Data Science Foundation

Lesson #10 - Choropleth Maps

Ivanovitch Silva

October, 2017

Agenda

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

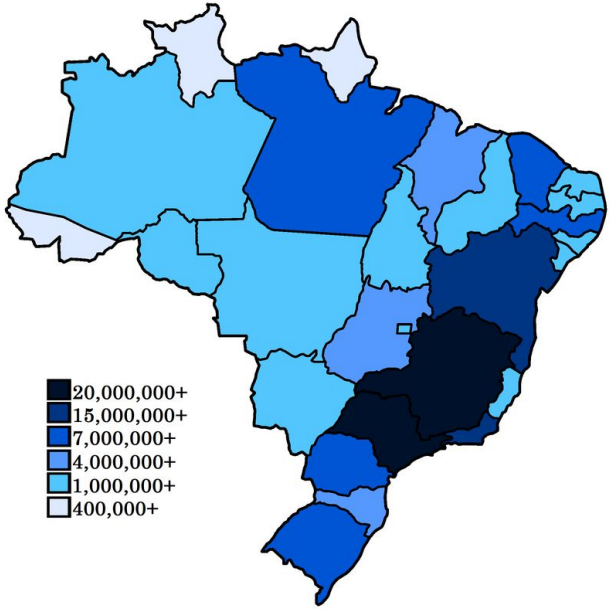
Update the repository

```
git clone https://github.com/ivanovitchm/EEC2006.git
```

Or ....

```
git pull
```

Motivation



https://en.wikipedia.org/wiki/Choropleth_map

IBGE
Instituto Brasileiro de Geografia e Estatística

https://downloads.ibge.gov.br/downloads_estatisticas.htm

UBER Developers

BRASIL
dados.gov.br
PORTAL BRASILEIRO DE DADOS ABERTOS

DADOS ABERTOS
BETA

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

Introduction to dataset (IBGE)

