folium

July 26, 2019

O ***Folium*** é um pacote que possibilita a criação de mapas *online*, facilitando a visualização dos dados manipulados no *Python* em um mapa que usa a biblioteca JavaScript ***Leaflet***.

A biblioteca possui vários conjuntos de blocos internos do OpenStreetMap, Mapbox e Stamen, além de suportar conjuntos de blocos customizados com as chaves da API Mapbox ou Cloudmade. ***Folium*** suporta sobreposições de imagem, vídeo, GeoJSON e TopoJSON.

{: .box-warning} **Aviso:** Esse *post* tem a finalidade de mostrar os comandos básicos e me deixar com uma "cola" rápida para meu uso cotidiano. Todas os códigos são exemplificativos e podem/devem ser alterados, indicando o nome dos arquivos e diretórios corretamente.

{: .box-warning} **Aviso:** É possível acessar esse *post* em formato .pdf e, ainda, no repositório do GitHub.

0.1 Importando Bibliotecas

As bibliotecas básicas, ou *packages*, necessárias para criação do mapa são: - O ***Pandas***, que tem a missão de trabalhar com dados, criar *subsets*, selecionar e filtros dados e; - O ***Folium***, que é a biblioteca que cria, na prática, o mapa!

[1]:
```
import pandas as pd
import folium
```

0.2 Criando um mapa

Basta um par de coordenadas – que pode ser obtida facilmente no *link* de qualquer endereço usando ***Google Maps*** – e um nível de zoom que o mapa já está criado.

[2]:
```
folium.Map(
    location=[-23.9619271,-46.3427499],      # Define coordenadas iniciais
    #min_zoom = 6,                           # Define qual o menor zoom
    #max_zoom = 14,                          # Define qual o maior zoom
    #no_wrap = True,
    #max_bounds = True,
    zoom_start=12                            # Define o zoom do início
)
```

[2]:
```
<folium.folium.Map at 0x7fa8c4296cf8>
```

Utilizando um conjunto de dados apresentado em **Jessica Temporal**, contendo coordenadas geográficas de empresas, podemos extrair uma empresa específica e plotar no mapa, ou ainda trabalhar de outras maneiras com esses dados.

[3]:
```
# Lendo e filtrando dados
empresas = pd.read_csv('../data/empresas.xz')
empresas = empresas[empresas['state'] == 'SP']
empresas = empresas[empresas['city'] == 'SANTOS']

empresas.dtypes
```

[3]:
```
name                           object
situation                      object
neighborhood                   object
address                        object
number                         object
zip_code                       object
city                           object
state                          object
cnpj                           object
status                         object
additional_address_details     object
main_activity                  object
latitude                      float64
longitude                     float64
dtype: object
```

0.2.1 Inserindo algumas coordenadas

[4]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
)

# Extrai informações de duas empresas
empresa1 = empresas.iloc[0]
empresa2 = empresas.iloc[1]

# Adiciona no mapa tais empresas
folium.Marker(
    location=[empresa1['latitude'], empresa1['longitude']],
).add_to(webmap)

folium.Marker(
    location=[empresa2['latitude'], empresa2['longitude']],
).add_to(webmap)

# Apresenta o mapa
webmap
```

[4]:
```
<folium.folium.Map at 0x7fa8c1a74898>
```

0.2.2 Inserindo multiplas coordenadas

[5]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
)

# Adiciona todas as empresas selecionadas
for _, empresa in empresas.iterrows():
    folium.Marker(
        location=[empresa['latitude'], empresa['longitude']],
        tooltip=empresa['neighborhood'],
    ).add_to(webmap)

# Apresenta o mapa
#webmap
```

0.3 Tipos diferentes de Marcadores

As feições que são possiveis de apresentar são àquelas típicas do geoprocessamento: - Pontos; - Linhas; - Polígonos

Abaixo são apresentados alguns tipos de marcadores.

