8, ha='center') ax2.set_title('2024 Election
Votes by Country') plt.tight_layout() plt.show()
```

Key Adjustments:

1. **Side-by-side layout:** I used `fig, (ax1, ax2)` to create two subplots side by side for 2019 and 2024 election data.
2. **Country name and votes in the legend:** The `for` loop now adds both the country name and the number of votes directly on the map for each country.
3. **Formatting of votes:** The votes are formatted with commas (e.g., `1,234,567 votes`) to make the numbers more readable.

This should display two maps side by side, with country names and votes labeled for both the 2019 and 2024 elections. Let me know how it looks!