

# Programação II

+

# Estruturas de Dados para Bioinformática

Hugo Pacheco

DCC/FCUP  
23/24

# Mapas

(geopandas + contextily +  
matplotlib)

# Mapas

- Últimas aulas:
  - Gráficos estáticos e dinâmicos em *matplotlib*
- Esta aula:
  - Mapas em *geopandas* + *contextily* + *matplotlib*
    - Desenhar conjuntos de dados com coordenadas geográficas associadas
    - Desenhar (projeções do) mapa Mundo
    - Combinar gráficos *matplotlib* “normais” com mapas
  - Agora não vamos ver: outras bibliotecas para mapas com mais funcionalidades (*basemap*, *plotly*, *GeoPy*, *CartoPy*, *Folium*, ...)

# GeoJSON

- Formato de dados textual para representar dados geográficos
  - Sub-formato de JSON (qualquer ficheiro *geojson* é um ficheiro *json* válido) (relembrar Projeto 2)
  - Suporta tipos geométricos (e.g., pontos, linhas, polígonos) com coordenadas GPS (*latitude,longitude*)
  - Pode ser visualizado facilmente
    - E.g., mapa das regiões de Portugal
    - Podem experimentar visualizar qualquer ficheiro *geojson* em [geojson.io](#)

# GeoPandas

- Biblioteca de manipulação de dados geográficos construída sobre o *Pandas*
- Introduz um novo tipo, o *GeoDataFrame*
  - Funciona tal e qual como um *DataFrame*
  - Tem uma coluna adicional *geometry*, com dados geográficos
  - Nota: os tipos geométricos são os do formato *GeoJSON*

	CCDR		geometry
0	Alentejo	MULTIPOLYGON	(((-8.18725 37.34111, -8.18728 37...
1	Algarve	MULTIPOLYGON	(((-7.88042 36.97347, -7.88042 36...
2	Açores	MULTIPOLYGON	(((-25.01910 36.94788, -25.01908 ...
3	Centro	MULTIPOLYGON	(((-8.80879 39.48166, -8.80883 39...
4	Madeira	MULTIPOLYGON	(((-16.05836 30.03023, -16.05835 ...
5	Norte	MULTIPOLYGON	(((-8.31622 40.78681, -8.31637 40...
6	RLVT	MULTIPOLYGON	(((-8.76458 38.50812, -8.76458 38...

# GeoDataFrame ≈ GeoJSON

- Podemos facilmente ler um ficheiro *geojson* com o *geopandas*
- E.g., mapa das regiões de Portugal disponível [aqui](#)

```
import geopandas as gpd
df_map = gpd.read_file("portugal.geojson")
print(df_map)
```

	CCDR		geometry
0	Alentejo	MULTIPOLYGON	(((-8.18725 37.34111, -8.18728 37...
1	Algarve	MULTIPOLYGON	(((-7.88042 36.97347, -7.88042 36...
2	Açores	MULTIPOLYGON	(((-25.01910 36.94788, -25.01908 ...
3	Centro	MULTIPOLYGON	(((-8.80879 39.48166, -8.80883 39...
4	Madeira	MULTIPOLYGON	(((-16.05836 30.03023, -16.05835 ...
5	Norte	MULTIPOLYGON	(((-8.31622 40.78681, -8.31637 40...
6	RLVT	MULTIPOLYGON	(((-8.76458 38.50812, -8.76458 38...

# GeoDataFrame ≈ GeoJSON

- Podemos facilmente criar um ficheiro *geojson* com o *geopandas*
- E.g., o centro de cada região de Portugal

```
# calcular o centro de cada polígono
df_map['geometry'] = df_map['geometry'].centroid
print(df_map)
      CCDR           geometry
0  Alentejo  POINT (-7.91979 38.39159)
1   Algarve  POINT (-8.13324 37.24494)
2    Açores  POINT (-27.28376 38.34802)
...
      ...
# escrever para ficheiro
df_map.to_file("centros.geojson", driver='GeoJSON')
```

# GeoPandas (operações geométricas)

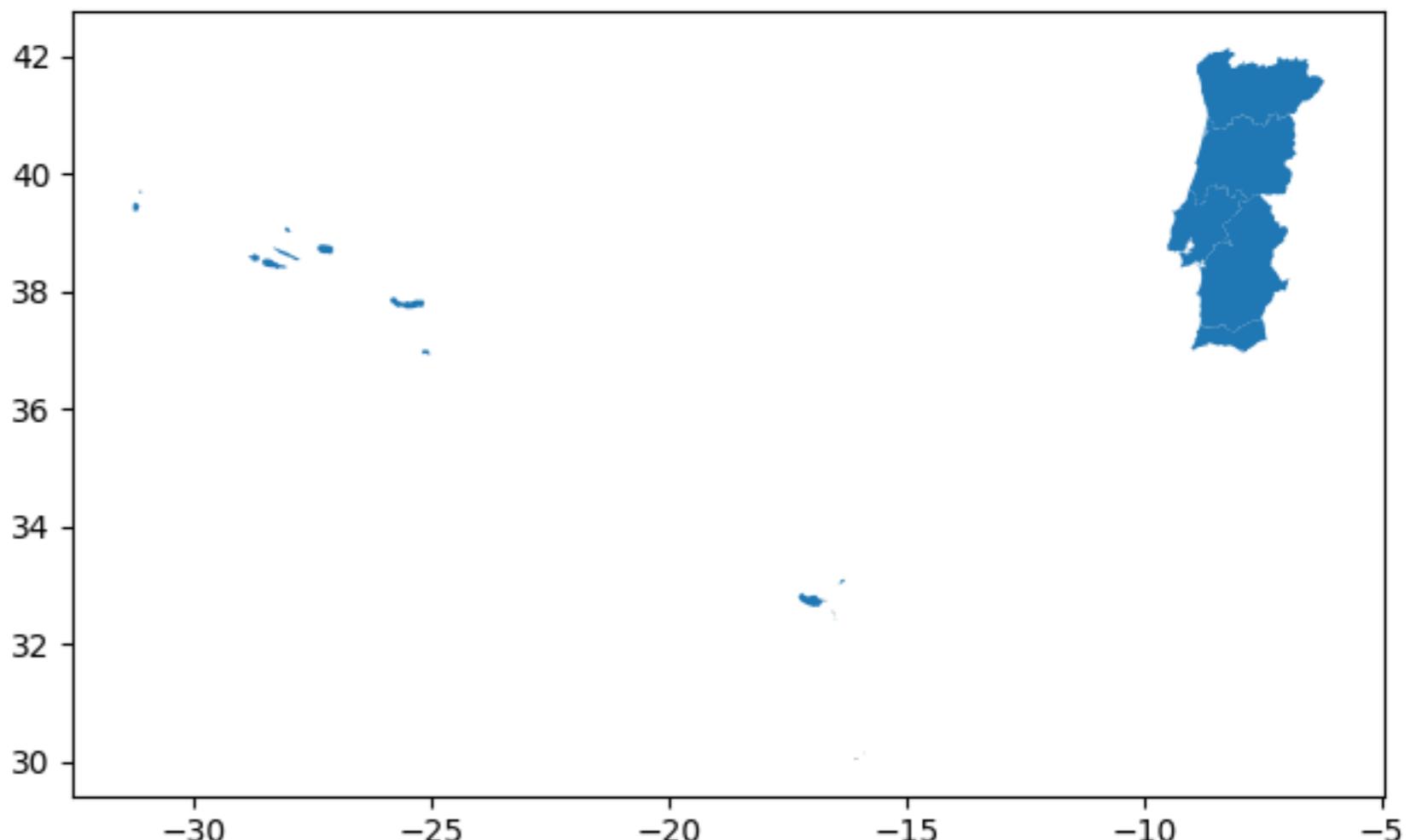
- Calcular o centro: *centroid*
  - Calcular limites exteriores: *boundary*

- E outros... utiliza o *Shapely* ([documentação completa](#))