Estimated population

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

Geojson

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

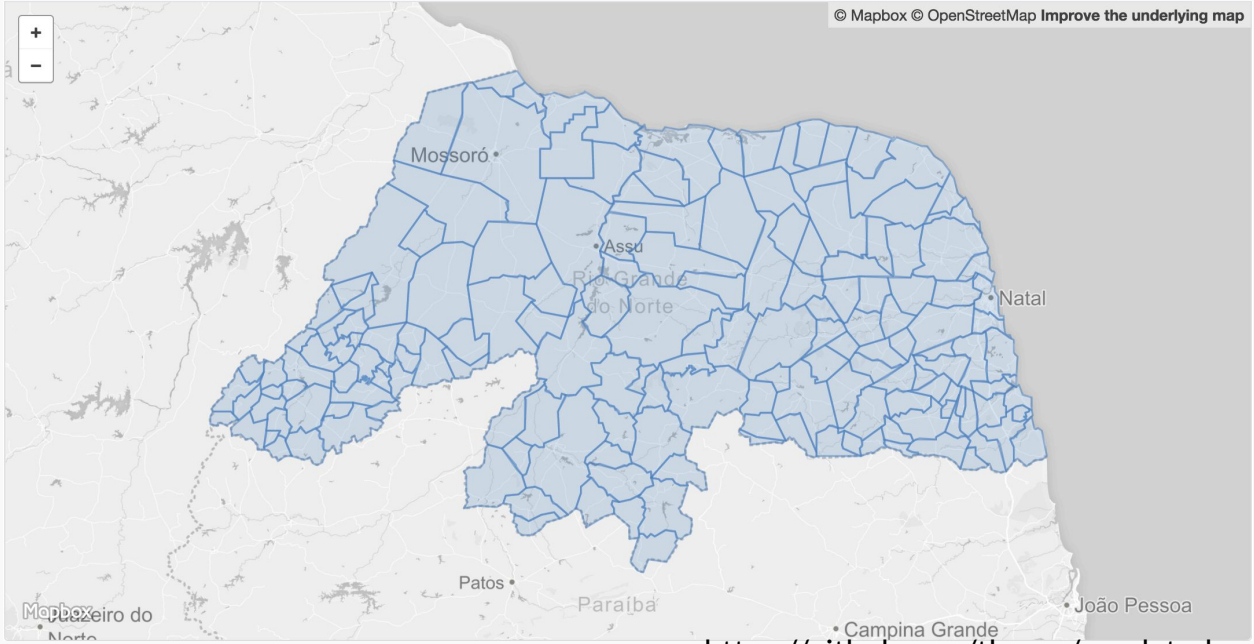
```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

http://geojson.org/

GeoJSON supports the following geometry types:

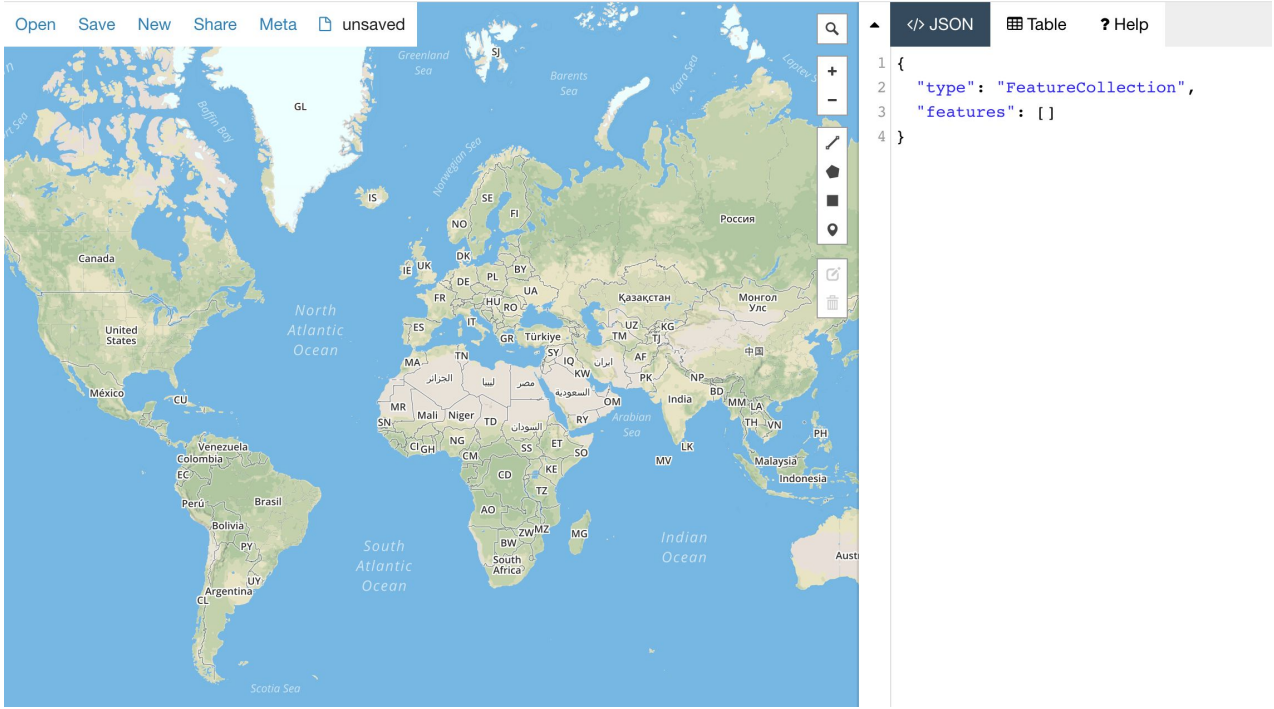
- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon

Geodata-BR



https://github.com/tbrugz/geodata-br

geojson.io


Open
Save
New
Share
Meta
unsaved
JSON
Table
Help
{
"type": "FeatureCollection",
"features": []
}
North Atlantic Ocean
South Atlantic Ocean
Indian Ocean
Greenland Sea
Barents Sea
Kara Sea
Norwegian Sea
Baffin Bay
Arabian Sea
Scotia Sea
Canada
United States
México
Venezuela
Colombia
Perú
Brasil
Bolivia
Argentina
Россия
Қазақстан
Монгол Улс
Türkiye
India
Malaysia
Indonesia
South Africa

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],
      [-36.625658466, -6.2854823428],
      [-36.6151351174, -6.292907263],
```

```
      [-36.68125826, -6.2694071238],
      [-36.6752824479, -6.2695704427]]],
    'type': 'Polygon'},
  'properties': {'description': 'Acari',
   'highlight': {'color': 'green',
    'dashArray': '5, 5',
    'fillColor': '#fdffca',
    'weight': 3},
   'id': '2400109',
   'name': 'Acari',
   'style': {'color': 'black',
    'dashArray': '5, 5',
    'fillColor': '#fdffca',
    'fillOpacity': 0.9,
    'weight': 1}},
```


