

# IMD0905 - Data Science I

## Lesson #15 - Choropleth Maps

Ivanovitch Silva  
October, 2018



# Agenda

---

- Case study: IBGE, API Uber
- Geojson
- Importing files
- Creating maps
- Choropleths maps

# Update the repository

---

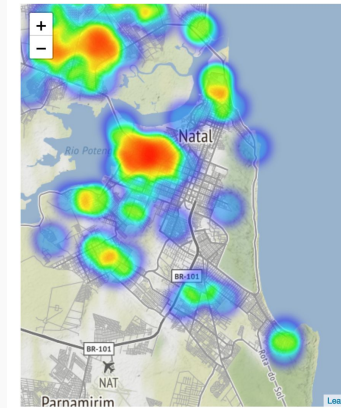
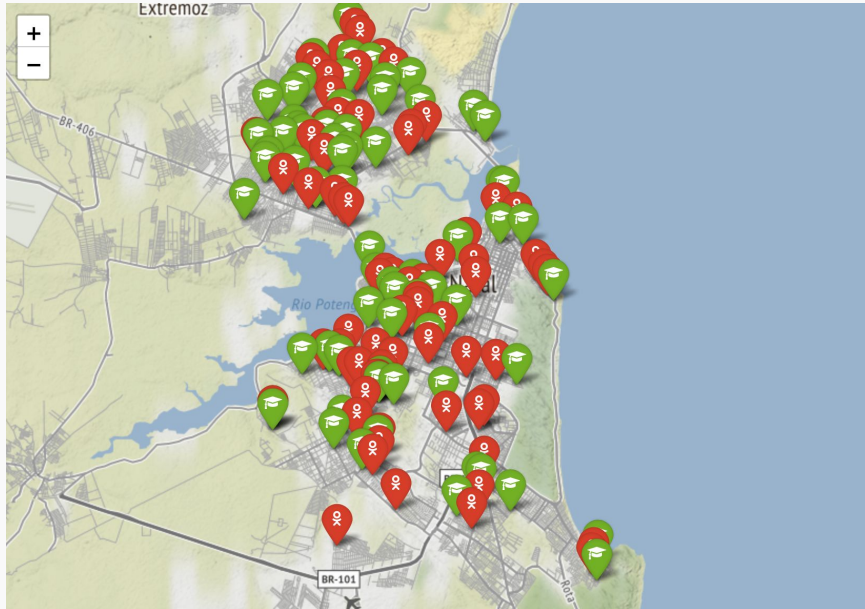
```
git clone https://github.com/ivanovitchm/IMD0905_datascience_one.git
```

Or ....