0.3.1 Pontos Simples

[6]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
)

# Cria cores para as tags
colors = {
    'PONTA DA PRAIA': 'pink',
    'CENTRO': 'blue',
    'GONZAGA': 'green',
    'JOSÉ MENINO': 'red',
    'EMBARE': 'beige',
    'MACUCO': 'blue',
    'VILA MATHIAS': 'lightblue',
    'POMPEIA': 'red',
    'APARECIDA': 'purple',
}

# Adiciona as diferentes empresas com cores por bairros
for _, empresa in empresas.iterrows():
```

```
    if empresa['neighborhood'] in colors.keys():
        folium.Marker(
            location=[empresa['latitude'], empresa['longitude']],
            popup=empresa['name'],
            tooltip=empresa['neighborhood'],
            icon=folium.Icon(color=colors[empresa['neighborhood']], icon='leaf')
        ).add_to(webmap)

# Apresenta o mapa
#webmap
```

0.3.2 Marcador Circular

[7]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
)

# Adiciona as diferentes empresas com cores por bairros
for _, empresa in empresas.iterrows():
    if empresa['neighborhood'] in colors.keys():
        folium.CircleMarker(
        location=[empresa['latitude'], empresa['longitude']],
        radius=10,
        popup='<strong>Empresa</strong>',
        tooltip='Dica',
        fill=True,
        #fill_color='#428bca'
        fill_color=colors[empresa['neighborhood']]
    ).add_to(webmap)

# Apresenta o mapa
#webmap
```

0.3.3 Custom Icon

[8]:
```
#logoIcon = folium.features.CustomIcon('logo.png', icon_size=(50,50))
```

0.3.4 Vegas

O *folium* tem o vegas https://vega.github.io/vega/ como default

[9]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
```

```
)

# Importa bibliotecas e lê o json
import os
import json
vis = os.path.join('..','data', 'vis.json')


# Adiciona as diferentes empresas com gráficos no popup
for _, empresa in empresas.iterrows():
    if empresa['neighborhood'] in colors.keys():
        folium.Marker(
            location=[empresa['latitude'], empresa['longitude']],
            popup=folium.Popup(max_width=450).add_child(folium.Vega(json.
 ↪load(open(vis)), width=450, height=250))
).add_to(webmap)

# Apresenta o mapa
#webmap
```

0.3.5 Geojson

É possivel também inserir desenhos em formato ***GeoJson***, o que abre grandes possibilidades. Contudo, para rabiscos aleatórios, é possivel criar o arquivo usando http://geojson.io.

[10]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
)

# Importa bibliotecas e lê o json
import os
import json
shp = os.path.join('..', 'data', 'trajetos.json')

# Adiciona as diferentes empresas com gráficos no popup
folium.GeoJson(shp, name='Trajetos').add_to(webmap)

# Apresenta o mapa
webmap
```

[10]:
```
<folium.folium.Map at 0x7fa8ec1b40f0>
```

0.3.6 *Join* e Categorias

[11]:
```
import folium
import pandas as pd
import os

states = os.path.join('..', 'data', 'us-states.json')
unemployement_data = os.path.join('..', 'data', 'us_unemployment.csv')
state_data = pd.read_csv(unemployement_data)

m = folium.Map(location=[48, -102], zoom_start=3)

m.choropleth(
    geo_data=states,
    name='choropleth',
    data=state_data,
    columns=['State', 'Unemployment'],
    key_on='feature.id',
    fill_color='YlGn',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Unemployment Rate %'
)

folium.LayerControl().add_to(m)

m.save('../maps/map_us.html')
```

```
/home/michel/miniconda/envs/pablocarreira-py36/lib/python3.6/site-
packages/folium/folium.py:415: FutureWarning: The choropleth  method has been
deprecated. Instead use the new Choropleth class, which has the same arguments.
See the example notebook 'GeoJSON_and_choropleth' for how to do this.
  FutureWarning
```

0.4 Basemap

O mapa pode ter diferentes *basemaps*, que são, na essência, o mapa de fundo renderizado em *titles*. O *folium* utiliza, por *default*, o basemap do *OpenStreetMap*, contudo existe a possibilidade d eadicionar outros serviços, conforme se vê abaixo.