# GeoPandas (plot)

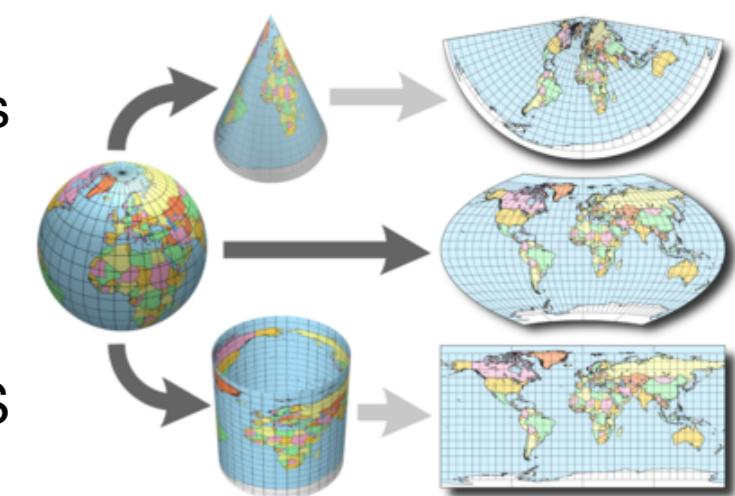
- Podemos desenhar um *GeoDataFrame* com o *matplotlib*
- X = longitude, Y = latitude

```
import  
matplotlib.pyplot  
as plt  
  
df_map.plot()  
plt.show()
```



# Contextily (mapas)

- Podemos desenhar facilmente um mapa de background com o *contextily*
- **Importante:** garantir que o CRS dos dados e do mapa são iguais
  - CRS = Coordinate Reference System
  - Pode haver alguma distorção de perspetiva consoante o CRS

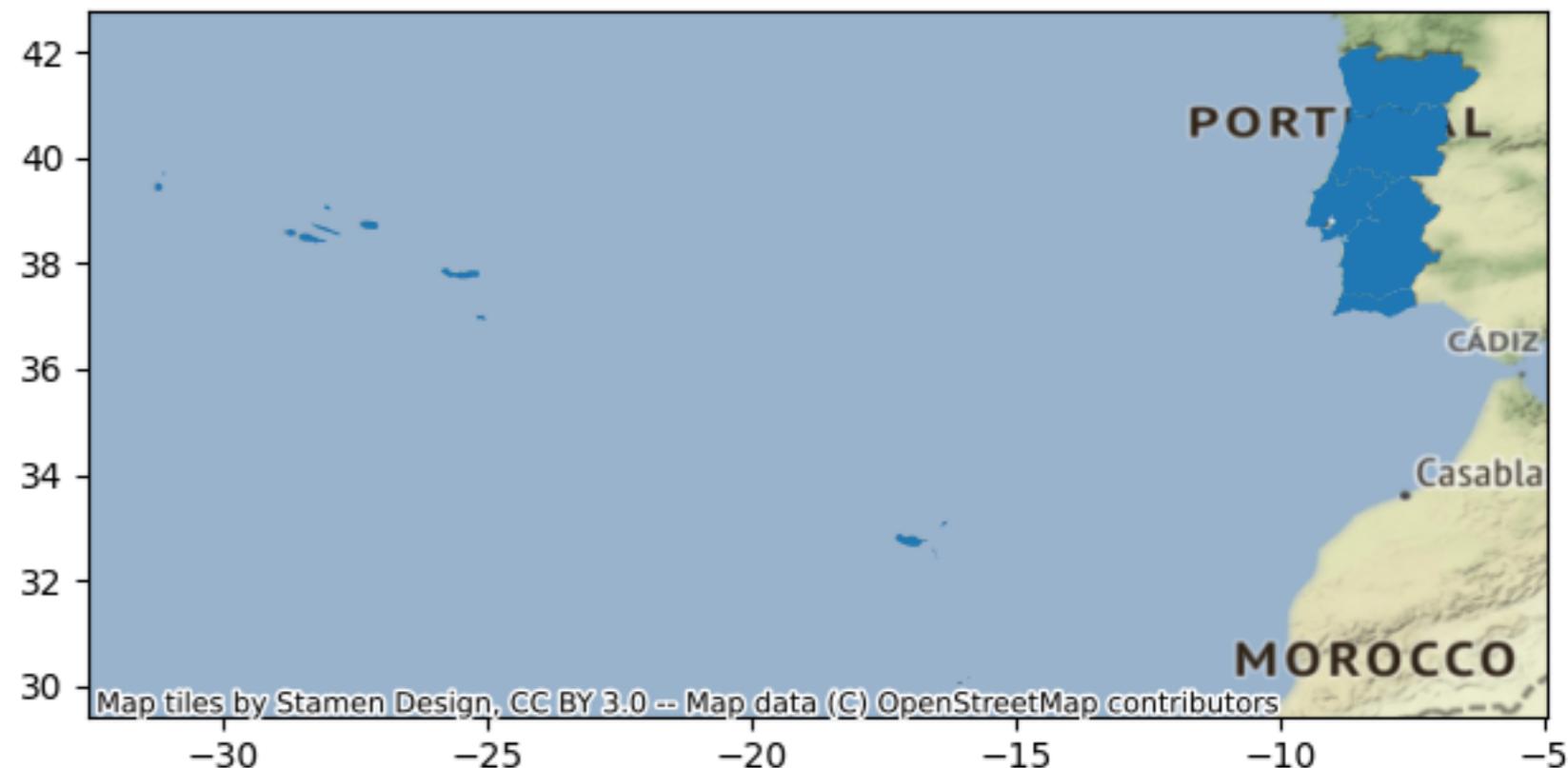


```
import contextily as ctx

# desenha as regioes
ax = df_map.plot()

# acrescenta o mapa
ctx.add_basemap(ax, zoom=5
, crs=df_map.crs)

plt.show()
```



# Contextily (zoom)

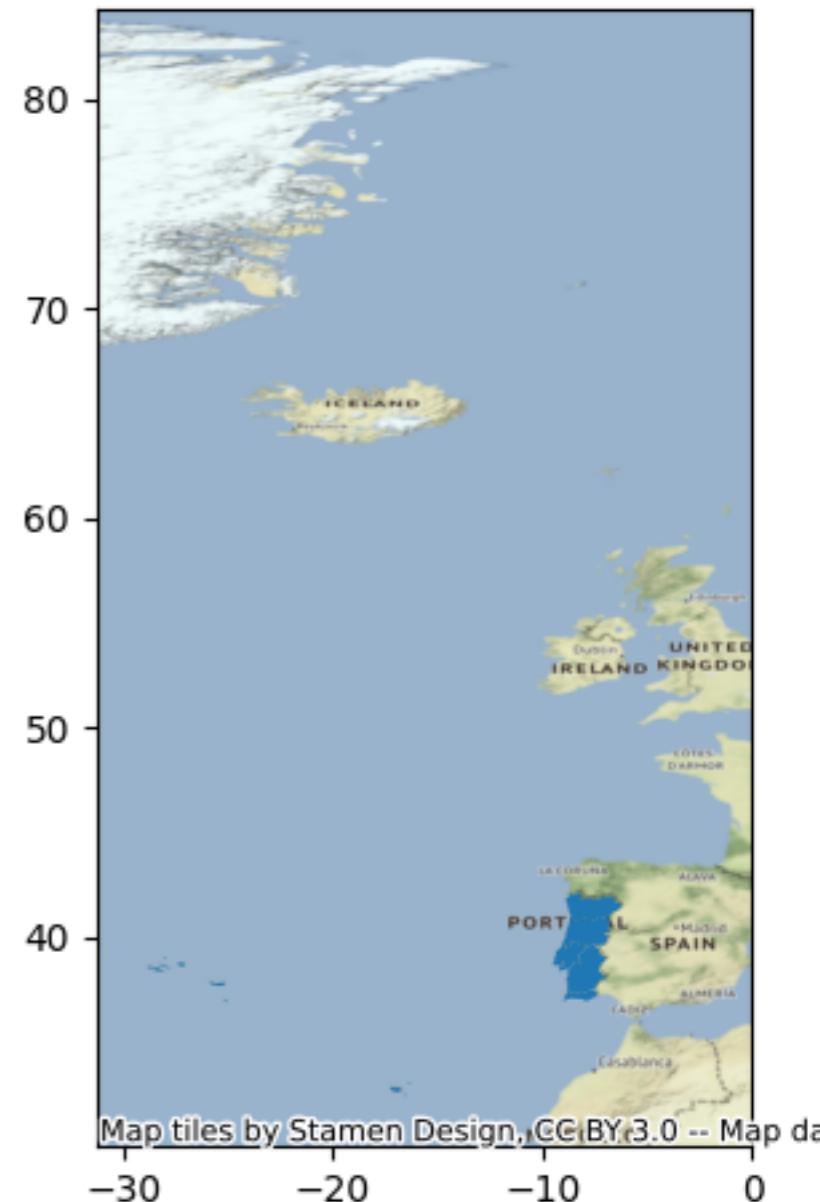
- Podemos alterar o zoom do mapa, controlando os eixos do gráfico
- **Relembrar:** é um gráfico matplotlib normal

```
import contextily as ctx

# desenha as regioes
ax = df_map.plot()
# altera os limites
minx, miny, maxx, maxy = df_map.total_bounds
ax.set_xlim(minx, 0)
ax.set_ylim(miny, maxy*2)

# acrescenta o mapa
ctx.add_basemap(ax, zoom=5, crs=df_map.crs)

plt.show()
```