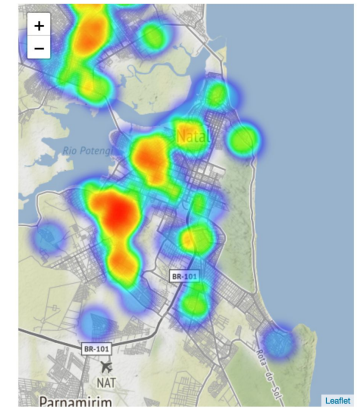
```
git pull
```

# Case study: open data Natal

---



Elementary School



Kindergarten

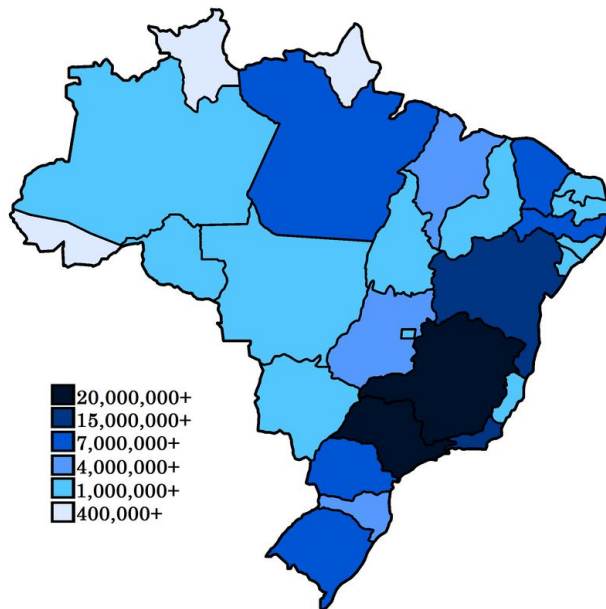
# References

---

1. <https://www.kaggle.com/daveianhickey/how-to-folium-for-maps-heatmaps-time-analysis>
2. <http://nbviewer.jupyter.org/github/python-visualization/folium/blob/master/examples/Plugins.ipynb>
3. <https://github.com/helmutd/datacamp/blob/master/Recreating%20John%20Snow's%20Ghost%20Map.ipynb>
4. <https://www.coursera.org/lecture/python-for-data-visualization/introduction-to-folium-CpjW0>
5. <https://pythonhow.com/web-mapping-with-python-and-folium/>

# Motivation

---



[https://en.wikipedia.org/wiki/Choropleth\\_map](https://en.wikipedia.org/wiki/Choropleth_map)



Instituto Brasileiro de Geografia e Estatística

[https://downloads.ibge.gov.br/downloads\\_estatisticas.htm](https://downloads.ibge.gov.br/downloads_estatisticas.htm)



<https://dadosabertos.camara.leg.br/>



# Introduction to dataset (IBGE)

Estimated population (2017)

	UF	COD._UF	COD._MUNIC	NOME_DO_MUNICÍPIO	POPULAÇÃO_ESTIMADA
<b>1075</b>	RN	24.0	109.0	Acari	11333.0
<b>1077</b>	RN	24.0	307.0	Afonso Bezerra	11211.0
<b>1079</b>	RN	24.0	505.0	Alexandria	13827.0
<b>1080</b>	RN	24.0	604.0	Almino Afonso	4854.0
<b>1081</b>	RN	24.0	703.0	Alto do Rodrigues	14365.0

# Geojson

---

GeoJSON is a format for encoding a variety of geographic data structures.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

<http://geojson.org/>

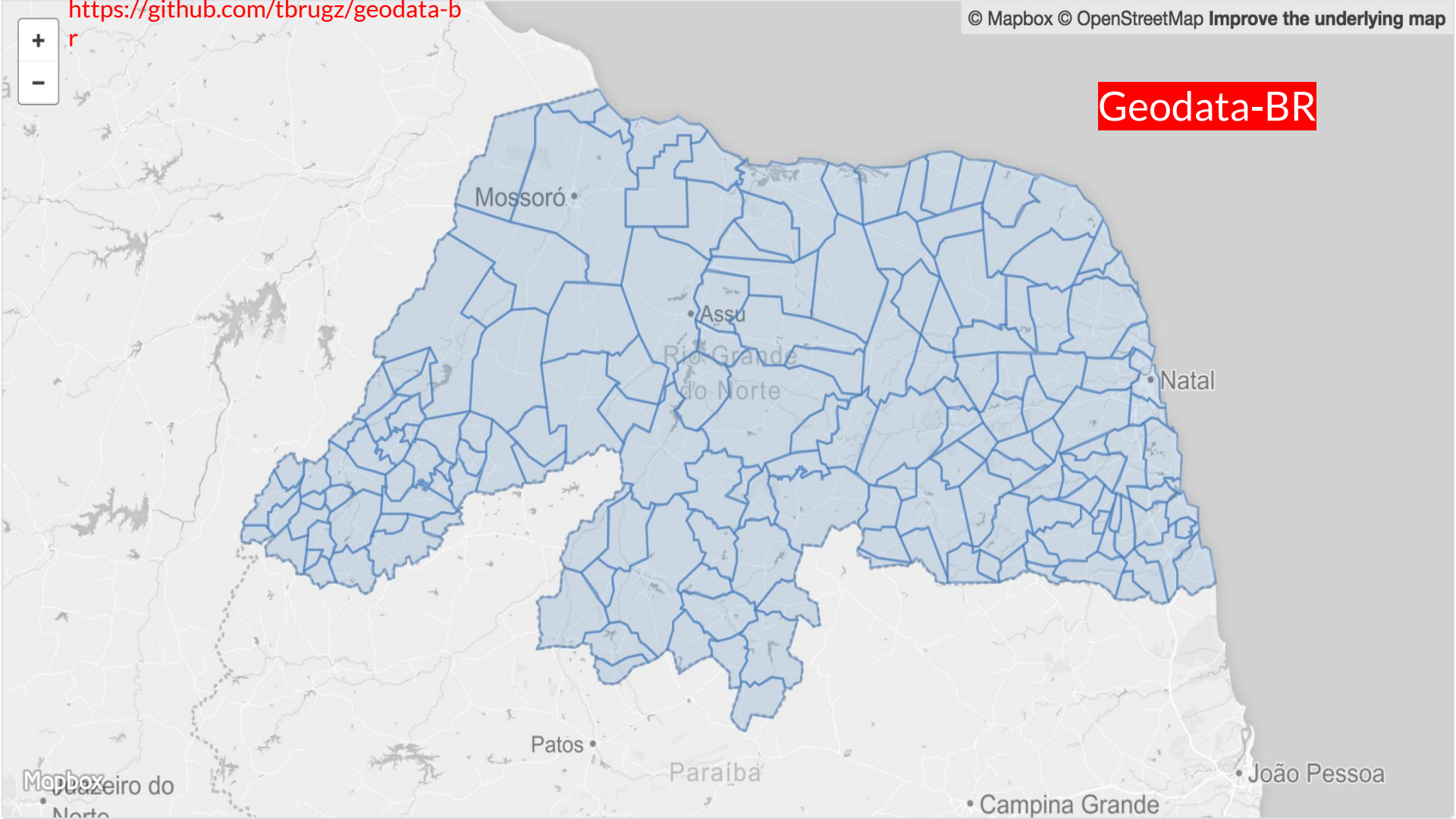
GeoJSON supports the following geometry types:

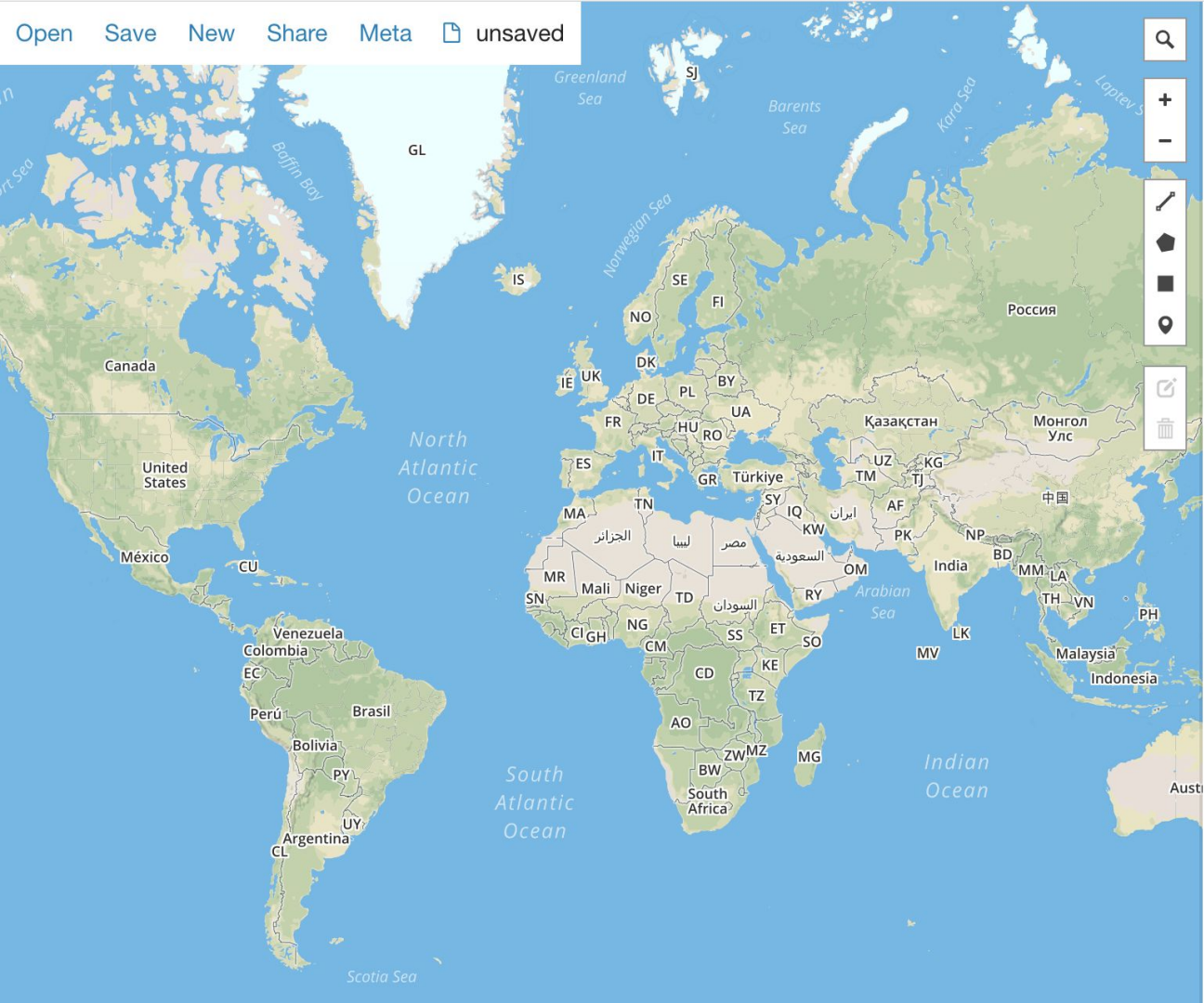
- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon





Geodata-BR





```
1 {  
2   "type": "FeatureCollection",  
3   "features": []  
4 }
```

# Importing geojson files

---

```
# searching the files in geojson/geojs-xx-mun.json  
br_states = os.path.join('geojson', 'geojs-24-mun.json')  
  
# load the data and use 'latin-1' encoding because the accent  
geo_json_data = json.load(open(br_states, encoding='latin-1'))
```

# Importing geojson files

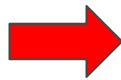
```
{'features': [{'geometry': {'coordinates': [[[-36.6752824479, -6.2695704427],  
[-36.6721661976, -6.2748710057],  
[-36.6621971359, -6.2781206182],  
[-36.6544080838, -6.2718175581],  
[-36.6302770363, -6.2681148661],
```

City

■ ■ ■

```
[-36.6892626008, -6.2741768473],  
[-36.68125826, -6.2694071238],  
[-36.6752824479, -6.2695704427]]],  
'type': 'Polygon'},  
'properties': {'description': 'Acari', 'id': '2400109', 'name': 'Acari'},  
'type': 'Feature'},  
{'geometry': {'coordinates': [[[-37.0150184398, -5.8704516715],  
[-37.0352362699, -5.8906742235],  
[-37.0354495717, -5.8906136983],
```

■ ■ ■



Coordinates: long, lat

# Cleaning & EDA

---

```
# http://cidades.ibge.gov.br/painel/historico.php?codmun=241030
# Presidente Juscelino city changes your name to Serra Caiada
geo_json_data['features'][112]['properties']['description'] = 'Serra Caiada'
geo_json_data['features'][112]['properties']['name'] = 'Serra Caiada'

cities = []
# list all cities in the state
for city in geo_json_data['features']:
    cities.append(city['properties']['description'])
cities
```