[12]:
```
folium.Map(
    location=[-23.9619271,-46.3427499],
    #tiles='Mapbox Bright',
    #tiles='Mapbox Control Room',
    #tiles='Stamen Toner',
    tiles='Stamen Terrain',
    #tiles='OpenStreetMap',
    zoom_start=12
```

```
)
```

[12]: `<folium.folium.Map at 0x7fa8c1ac8588>`

Um outro jeito de inserir *basemaps* é utilizado o MapBox, onde é possível customizar um *basemap* personalizado, bem como utilizar outros *basemaps* pré-existentes, incluindo imagens de satélite de alta resolução, etc.

Para melhor utilização, com a possiblidade de disponibilizar códigos, é necessário estudar a melhor maneira de ocultar a *API key*. Um início: - http://www.blacktechdiva.com/hide-api-keys/ - https://www.quora.com/How-do-you-hide-your-API-customer-key-token-when-youre-pushing-code-to-Github

[13]:
```
#folium.Map(location=[-23.9619271,-46.3427499],
#            tiles='Mapbox',
#            API_key='your.API.key',
#            zoom_start=12
#           )
```

Por fim, é possivel ainda inserir *basemaps* personalizados, disponibilizados em algum servidor.

[14]:
```
# Cria o mapa com servidores dos tiles
folium.Map(location=[-23.9619271,-46.3427499],
           zoom_start=12,
           tiles='http://{s}.tile.osm.org/{z}/{x}/{y}.png',
           attr='s'
          )
```

[14]: `<folium.folium.Map at 0x7fa8c2d064e0>`

[15]:
```
# %load '~/Documents/SourceCode/Codes/maps/create_tiles_folium.py'
def create_tiles_folium(tile_service=1, location=[-23.9619271, -46.3427499],␣
 ↪zoom_start=10):
    """
    Function to create map using tiles... a list of them
    - https://www.spatialbias.com/2018/02/qgis-3.0-xyz-tile-layers/
    - https://xyz.michelstuyts.be/
    - https://www.trailnotes.org/FetchMap/TileServeSource.html

    :param tile_service:
    :param location:
    :param zoom_start:
    :return:
    """
    # Import Packages
    import pandas as pd
    import folium

    # Read table with all tiles servers
    tiles_services = pd.read_csv('~/Documents/SourceCode/Codes/data/tiles.csv',␣
 ↪index_col=0)
    # print(tiles_services)
```

```
    # Create reference to attribution
    ref = ('<a href="' +
           tiles_services.loc[tile_service, 'attribution'] +
           '" target="blank">' +
           tiles_services.loc[tile_service, 'name'] +
           '</a>')

    return folium.Map(location=location,
                      zoom_start=zoom_start,
                      tiles=tiles_services.loc[tile_service, 'link'],
                      attr=ref)
```

[16]:
```
create_tiles_folium(1)
```

[16]:
```
<folium.folium.Map at 0x7fa8c2c9e438>
```

0.4.1 Outros elementos do *WebMap*

[17]:
```
# Adiciona legenda
folium.LayerControl().add_to(webmap)
webmap

# Adiciona a possibilidade de pontos, on-the-fly
webmap.add_child(folium.ClickForMarker(popup='Waypoint'))

# Adiciona a possibilidade de, a cada clique, descobrir as coordenadas
webmap.add_child(folium.LatLngPopup())
```

[17]:
```
<folium.folium.Map at 0x7fa8ec1b40f0>
```

[18]:
```
# Cria o mapa
webmap = folium.Map(
    location=[-23.9619271,-46.3427499],
    zoom_start=12
)

# Cálculo de Distâncias
from folium import plugins

# Adiciona ferramenta de medição
from folium.plugins import MeasureControl
webmap.add_child(MeasureControl())

# Fairly obvious I imagine - works best with transparent backgrounds
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/
 ↪AAEAAQAAAAAAAAlgAAAAJGE3OTA4YTdlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
```

```
FloatImage(url, bottom=5, left=85).add_to(webmap)

plugins.Fullscreen(
    position='topleft',
    title='Clique para Maximizar',
    title_cancel='Mininizar',
    force_separate_button=True).add_to(webmap)

webmap
```

