GeoPandas (geoprocessing): Folium

Dr. Cornelis Stal

April 28, 2022

1 Kaartjes met Folium

1.1 Introductie

Folium bouwt voort op de kracht van het Python ecosysteem voor dataverwerking enerzijds en de cartografische visualisatiemogelijkheden van de leaflet.js bibliotheek. Deze combinatie laat toe om uw data te manipuleren in GeoPandas en vervolgens te visualiseren in een Leaflet-kaart via Folium.

In deze demo zullen we Geopandas gebruiken om geometriën in te laden(vereenvoudigde versie van Vlaamse aanduidingsobjecten). Vervolgens maken we een Folium-kaart, waarbij markers gebruikt worden om verschillende type aanduidingsobjecten weer te geven.

> Deze tutorial is een vertaling van de 'getting started guide' op https://python-visualization.github.io/folium.

1.2 Geometriën inladen

Voor deze demo maken we gebruik van een vereenvoudigde versie van de Vlaamse aanduidingsobjecten. In de aangepaste versie zijn een groot aantal objecten en velden weggelaten. De naam van de gemeente hebben we echter wel toegevoegd op basis van de authentieke dataset met Vlaamse gemeenten. We zullen het CSV-bestand inlezen met behulp van pandas, en vervolgens de pandas-`DataFrame` naar een Geopandas `GeoDataFrame`.

We beginnen met het importeren van de benodigde bibliotheken:

[ ]:
```
import pandas as pd
import geopandas as gpd
import folium
import matplotlib.pyplot as plt
```

Vervolgens laden we de data met aanduidingsobjecten, bereiden we de data voor en bekijken we de verschillende datatypes:

[ ]:
```
ao_raw = pd.read_csv('data/aanduidingsobjecten.csv')
ao = ao_raw.loc[:, ("NAAM", "TYPE_NAAM", "GEMEENTE", "LAT", "LON")]
ao.info()
```

OP basis van de `LAT`- en `LON`-velden kunnen we de entiteiten converteren naar een `GeoPandas GeoDataFrame`. Uiteraard bestuderen we de verschillende entiteiten in de dataset:

[ ]:
```
geometry = gpd.points_from_xy(ao.LON, ao.LAT)
gao = gpd.GeoDataFrame(ao[["NAAM", "TYPE_NAAM", "GEMEENTE", "LAT", "LON"]],␣
↪geometry=geometry)
gao.head()
```

De volgende unieke waarden zijn aanwezig voor het veld `TYPE_NAAM`:

[ ]:
```
gao.TYPE_NAAM.unique()
```

De aanduidingsobjecten zullen we projecteren op de Vlaamse gemeenten. We laden deze data ook in en visualiseren beide datasets:

[ ]:
```
refgem = gpd.read_file('data/refgem_2018.shp')
fig, ax = plt.subplots(figsize=(24,18))
refgem.plot(ax=ax, alpha=0.4, color='grey')
gao.plot(column='TYPE_NAAM', ax=ax, legend=True)
plt.title('Vlaamse aanduidingsobjecten')
```

Het valt op dat de aanduidingsobjecten en de Vlaamse gemeenten totaal niet correct gevisualiseerd worden. Dit heeft uiteraard te namen met de verschillende coördinaatreferentiesystemen. De data met aanduidingsopjecten staan in WGS84 (epsg:4326), terwijl de Vlaamse gemeenten in het Belgische Lambert'72 systeem (epsg:31370) geprojecteerd worden. Om dit op te lossen, projecteren we de Vlaamse gemeentegrenzen in WGS'84 met behulp van de `to_crs()`-methode. Het standaard CRS van de aanduidingsobjecten is WGS'84, wat we eveneens expliciet toekennen aan de dataset met de `set_crs()`-methode:

[ ]:
```
refgem = refgem.to_crs(4326)
gao = gao.set_crs(4326)
fig, ax = plt.subplots(figsize=(24,18))
refgem.plot(ax=ax, alpha=0.4, color='grey')
gao.plot(column='TYPE_NAAM', ax=ax, legend=True)
plt.title('Vlaamse aanduidingsobjecten')
```

We kunnen deze data uiteraard ook in een ander CRS projecteren, zoals het Belgische Lambert '72 CRS:

[ ]:
```
refgem_lb72 = refgem.to_crs(31370)
gao_lb72 = gao.to_crs(31370)
fig, ax = plt.subplots(figsize=(24,18))
refgem_lb72.plot(ax=ax, alpha=0.4, color='grey')
gao_lb72.plot(column='TYPE_NAAM', ax=ax, legend=True)
plt.title('Vlaamse aanduidingsobjecten')
```

1.3 Folium-kaart aanmaken

Folium beschikt over een aantal *built-in* tilesets voor onder meer OpenStreetMap, Mapbox, and Stamen. Hieronder demonstreren we dit met enkele voorbeelden.