# EDA - Creating a map

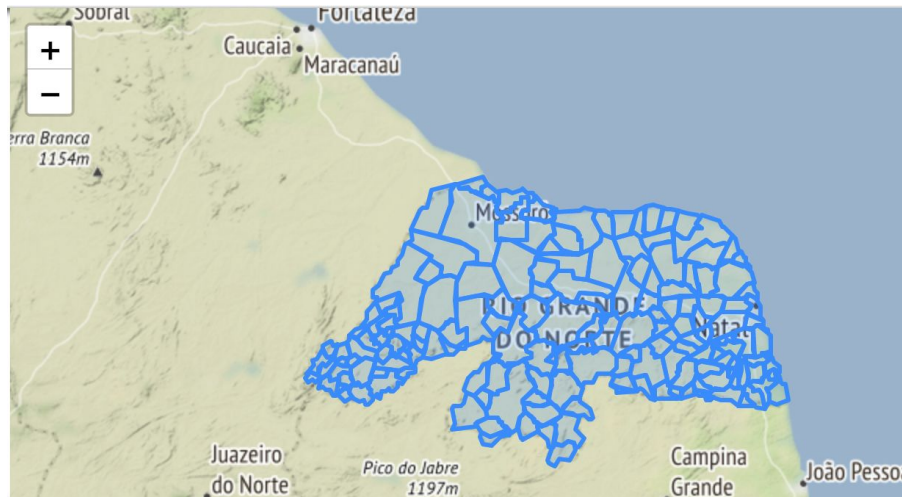
---

```
# Create a map object
```

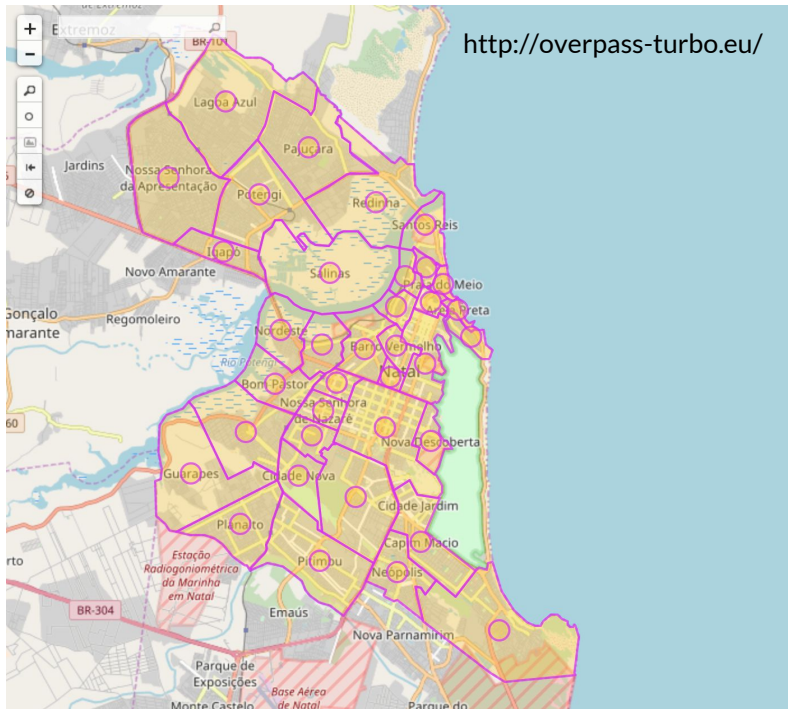
```
m = folium.Map(  
    location=[-5.826592, -35.212558],  
    zoom_start=7,  
    tiles='Stamen Terrain'  
)
```

```
# Configure geojson layer
```

```
folium.GeoJson(geo_json_data).add_to(m)
```



# Importing geojson files from other sources



A web based data mining tool for [OpenStreetMap](https://www.openstreetmap.org/) using Overpass API

<https://github.com/tyrasd/overpass-turbo>

```
[out:json][timeout:25];
{{geocodeArea:Natal RN Brasil}}->.searchArea;
(
  relation["admin_level"="10"](.searchArea);
);
out body;
>;
out skel qt;
```