# *Contextily* (mapas)

- Podemos utilizar diferentes mapas
- Listagem dos mapas disponíveis:

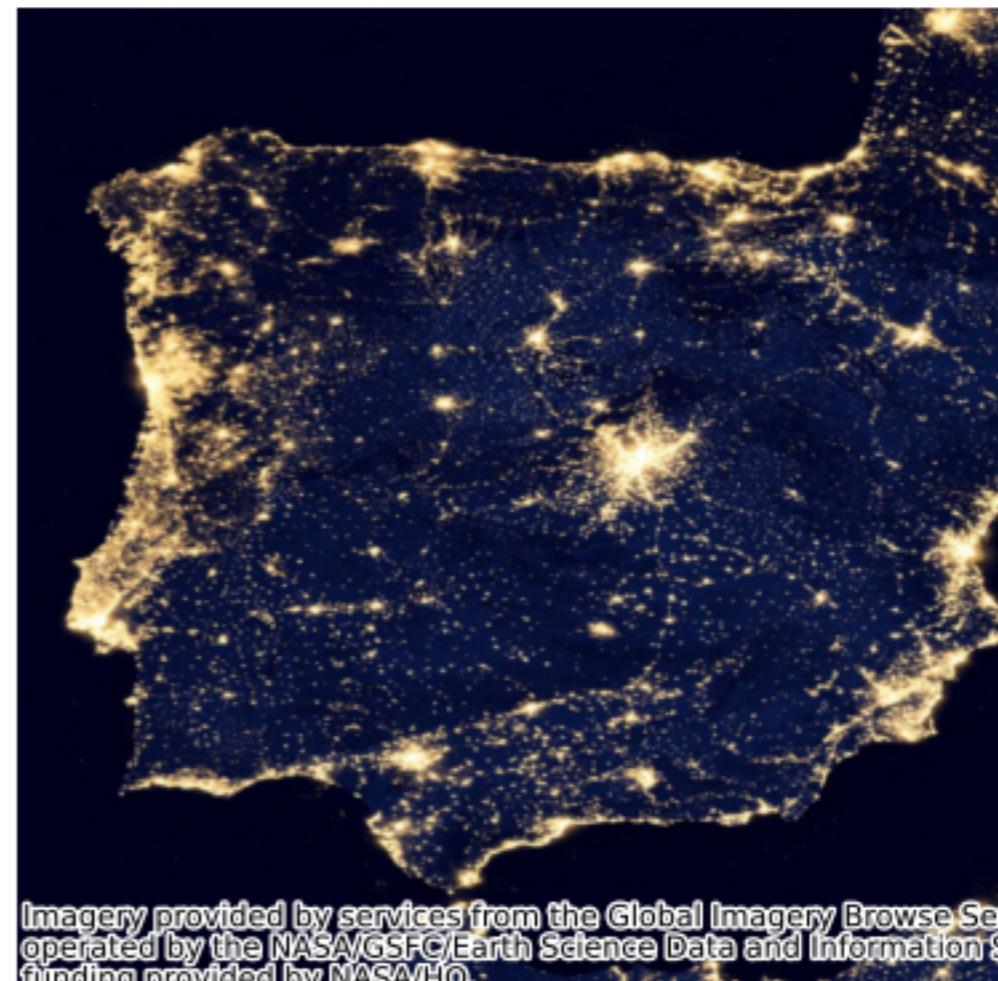
```
>>> ctx.providers.keys()
dict_keys(['OpenStreetMap', 'OpenSeaMap', 'OpenPtMap',
'OpenTopoMap', 'OpenRailwayMap', 'OpenFireMap',
'SafeCast', 'Thunderforest', 'OpenMapSurfer', 'Hydda',
'MapBox', 'Stamen', 'Esri', 'OpenWeatherMap', 'HERE',
'FreeMapSK', 'MtbMap', 'CartoDB', 'HikeBike', 'BasemapAT',
'nImaps', 'NASAGIBS', 'NLS', 'JusticeMap', 'Wikimedia',
'GeoportailFrance', 'OneMapSG'])
>>> ctx.providers['CartoDB'].keys()
dict_keys(['Positron', 'PositronNoLabels',
'PositronOnlyLabels', 'DarkMatter', 'DarkMatterNoLabels',
'DarkMatterOnlyLabels', 'Voyager', 'VoyagerNoLabels',
'VoyagerOnlyLabels', 'VoyagerLabelsUnder'])
```

# Contextily (mapas)

- Podemos utilizar diferentes mapas

```
# península ibérica
fig,ax = plt.subplots()
ax.set_xlim(-10,0)
ax.set_ylim(35,45)
ax.axis('off')

night =
ctx.providers.NASAGIBS.ViirsEarthAtNight2012
ctx.add_basemap(ax,crs=4326,source=night)
plt.show()
```