Coordinates: long, lat

Cleaning

```
# 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'
```

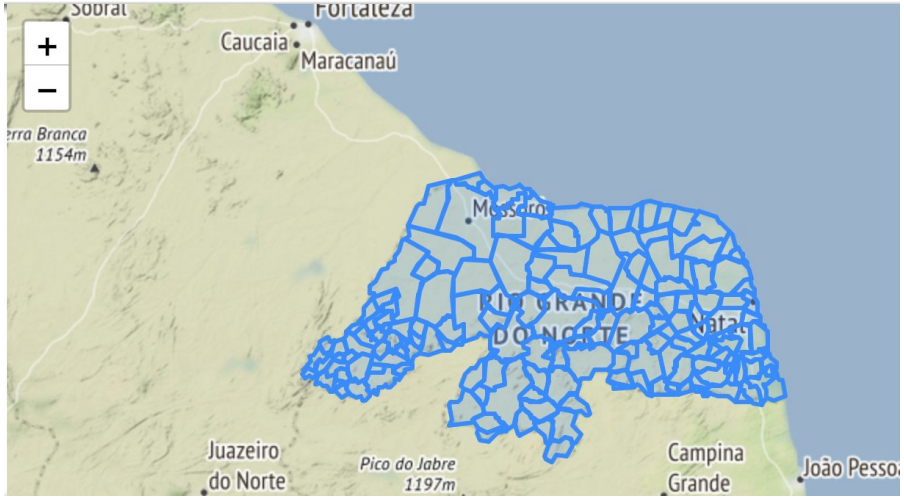
EDA - Listing all cities

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

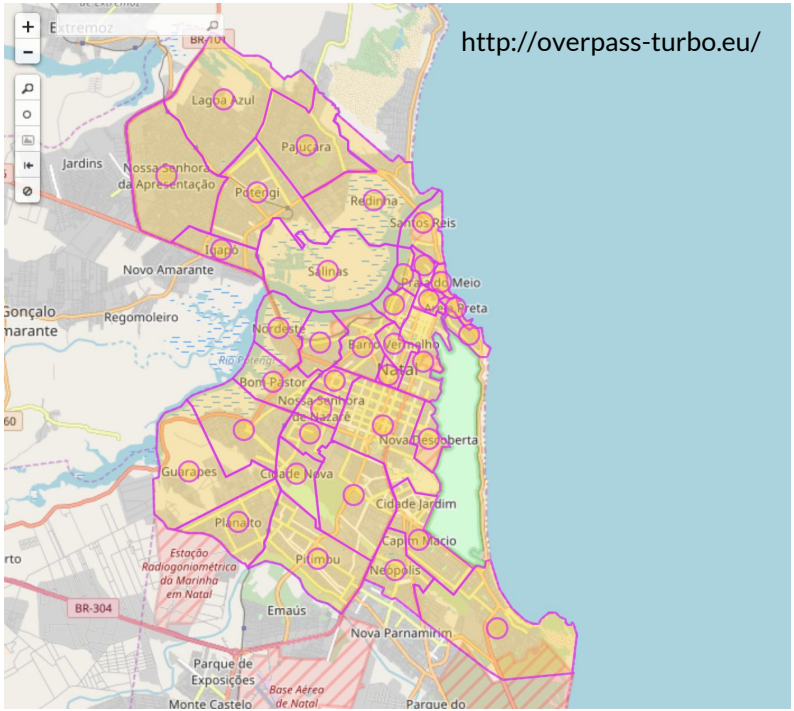
EDA - Creating a map

```
# 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://openstreetmap.org) using Overpass API

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

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

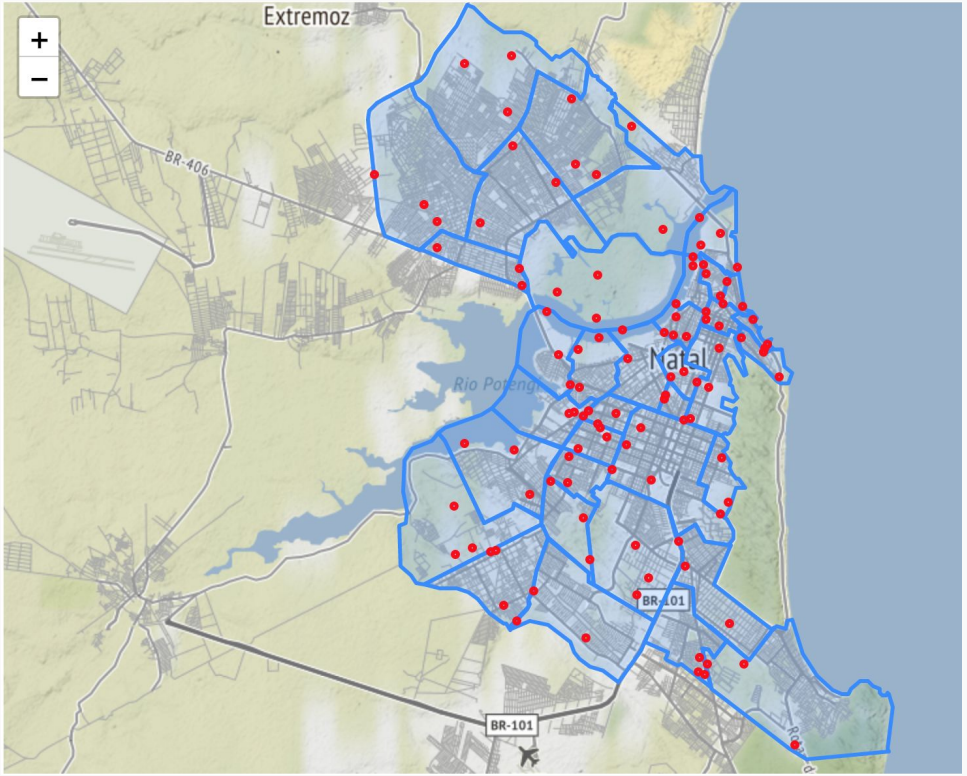
Case study: neighborhoods of Natal-RN