# Case study: neighborhoods of Natal-RN

---

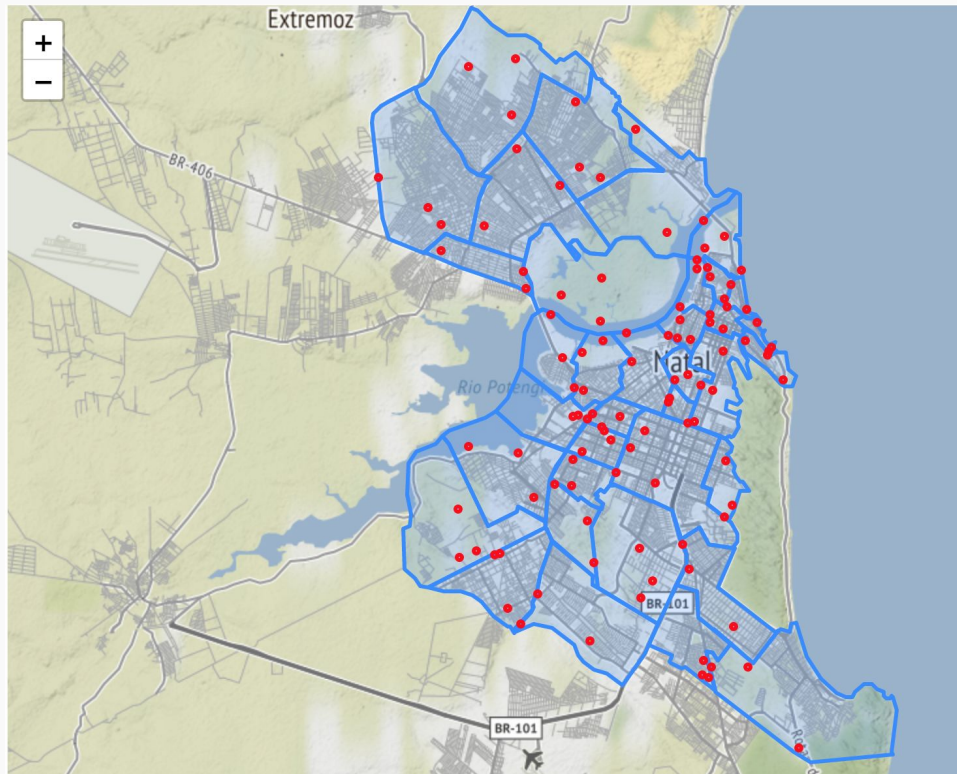
```
# import geojson file about natal neighborhood
natal_neigh = os.path.join('geojson', 'natal.geojson')

# load the data and use 'UTF-8' encoding
geo_json_natal = json.load(open(natal_neigh, encoding='UTF-8'))

neighborhood = []
# list all neighborhoods
for neigh in geo_json_natal['features']:
    neighborhood.append(neigh['properties']['name'])
neighborhood
```



# Neighborhoods as a polygon



Problem:  
spread  $X$  points using a  
uniform distribution within  
the limits of neighborhoods

# Spread X points within in polygon

---

```
from shapely.geometry import Polygon
from shapely.geometry import Point
```

```
# return a number of points inside the polygon
def generate_random(number, polygon, neighborhood):
    list_of_points = []
    minx, miny, maxx, maxy = polygon.bounds
    counter = 0
    while counter < number:
        x = random.uniform(minx, maxx)
        y = random.uniform(miny, maxy)
        pnt = Point(x, y)
        if polygon.contains(pnt):
            list_of_points.append([x,y,neighborhood])
            counter += 1
    return list_of_points
```

```
number_of_points = 3
```

```
# search all features
```

```
for feature in geo_json_natal['features']:
```

```
    # get the name of neighborhood
```

```
    neighborhood = feature['properties']['name']
```

```
    # take the coordinates (lat,log) of neighborhood
```

```
    geom = feature['geometry']['coordinates']
```

```
    # create a polygon using all coordinates
```

```
    polygon = Polygon(geom[0])
```

```
    # return number_of_points by neighborhood as a list [[log,lat],....]
```

```
    points = generate_random(number_of_points,polygon, neighborhood)
```

```
    # iterate over all points and print in the map
```

```
    for i,value in enumerate(points):
```

```
        log, lat, name = value
```

```
        # Draw a small circle
```

```
        folium.CircleMarker([lat,log],
```

```
                               radius=2,
```

```
                               popup='%s %s%d' % (name, '#', i),
```

```
                               color='red').add_to(m)
```