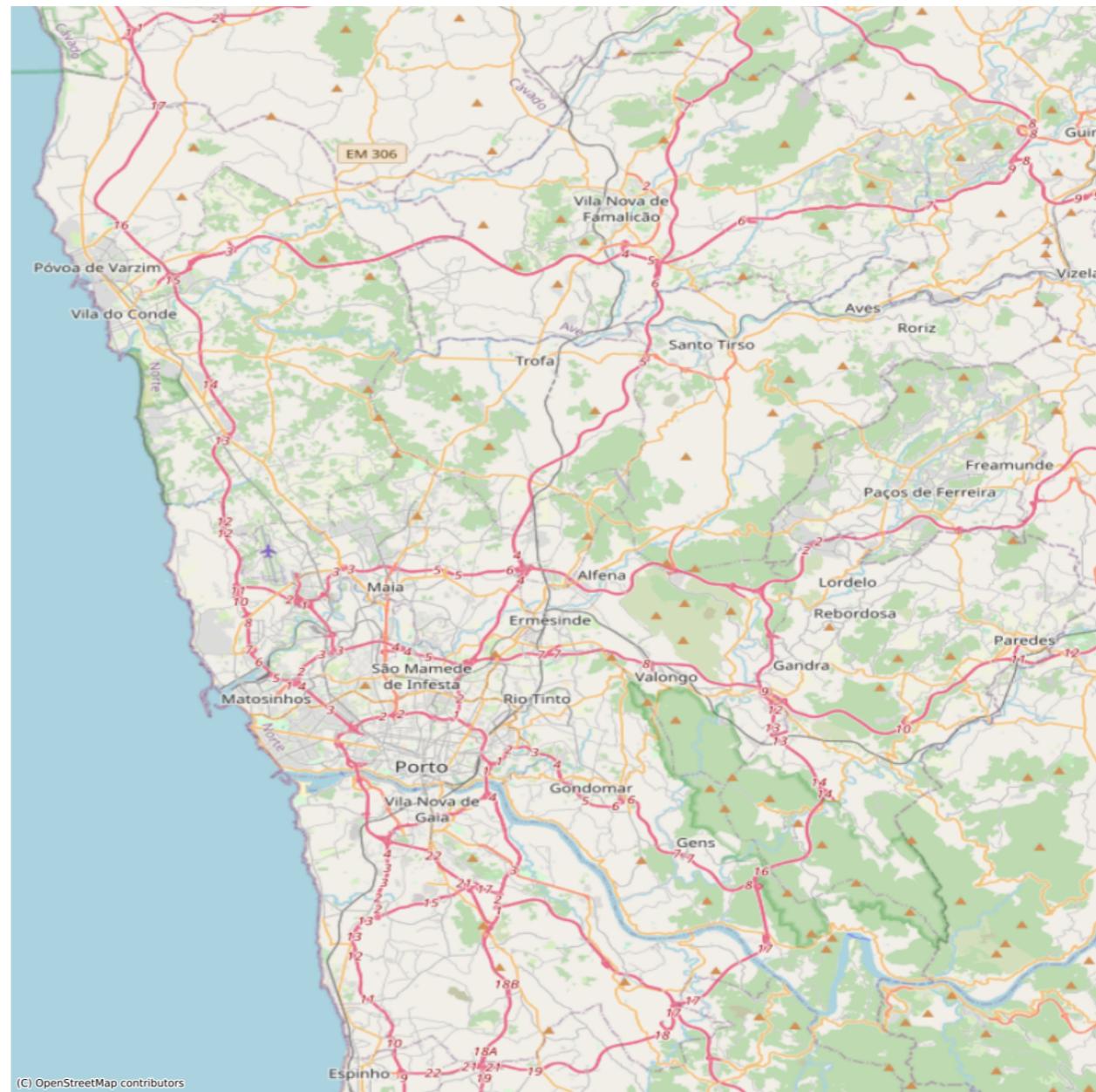


# Contextily (mapas)

- Podemos utilizar diferentes mapas

```
# Porto
fig,ax =
plt.subplots(figsize=(15,15))
ax.set_xlim(-8.8,-8.3)
ax.set_ylim(41,41.5)
ax.axis('off')

mapnik =
ctx.providers.OpenStreetMap.Mapnik
ctx.add_basemap(ax,crs=4326,source=
mapnik)
plt.show()
```

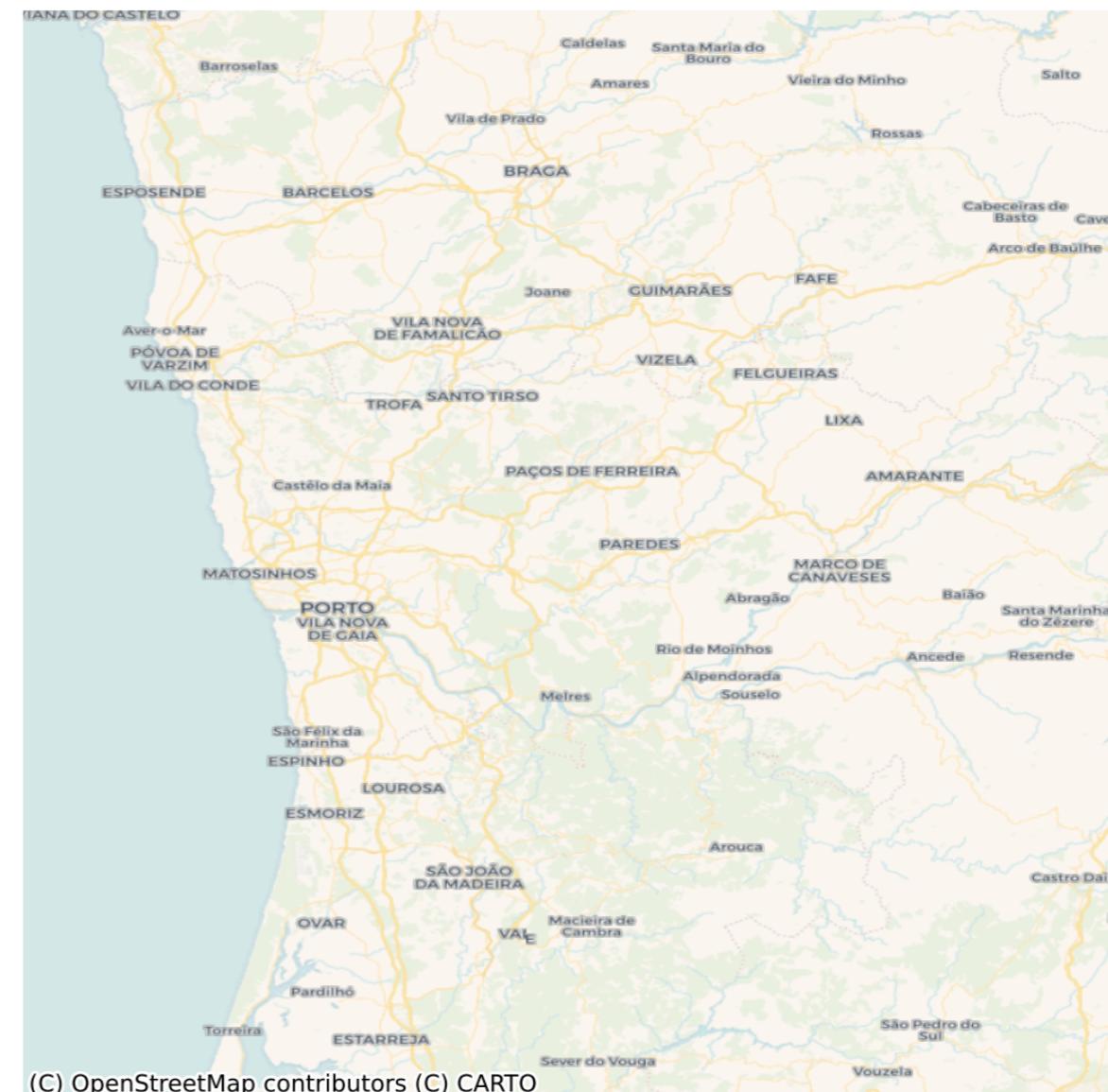


# Contextily (mapas)

- Podemos utilizar diferentes mapas (em simultâneo)

```
# Porto alargado
fig,ax = plt.subplots(figsize=(10,10))
ax.set_xlim(-8.9,-7.9)
ax.set_ylim(40.7,41.7)
ax.axis('off')

# múltiplos mapas
colors =
ctx.providers.CartoDB.VoyagerNoLabels
labels =
ctx.providers.CartoDB.PositronOnlyLabels
ctx.add_basemap(ax,crs=4326,source=colors)
ctx.add_basemap(ax,crs=4326,source=labels)
```



# GeoPandas (criação)

- Podemos criar um novo *GeoDataFrame* a partir de um *DataFrame*, acrescentando uma coluna *geometry*
- E.g., lista de cidades de Portugal disponível [aqui](#) em formato CSV
- Exportar para ficheiro GeoJSON

```
import geopandas as gpd
import pandas as pd

data = pd.read_csv('pt.csv')
df = data[['city', 'population']].copy()
df.fillna(1000, inplace=True)
# cria um ponto por coordenada
gdf =
gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(data
['lng'], data['lat']))
# escrever para ficheiro
gdf.to_file("cidades.geojson", driver='GeoJSON')
```