```
# 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)
m
```

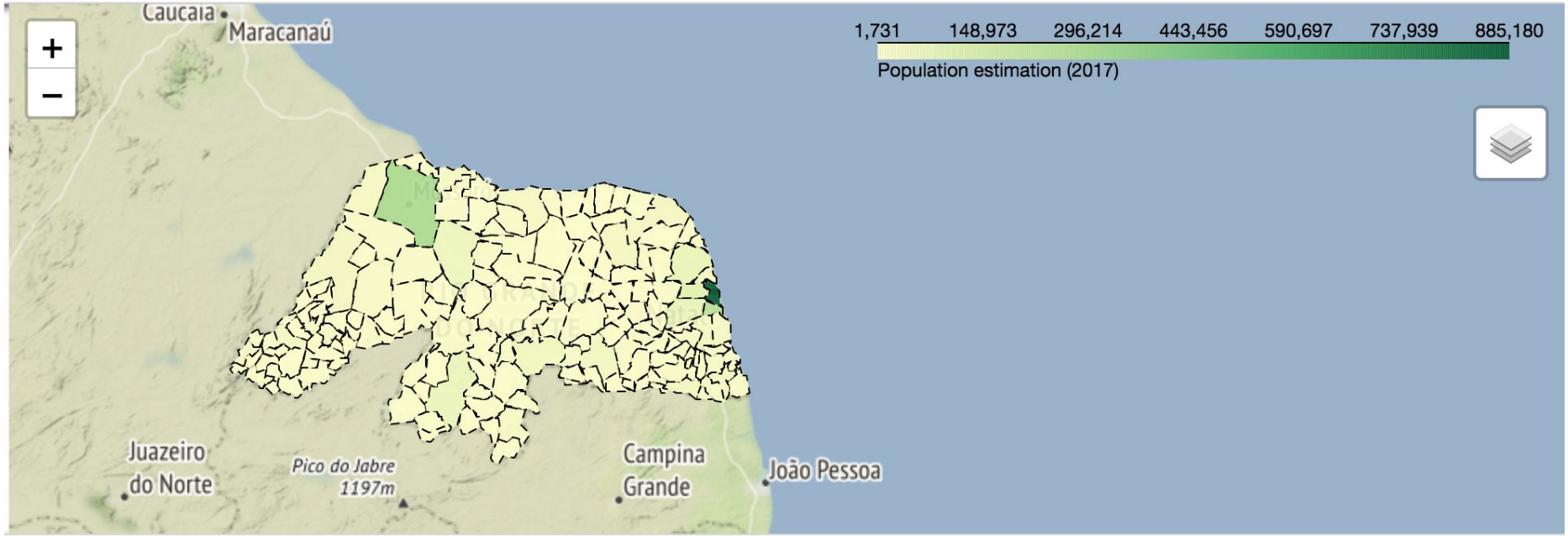
Drawing a choropleth map (colormap)

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

print(colormap(5000.0))
```

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

Drawing a choropleth map



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/

UBER.Developers

Products Solutions Showcase Docs Support News Team

SIGN IN

Moving

City Explorers

FOURSQUARE

UBER

https://developer.uber.com/

Explore the power of the API

↓

START BUILDING

Uber API

1. create user in https://developer.uber.com/
2. create an app in https://developer.uber.com/
3. install uber-rides package

```
!pip install uber-rides
```

Create an Uber session with a server token

```
from uber_rides.session import Session
from uber_rides.client import UberRidesClient

session = Session(server_token='your_token')
client = UberRidesClient(session)
```

Get a list of available products

```
response = client.get_products(-5.8323,-35.2054)

# API - get/products
products = response.json.get('products')
```

Get a list of available products

```
products[0]
```

```
{'capacity': 4,
 'cash_enabled': False,
 'description': 'THE LOW-COST UBER',
 'display_name': 'uberX',
 'image': 'http://d1a3f4spazzrp4.cloudfront.net/car-types/mono/mono-uberx.png',
 'price_details': {'base': 2.5,
  'cancellation_fee': 6.75,
  'cost_per_distance': 1.2,
  'cost_per_minute': 0.17,
  'currency_code': 'BRL',
  'distance_unit': 'km',
  'minimum': 6.75,
  'service_fees': [{'fee': 0.75, 'name': 'Booking fee'}]},
 'product_group': 'uberx',
 'product_id': '65cb1829-9761-40f8-acc6-92d700fe2924',
 'shared': False,
 'short_description': 'uberX',
 'upfront_fare_enabled': True}
```

Get price and times estimate

```
response = client.get_price_estimates(
    start_latitude=-5.8323,
    start_longitude=-35.2054,
    end_latitude= -5.8734,
    end_longitude=-35.1776,
    seat_count=2
)
```