# Drawing a choropleth map (colormap)

```
# colormap yellow and green (YlGn)
colormap = linear.YlGn_03.scale(
    dataRN.POPULAÇÃO_ESTIMADA.min(),
    dataRN.POPULAÇÃO_ESTIMADA.max())
```

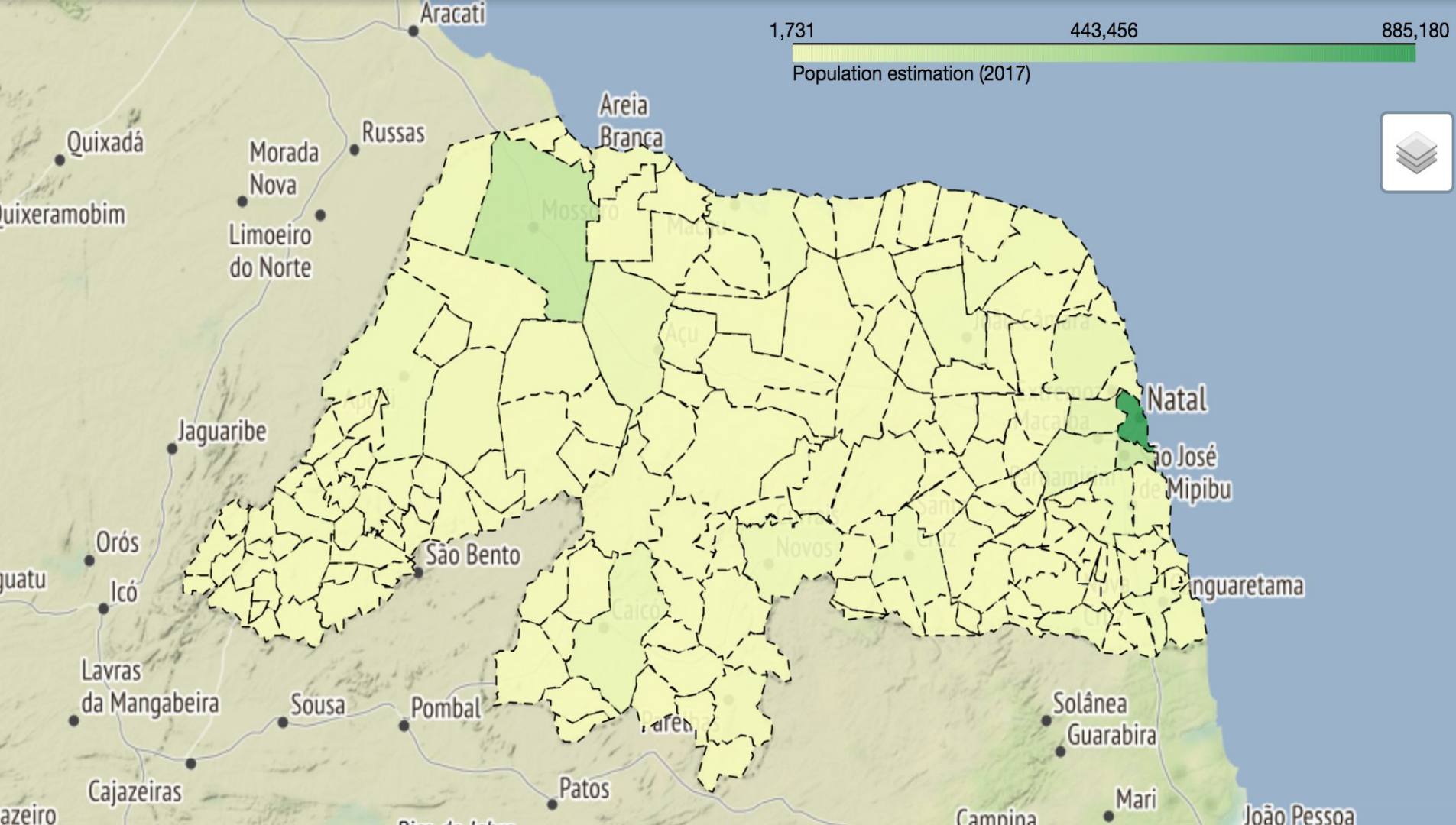
```
print(colormap(5000.0))
```

colormap



NOME_DO_MUNICÍPIO	POPULAÇÃO_ESTIMADA
Acari	11333.0
Afonso Bezerra	11211.0
Alexandria	13827.0
Almino Afonso	4854.0
Alto do Rodrigues	14365.0