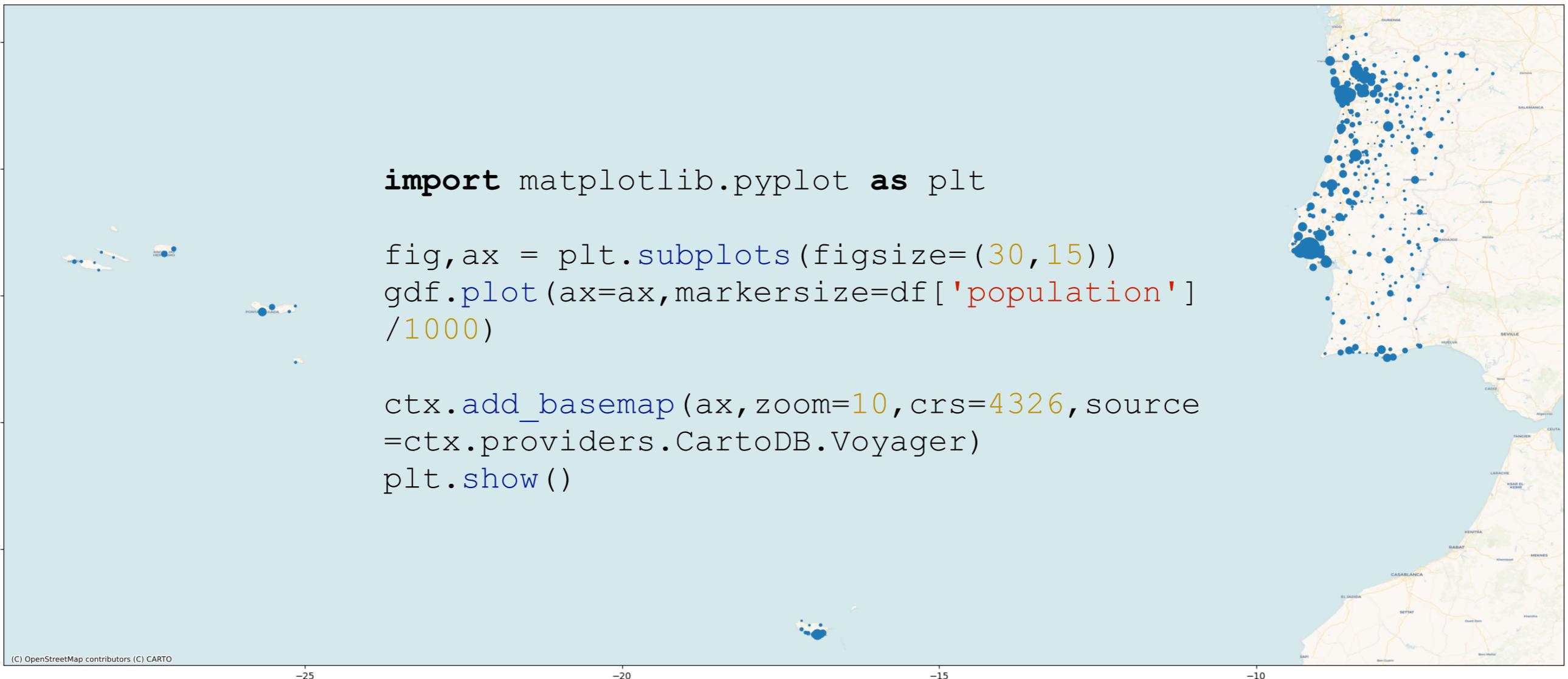
# GeoPandas (plot pontos)

- *GeoDataFrame* de cidades do exemplo anterior
- Desenhar com o *matplotlib*, controlar tamanho de cada ponto

```
import matplotlib.pyplot as plt

fig,ax = plt.subplots(figsize=(30,15))
gdf.plot(ax=ax, markersize=df['population']/1000)

ctx.add_basemap(ax, zoom=10, crs=4326, source
=ctx.providers.CartoDB.Voyager)
plt.show()
```



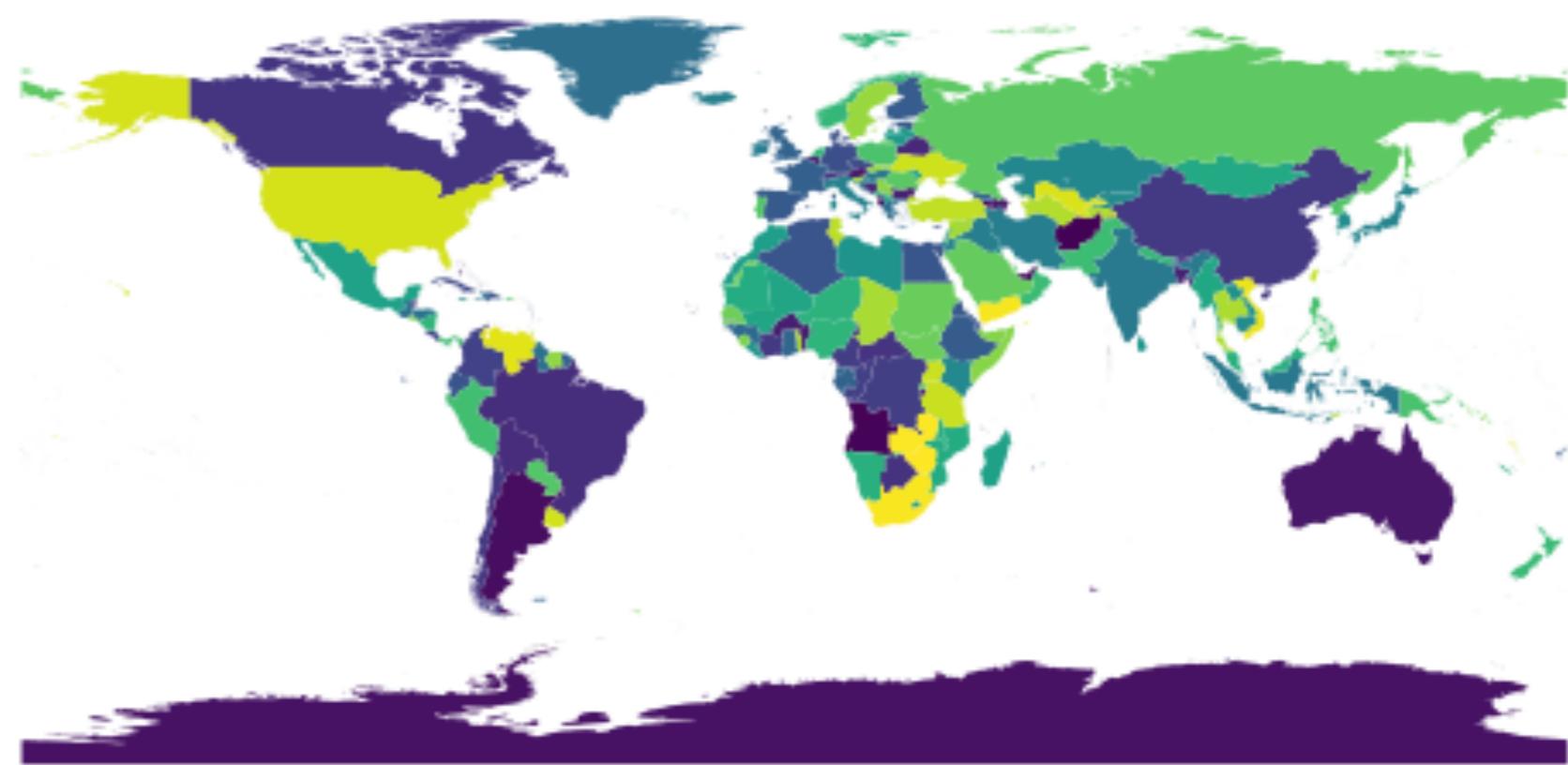
# GeoPandas (plot cores)

- Podemos desenhar mapas de cores, os chamados *choropleth maps*
- E.g., mapa de países de todo o mundo, disponível [aqui](#) em formato GeoJSON

```
import geopandas as gpd
import matplotlib.pyplot
as plt

gdf =
gpd.read_file("countries.
geojson")

ax =
gdf.plot(cmap='viridis')
ax.axis('off')
plt.show()
```