Get price and times estimate

```
response.json
```


```
{'prices': [{'currency_code': 'BRL',
   'display_name': 'uberX',
   'distance': 5.02,
   'duration': 780,
   'estimate': 'R$14-18',
   'high_estimate': 18.0,
   'localized_display_name': 'uberX',
   'low_estimate': 14.0,
   'product_id': '65cb1829-9761-40f8-acc6-92d700fe2924'},
  {'currency_code': 'BRL',
   'display_name': 'UberSELECT',
   'distance': 5.02,
   'duration': 780,
   'estimate': 'R$16-21',
   'high_estimate': 21.0,
   'localized_display_name': 'UberSELECT',
   'low_estimate': 16.0,
   'product_id': 'bf8f99ca-f5f2-40d4-8ffc-52f1e2b17138'}]}
```

Get price and times estimate

```
wait_time = client.get_pickup_time_estimates(-5.8323,
                                              -35.2054,
                                             'bf8f99ca-f5f2-40d4-8ffc-52f1e2b17138')
```

```
wait_time.json
```

```
{'times': [{'display_name': 'UberSELECT',
   'estimate': 420,
   'localized_display_name': 'UberSELECT',
   'product_id': 'bf8f99ca-f5f2-40d4-8ffc-52f1e2b17138'}]}
```

```
wait_time.json.get('times')[0]['estimate']
```

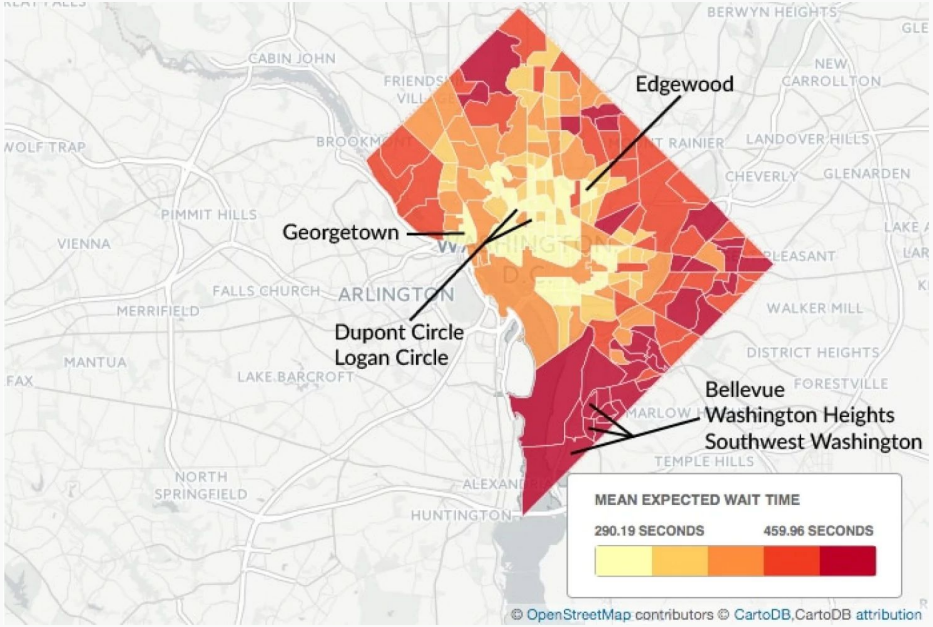
```
420
```

Rating limits

The Uber API enforces rate limits to help distribute resources among apps. Based on registered app's **server_token**, there are a limit to **2000 requests per hour**.

Exercise (motivation)

Uber seems to offer better service in areas with more white people. That raises some tough questions.