# Drawing a choropleth map (option #1)

---

## Preparing the data

```
population_dict = dataRN.set_index('NOME_DO_MUNICÍPIO')['POPULAÇÃO_ESTIMADA']
```

# Drawing a choropleth map (option #1)

---

*# Configure geojson layer*

```
folium.GeoJson(  
    geo_json_data,  
    name='Population estimation of RN State in 2017',  
    style_function=lambda feature: {  
        'fillColor': colormap(population_dict[feature['properties']]['description']),  
        'color': 'black',  
        'weight': 1,  
        'dashArray': '5, 5',  
        'fillOpacity': 0.9,  
    }  
).add_to(m)  
  
colormap.caption = 'Population estimation (2017)'  
colormap.add_to(m)  
  
folium.LayerControl().add_to(m)
```

# Drawing a choropleth map (option #2)

---

```
# create a threshold of legend
```

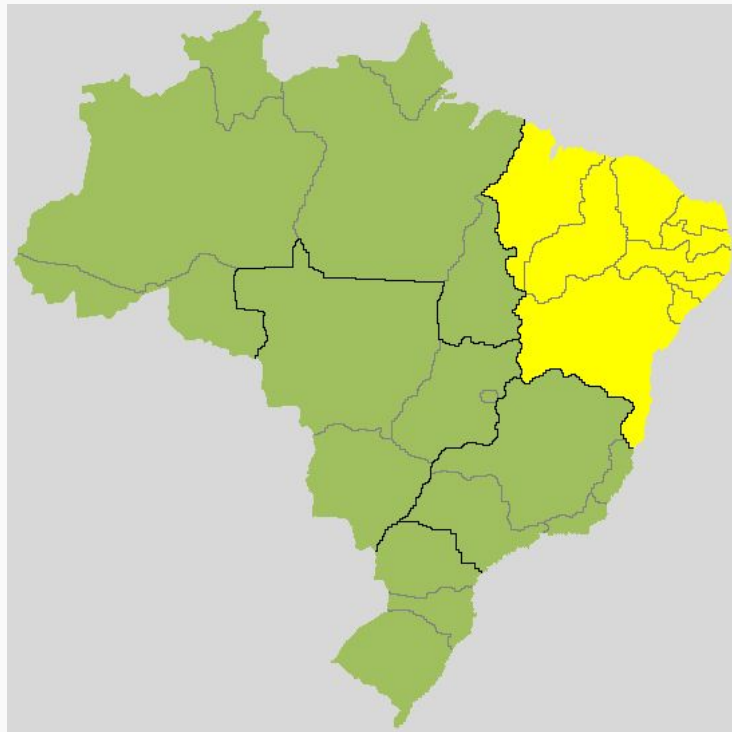
```
threshold_scale = np.linspace(dataRN[ 'POPULAÇÃO_ESTIMADA' ].min(),  
                              dataRN[ 'POPULAÇÃO_ESTIMADA' ].max(), 6, dtype=int).tolist()
```

```
m.choropleth(  
    geo_data=geo_json_data,  
    data=dataRN,  
    columns=[ 'NOME_DO_MUNICÍPIO', 'POPULAÇÃO_ESTIMADA' ],  
    key_on='feature.properties.description',  
    fill_color='YlGn',  
    legend_name='Population estimation (2017)',  
    highlight=True,  
    threshold_scale = threshold_scale  
)
```



# Exercise

---



1. Estimated population to Northeast Region
2. Other metrics



<https://dadosabertos.camara.leg.br/>