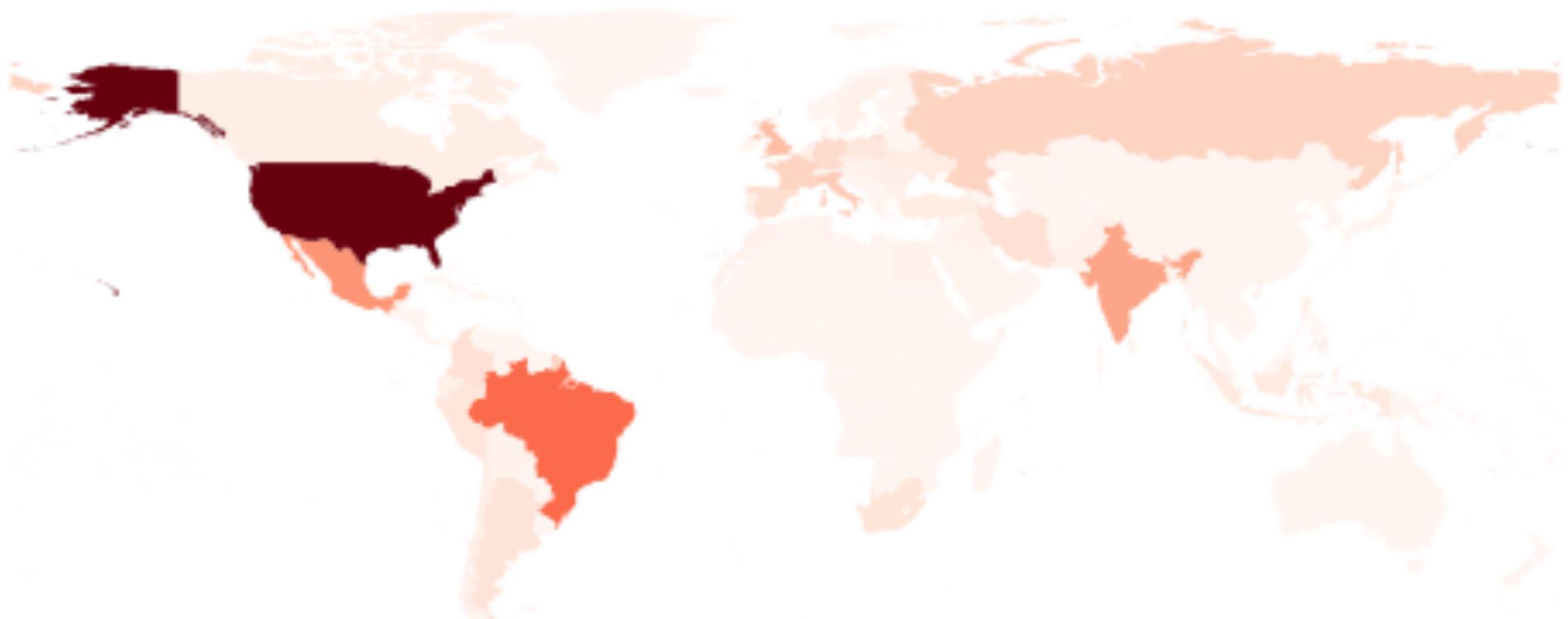
# GeoPandas (plot cores)

- E.g., dados COVID-19 mundiais partilhados pela OMS + códigos de países disponíveis aqui

```
# países com códigos 3-dígitos
gdf3 = gpd.read_file("countries.geojson")
gdf3 = gdf3.rename(columns={'ADMIN': 'Country'})
# conversão códigos
codes = pd.read_csv("country-codes.csv", usecols=['ISO3166-1-Alpha-3', 'ISO3166-1-Alpha-2'])
codes = codes.rename(columns={'ISO3166-1-Alpha-3': 'ISO_A3', 'ISO3166-1-Alpha-2': 'ISO_A2'})
# países com códigos 2- e 3-dígitos
gdf32 = pd.merge(gdf3, codes, how='left')
# dados covid-19 WHO
df = pd.read_csv("WHO-COVID-19-global-data.csv")
df = df.groupby(by='Country').tail(1)
df =
df.rename(columns={'Country_code': 'ISO_A2', 'Cumulative_deaths': 'Deaths'})
df = df[['ISO_A2', 'Deaths']]
gdf = pd.merge(gdf32, df, how='inner')
ax = gdf.plot(column='Deaths', cmap='Reds'); ax.axis('off'); plt.show()
```

# GeoPandas (plot cores)

- E.g., dados COVID-19 mundiais partilhados pela OMS + códigos de países disponíveis [aqui](#)



# Mapas

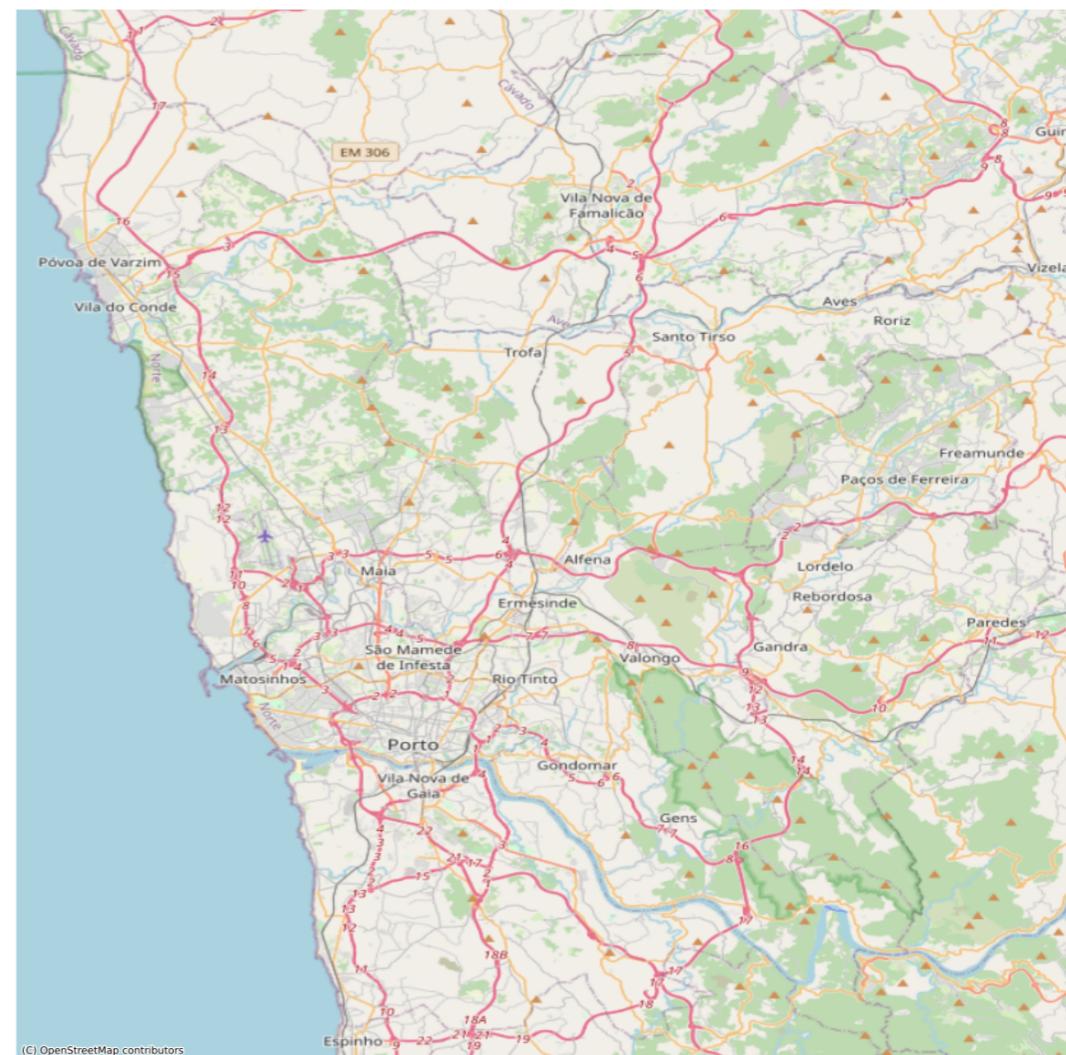
- Plots geopandas ou basemaps *contextily* são gráficos *matplotlib*
  - No gráfico X = longitude e Y = latitude
  - Mapeamento conforme o CRS definido
- Em *matplotlib* podemos desenhar múltiplos gráficos
  - Sobrepondo plots
  - Podemos costumizar mapas com as funcionalidades genéricas do *matplotlib*