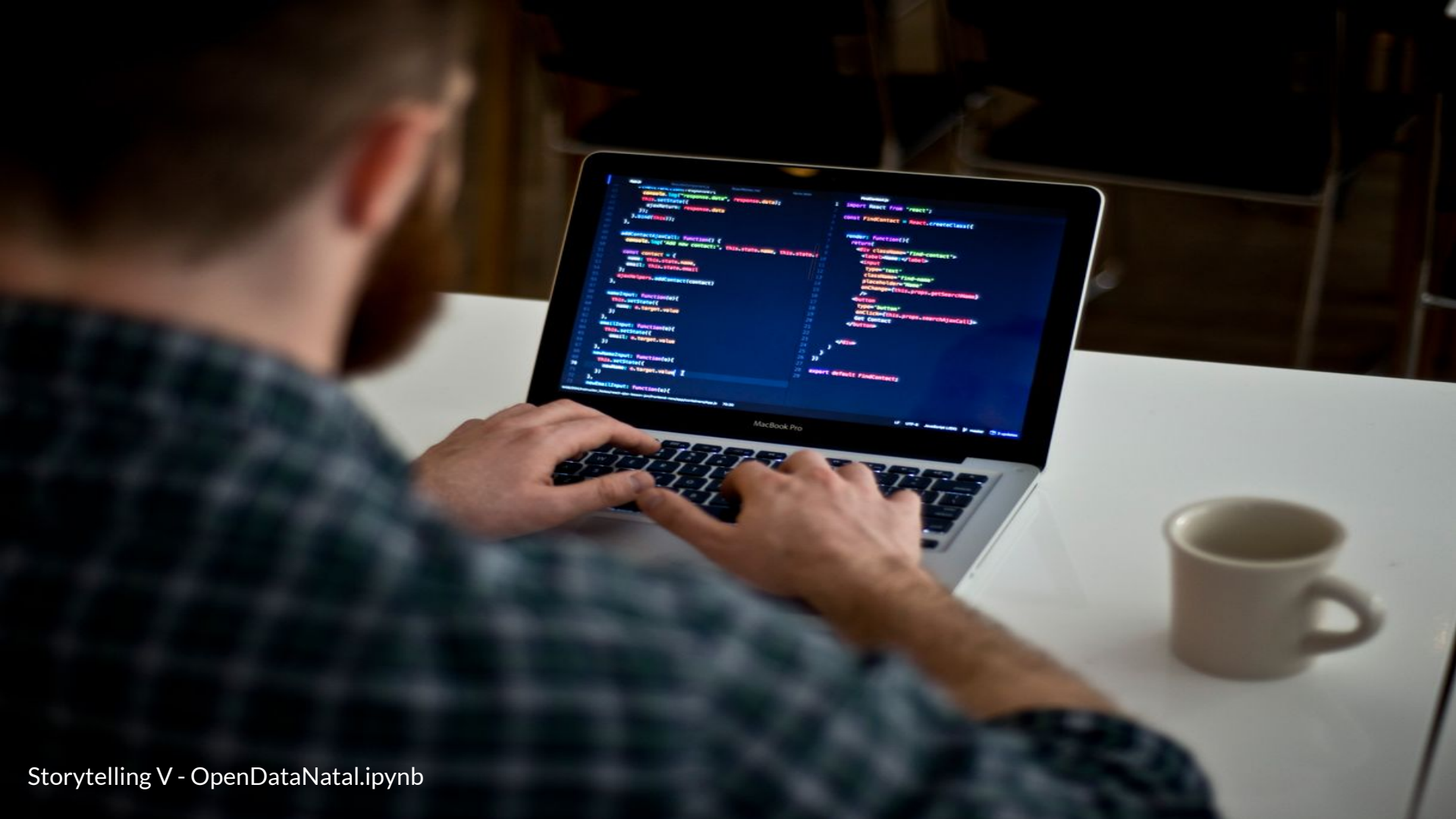


https://goo.gl/Jikb5h

Exercise

Interval: 3min; duration: 1 week

- query data about X uniform points for each neighborhood in Natal-RN.
- X must be selected according to rating limits of Uber API.
- compute the ETA average for all neighborhoods.
- use a choropleth map to generate an aesthetic plot.



Storytelling V - OpenDataNatal.ipynb