Stamen Terrain:

```
[ ]: kaart = folium.Map(location = [51.052,3.722], tiles = "Stamen Terrain",␣
      ↪zoom_start = 14)
     kaart
```

OpenStreetMap:

```
[ ]: kaart = folium.Map(location = [51.052,3.722], tiles='OpenStreetMap' ,␣
      ↪zoom_start = 14)
     kaart
```

Stamen Toner:

```
[ ]: kaart = folium.Map(location = [51.052,3.722], tiles='Stamen Toner', zoom_start␣
      ↪= 14)
     kaart
```

Stamen Terrain:

```
[ ]: kaart = folium.Map(location = [51.052,3.722], tiles = "Stamen Terrain",␣
      ↪zoom_start = 14)
     kaart
```

Verbinden met andere tile services: Folium laat eveneens toe om te verbinden met andere tile services die niet standaard zijn opgenomen in de bibliotheek. We geven dan de URL van de service mee aan de `tiles` attribuut. In dit geval is het meegeven van een waarde aan de `attr`-attribuut vereist ter omschrijving van de rechten op de data. Bijvoorbeeld:

```
[ ]: kaart = folium.Map(location = [51.052,3.722], tiles = "https://www.ngi.be/tiles/
      ↪arcgis/rest/services/cartoweb__topo__default__3857__latest/MapServer/tile/
      ↪{z}/{y}/{x}",
         attr="<a href=\"www.ngi.be\">NGI/IGN</a> &copy;", zoom_start = 14)
     kaart
```

1.4 Folium kaart opslaan

Het resultaat van bovenstaande code kunnen we eenvoudig opslaan als een HTML-bestand:

```
[ ]: kaart.save("index.html")
```

1.5 Markers toevoegen

Om de verschillende typen aanduidingsobjecten te visualiseren, zullen we voor ieder type een Folium-marker aanmaken en deze toevoegen aan de kaart.

```
[ ]: # Maak een lijst met geometriën van de GeoDataFrame
     gdfList = [[point.xy[1][0], point.xy[0][0]] for point in gao.geometry ]

     # Itereer over de lijst en voeg een marker toe voor ieder aanduidingsobject,␣
      ↪met verschillende kleuren per type.
```

```
i = 0
for coordinates in gdfList:
    # Ken een kleur toe aan een marker op basis van het type aanduidingsobject
    if gao.TYPE_NAAM[i] == "Beschermd cultuurhistorisch landschap":
        type_color = "blue"
    elif gao.TYPE_NAAM[i] == "Beschermd stads- of dorpsgezicht":
        type_color = "orange"
    elif gao.TYPE_NAAM[i] == "Erfgoedlandschap":
        type_color = "pink"
    else:
        type_color = "purple"


    # Plaats de markers, inclusief een popup op de kaart
    kaart.add_child(folium.Marker(location = coordinates,
                            popup =
                            "Naam: " + str(gao.NAAM[i]) + '<br>' +
                            "Type: " + str(gao.TYPE_NAAM[i]) + '<br>' +
                            "Gemeente: " + str(gao.GEMEENTE[i]) + '<br>'
                            "Coordinates: " + str(gdfList[i]),
                            icon = folium.Icon(color = "%s" % type_color)))
    i = i + 1
kaart
```

1.6 Choropleetkaarten maken met Folium

Veronderstellen we nu dat we een voorstelling willen maken van het aantal aanduidingsobjecten per gemeente in Vlaanderen. We zullen dus per gemeente het aantal puntobjecten moeten tellen, en vervolgens een kleurenschaal definiëren om het geheel voor te stellen.

De `GeoPandas`-bibliotheek bevat geen functie om rechtstreeks het aantal punten in een polygoon te tellen. We zijn daarom genoodzaakt om via een omweg te werken:

[ ]:
```
refgemCount = gpd.sjoin(refgem, gao)
refgemCount = refgemCount.groupby(['NISCODE']).size().reset_index(name='AANTAL')
refgemCount.head()
```

Vanuit de `folium`-bibliotheek zullen we nu een nieuwe kaart aanmaken, waaraan we een `Choropleth`-object toe zullen kennen:

[ ]:
```
kaart = folium.Map(location=[51, 4], tiles="https://www.ngi.be/tiles/wmts/
 ↪cartoweb/1.0.0/topo/default_bw/3857/latest/{z}/{y}/{x}.png",
    attr="<a href=\"www.ngi.be\">NGI/IGN</a> &copy;", name="NGI", zoom_start=9)

folium.Choropleth(
    geo_data=refgem,
    name="choropleth",
    data=refgemCount,
```

```
    columns=["NISCODE", "AANTAL"],
    key_on="feature.properties.NISCODE",
    fill_color="YlGn",
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name="Aantal aanduidingsobjecten per gemeente",
).add_to(kaart)

folium.LayerControl().add_to(kaart)

kaart
```