# Mapas (guardar)

- Podemos guardar um mapa para um ficheiro de imagem TIF
- Útil para se utilizarmos o mesmo mapa várias vezes ou utilizações offline

```
fig,ax = plt.subplots(figsize=(15,15))
west = -8.8; east = -8.3
south = 41; north = 41.5
ax.set_xlim(west,east)
ax.set_ylim(south,north)
ax.axis('off')

mapnik = ctx.providers.OpenStreetMap.Mapnik
# guarda o mapa para ficheiro
ctx.bounds2raster(west,south,east,north,ll=True,
source=mapnik,path='map.tif')
# lê o mapa de ficheiro
ctx.add_basemap(ax,crs=4326,source='map.tif')
plt.show()
```



# Mapas (sobreposição)

- E.g., desenhar a rede de paragens de transportes urbanos do Grande Porto, disponível aqui em formato CSV

```
df = pd.read_csv('stops.txt')
gdf =
gpd.GeoDataFrame(df,geometry=gpd.points_from_xy(df['stop_
lon'],df['stop_lat']))

fig,ax = plt.subplots(figsize=(15,15))
gdf.plot(ax=ax,markersize=1,color='red')

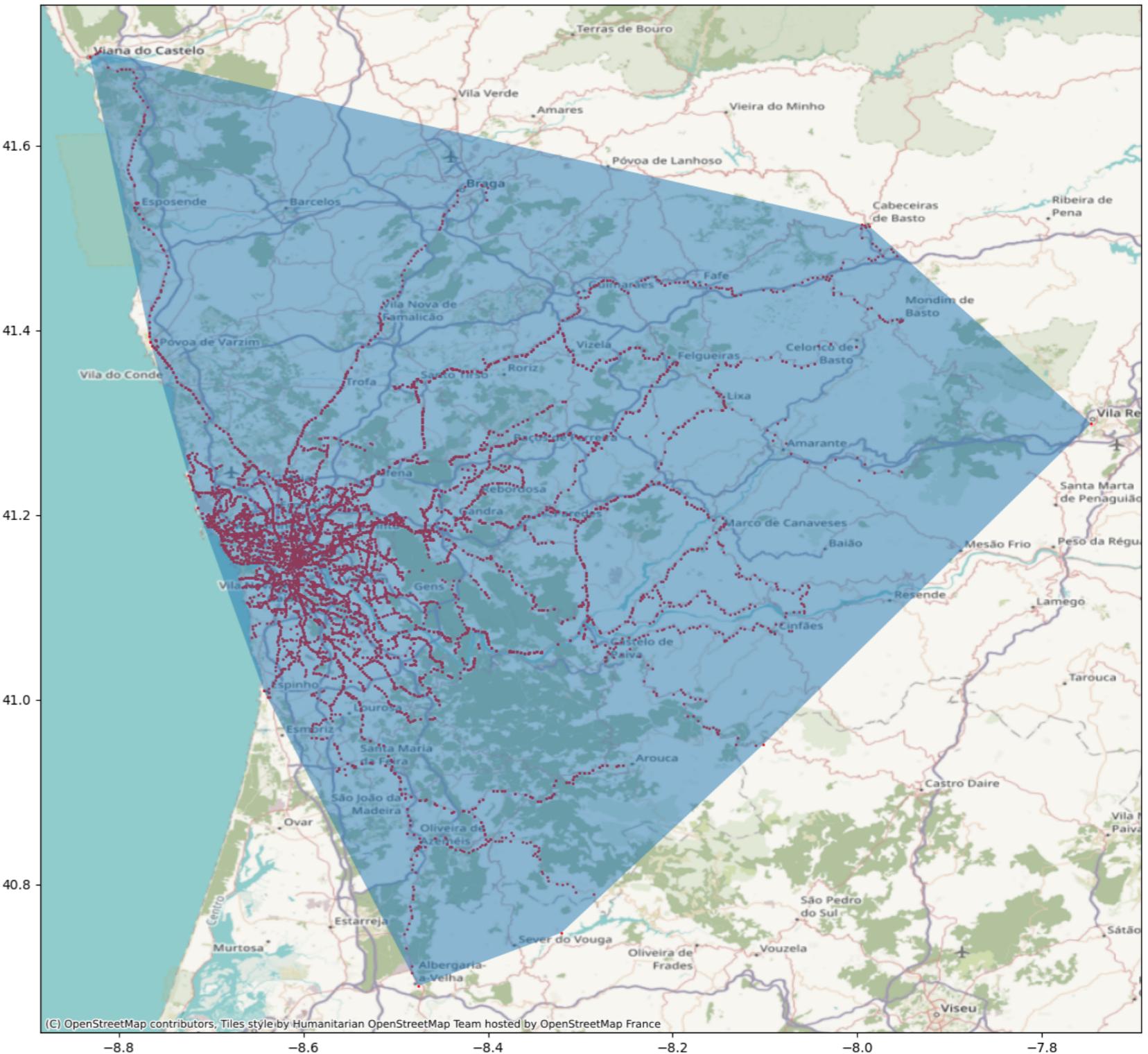
hull = gdf.unary_union.convex_hull
hulls = gpd.GeoSeries([hull])
hulls.plot(ax=ax,alpha=0.5)

ctx.add_basemap(ax,zoom=10,crs=4326,source=ctx.providers.
OpenStreetMap.HOT)

plt.show()
```

# Mapas (sobreposição)

- E.g., desenhar a rede de paragens de transportes urbanos do Grande Porto, disponível [aqui](#) em formato CSV



# Mapas + Matplotlib

- E.g., desenhar um ponto e uma anotação no mapa

```
# Porto
fig,ax = plt.subplots()
ax.set_xlim(-8.8,-8.3)
ax.set_ylim(41,41.5)
ax.axis('off')

#cidade Porto
y = 41.1647; x = -8.6308

topo =
ctx.providers.Stamen.TerrainBackground
ctx.add_basemap(ax,crs=4326,source=topo)

ax.plot(x,y,'ok',markersize=5)
ax.text(x,y,'Porto',fontsize=12);

plt.show()
```