[18]: <folium.folium.Map at 0x7fa8c2caf668>

0.5 Salvar o mapa em HTML

A grande vantagem é salvar o mapa como um arquivo *.html*, bastante possivel para dar um *embed* em qualquer página. Para salvar o resultado em um dado local, criei uma função que pode contribuir, avaliando se determinadas pastas estão criadas e, em caso negativo, cria as mesmas. Em uma destas pastas que ficará salvo o arquivo *.html* criado

[19]:
```
# %load '~/Documents/SourceCode/Codes/files/create_folders.py'
def create_folders(path, folders=['code', 'data', 'docs', 'imgs', 'maps']):
    """
    :param folders: Name os folders that you want create; E.g.: ['folder1',␣
 ↪'folder2']
    :return: Create directories if not exist
    """
    # Import Packages
    import os
    for folder in folders:
        directory=os.path.join(path, folder)
        try:
            if not os.path.exists(directory):
                os.makedirs(directory)
                print('Directory "', directory, '" created!', sep='')
            else:
                print('Directory "', directory, '" already exists...', sep='')
        except OSError:
            print('Error: Creating directory "', directory, '" fail.', sep='')
```

[20]:
```
create_folders("..")
```

```
Directory "../code" already exists...
Directory "../data" already exists...
Directory "../docs" already exists...
Directory "../imgs" already exists...
Directory "../maps" already exists...
```

[21]:
```
webmap.save('../maps/map.html')
```

O mapa em *.html*, que é possivel acessar por aqui, é apresentado a seguir: https://michelmetran.github.io/pages/folium/map/map.html

1 Teste

1.1 Exportando o *Juptyter Notebook* para outros formatos

Caso esse códigos sirvam para

[28]:
```
# %load '~/Documents/SourceCode/Codes/files/get_jupyternotebook_name.py'
def get_jupyternotebook_name():
    """
    Returns the name of the current notebook as a string
    From https://mail.scipy.org/pipermail/ipython-dev/2014-June/014096.html
    :return: Returns the name of the current notebook as a string
    """
    # Import Packages
    from IPython.core.display import Javascript
    from IPython.display import display

    display(Javascript('IPython.notebook.kernel.execute("theNotebook = " + \
    "\'"+IPython.notebook.notebook_name+"\'");'))

    # Result
    return theNotebook
```

[32]:
```
# %load '~/Documents/SourceCode/Codes/files/export_jupyter.py'
def export_jupyter(path, extensions=['html', 'markdown', 'latex', 'pdf']):
    """
    Export .ipynb file to others formats
    :return: File in other formats
    """
    # Import Packages
    import os
    import datetime

    # Data
    timestamp = datetime.datetime.now()
    srt_today = (str(timestamp.year) + '-' +
                 str(f"{timestamp.month:02d}") + '-' +
                 str(f"{timestamp.day:02d}"))

    # Extensions
    for extension in extensions:
        os.system('jupyter nbconvert --to {} {} --output {}'.
                  format(extension, get_jupyternotebook_name(),
                         os.path.join(path,␣
 ↪srt_today+'-'+get_jupyternotebook_name().split('.')[0])))
```

```
        print('Arquivo {} exportado corretamente para o formato .{}'.
              format(get_jupyternotebook_name(), extension))
```

[33]:
```
export_jupyter('../docs',['markdown', 'pdf'])
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


Arquivo folium.ipynb exportado corretamente para o formato .markdown

<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


Arquivo folium.ipynb exportado corretamente para o formato .pdf
```

1.2 Referências

- https://www.freecodecamp.org/news/real-world-data-science-project-traffic-accident-analysis-e5a36775ee11/
- Muita coisa interessante em https://www.youtube.com/watch?v=4RnU5qKTfYY
- https://jtemporal.com/folium/
- https://www.kaggle.com/rachan/how-to-folium-for-maps-heatmaps-time-analysis

[ ]:
```

```