De legende aan de rechterbovenzijde is automatisch aangemaakt op basis van de in het veld `AANTAL` beschikbare waarden. Folium zal de data standaard opdelen in 6 bins met gelijke grootte. We kunnen dit echter eenvoudig aanpassen:

[ ]:
```
bins = list(refgemCount["AANTAL"].quantile([0.00, 0.25, 0.50, 0.75, 1.00]))

kaart = folium.Map(location=[51, 4], name="NGI", zoom_start=9, tiles="https://
 ↪www.ngi.be/tiles/wmts/cartoweb/1.0.0/topo/default_bw/3857/latest/{z}/{y}/{x}.
 ↪png",
    attr="<a href=\"www.ngi.be\">NGI/IGN</a> &copy;")

folium.Choropleth(
    geo_data=refgem,
    name="Aantal aanduidingsobjecten",
    data=refgemCount,
    columns=["NISCODE", "AANTAL"],
    key_on="feature.properties.NISCODE",
    fill_color="YlOrRd",
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name="Aantal aanduidingsobjecten per gemeente",
    bins=bins,
    reset=True,
).add_to(kaart)

folium.LayerControl().add_to(kaart)

kaart
```

1.7 Folium Heatmaps

Folium wordt veel gebruikt voor het berekenen van zogenaamde *heatmaps*. Deze laten zich letterlijk vertalen als 'hittekaarten', maar 'densiteitskaart' zou een betere vertaling bieden. De *heatmap*-plugin maakt het mogelijk om een densiteitslaag te bereken voor een geveven dataset. Folium vereist hiervoor niets meer dan een verzameling coördinaten:

```
[ ]: from folium import plugins

kaart = folium.Map(location = [51, 4], tiles='Cartodb dark_matter', zoom_start␣
 ↪= 9)

heat_data = [[point.xy[1][0], point.xy[0][0]] for point in gao.geometry ]

heat_data
plugins.HeatMap(heat_data).add_to(kaart)

kaart
```

1.8 GeoJSON inladen in Folium

Folium laat toe om online bronnen rechtstreeks te visualiseren. Dit is vooral handig wanneer we GeoJSON-data op een kaart willen projecteren. De `folium.GeoJSON()`-methode gebruiken we om een bron aan te spreken en hier een naam aan toe te kennen. We kunnen echter ook de output van een `GET`-request uit de `requests`-bibliotheek gebruiken. In onderstaande voorbeeld projecteren we het administratief perceel uit het GRB op basis van de coordinaten van het aanduidingsobject met de naam 'Bijloke':

```
[ ]: import requests

kaart = folium.Map(location=[51.0449772, 3.7176290], tiles="https://tile.
 ↪informatievlaanderen.be/ws/raadpleegdiensten/wmts?SERVICE=WMTS&VERSION=1.0.
 ↪0&REQUEST=GetTile&LAYER=grb_bsk_grijs&STYLE=&FORMAT=image/
 ↪png&TILEMATRIXSET=GoogleMapsVL&TILEMATRIX={z}&TILEROW={y}&TILECOL={x}",
    attr="<a href=\"www.geopunt.be\">Informatie Vlaanderen</a> &copy;",␣
 ↪zoom_start=17)

aoBijloke = gao_lb72[gao_lb72["NAAM"].str.contains('Bijloke')]
aoPt = aoBijloke.to_wkt().values.tolist()[0][5]
cql_filter='INTERSECTS(SHAPE,%s)' % aoPt

url = 'https://geoservices.informatievlaanderen.be/overdrachtdiensten/GRB/wfs'
payload = {'SERVICE':'WFS','REQUEST':'GetFeature','VERSION':'2.0.0',
     'TYPENAMES':'GRB:ADP','cql_filter':cql_filter,
     'outputFormat':'application/json','srsName':'EPSG:4326'}
r = requests.get(url, params=payload)

folium.GeoJson(r.json(), name="ADP").add_to(kaart)

kaart
```

De zojuist ingeladen GeoJSON-inhoud kunnen we geheel naar eigen wens opmaken. Ook kunnen we zelf popup vensters voorbereiden:

[ ]:
```
def style_function(feature):
    return {
        "fillOpacity": 0.5,
        "weight": 0,
        "fillColor": "#0F0",
    }

parcelData = r.json()

geoJson = folium.map.FeatureGroup(name='ADP').add_to(kaart)

for fData in parcelData['features']:
    parcel = folium.GeoJson(fData, style_function=style_function)
    parcel.add_child(folium.Popup('<b>CAPAKEY: </b>' +␣
 ↪str(fData['properties']['CAPAKEY'])))
    geoJson.add_child(parcel)

geoJson.add_to(kaart)
kaart
```

[ ]:
```

```