# Mapas + Matplotlib

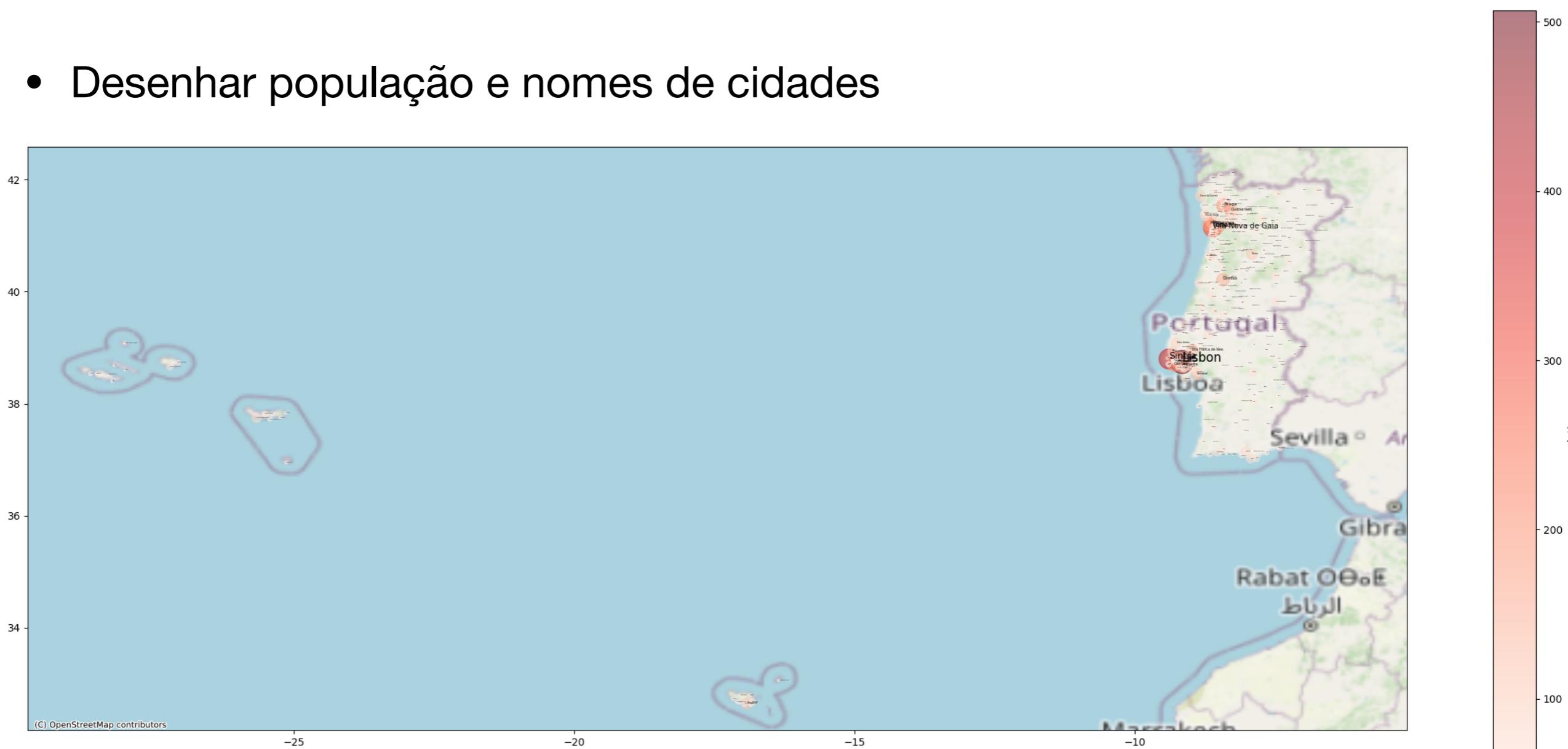
- E.g., lista de cidades de Portugal disponível [aqui](#) em formato CSV
- Desenhar população e nomes de cidades

```
data = pd.read_csv('pt.csv')
data.fillna(1000,inplace=True)
lons = data['lng']
lats = data['lat']
pops = data['population'] / 1000
mpop = data['population'].mean()
maxpop = data['population'].max()

fig,ax = plt.subplots(figsize=(30,15))
for i,row in data.iterrows():
    ax.text(row['lng'],row['lat'],row['city']\
            ,fontsize=row['population']*12/maxpop)
circles = ax.scatter(lons,lats,s=pops,c=pops,cmap='Reds', alpha=0.5)
plt.colorbar(circles,label='population')
ctx.add_basemap(ax,crs=4326,source=ctx.providers.Stamen.Watercolor)
plt.show()
```

# Mapas + Matplotlib

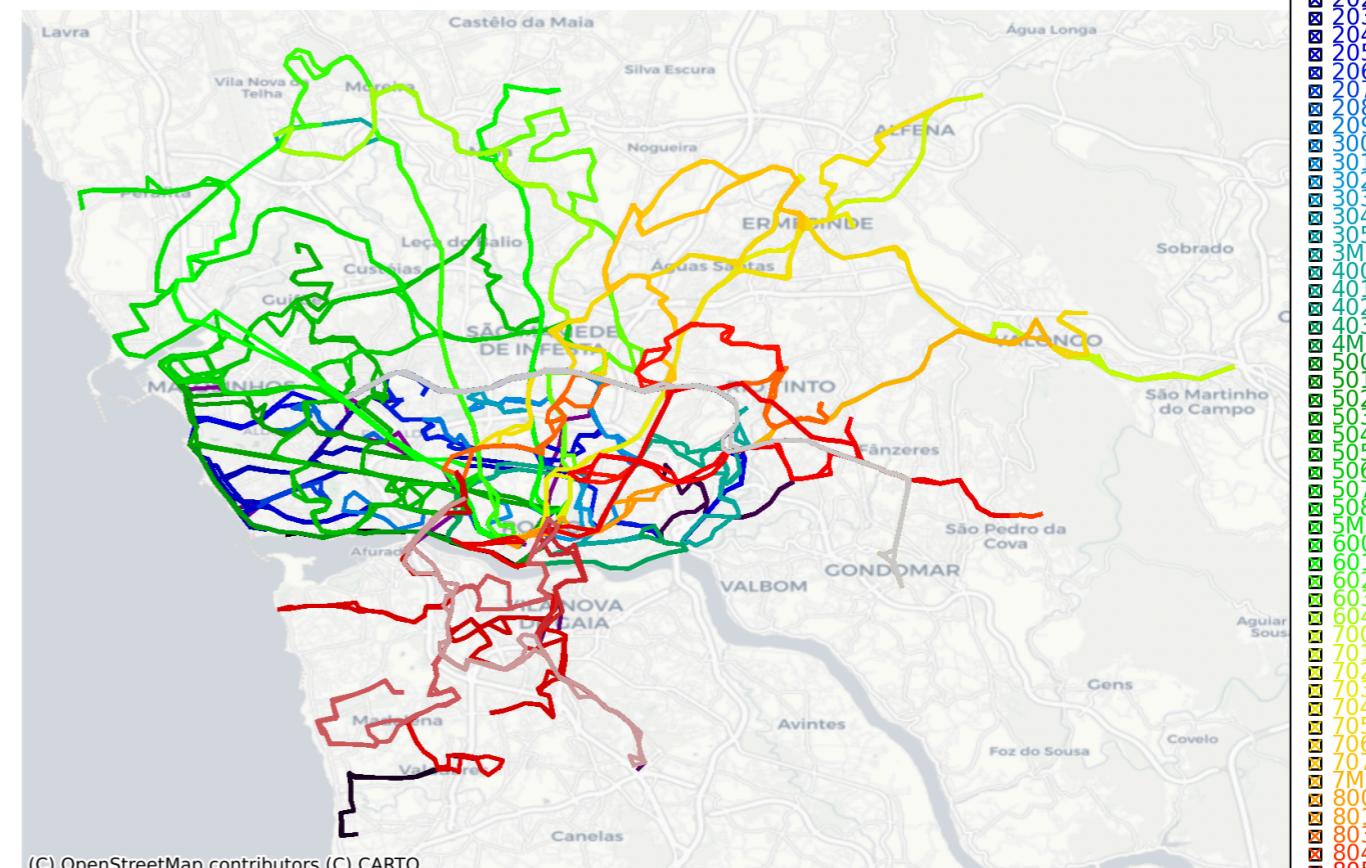
- E.g., lista de cidades de Portugal disponível [aqui](#) em formato CSV
  - Desenhar população e nomes de cidades



# Mapas + Matplotlib

- E.g., desenhar rotas da STCP, disponíveis [aqui](#)

1. Juntar vários CSV (formato GTFS) num só DataFrame com as rotas de cada linha
2. Gerar uma lista de cores, uma por linha
3. Para cada linha/cor, desenhar as várias rotas
4. Criar os botões



# Mapas + Matplotlib

- E.g., mapa de países de todo o mundo disponível [aqui](#) + dados COVID-19 mundiais partilhados pela [OMS](#) + códigos de países disponíveis [aqui](#)
1. Carregar e cruzar dados, como no gráfico não interativo
  2. Ordenar datas por dias desde o dia 0 (saltar dias sem informação)
  3. Fixar margens do mapa
  4. Escolher um mapa de cores
  5. Construir um slider temporal: número de mortes por dia

# Mapas + *Matplotlib*

Day

417

