

Programação II

Mapas

Parte 1

Hugo Pacheco

DCC/FCUP
21/22

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
 - Não vamos ver: outras bibliotecas para mapas com mais funcionalidades (*basemap*, *plotly*, *GeoPy*, *CartoPy*, ...)
 - Exceção para o *Folium* (mapas web)

GeoJSON

- Formato de dados textual para representar dados geográficos
 - Sub-formato de JSON (qualquer ficheiro *geojson* é um ficheiro *json* válido)
 - 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](#)

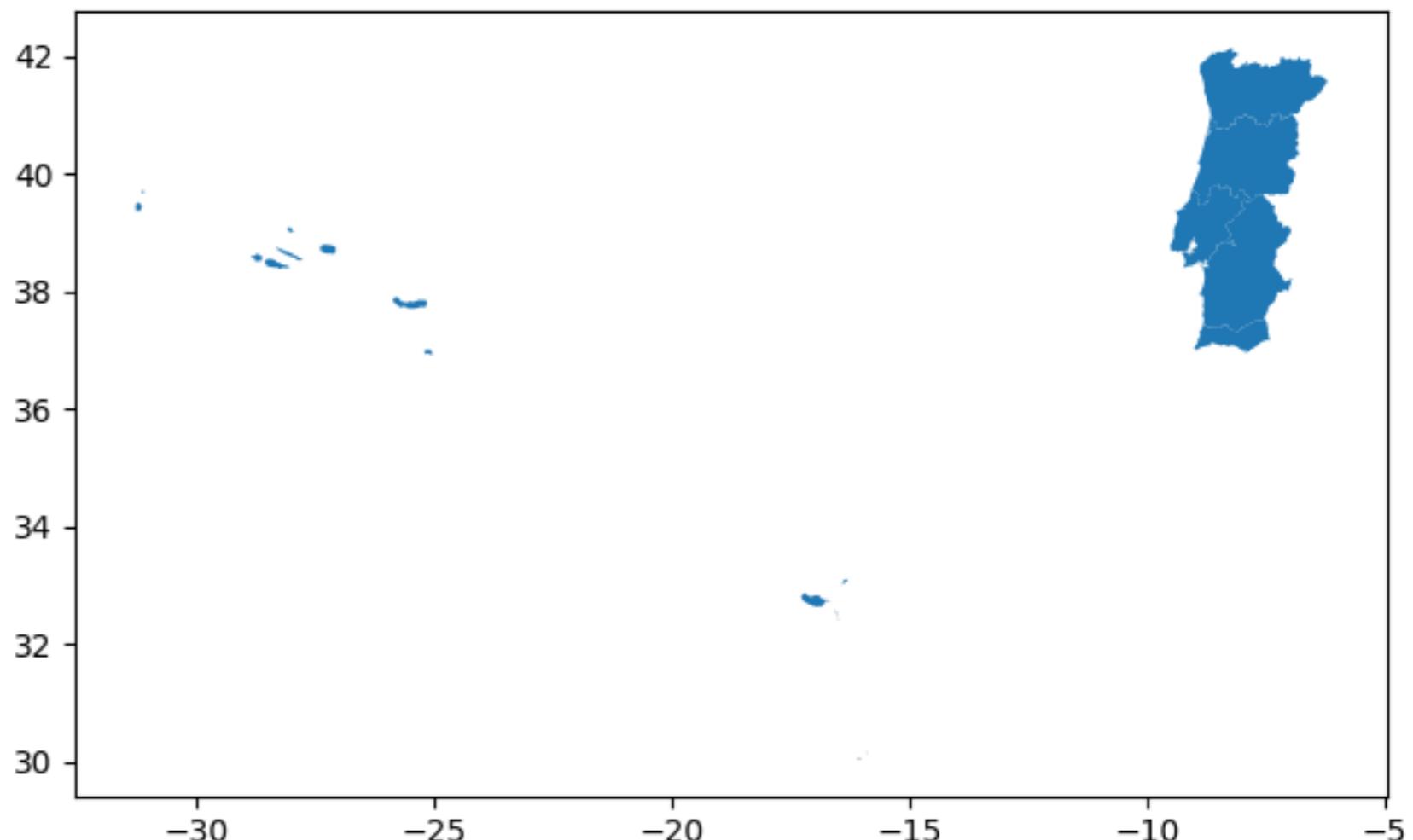
```
import geopandas as gpd
df_map = gpd.read_file("portugal.geojson")
print(df_map)
```

	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...

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)

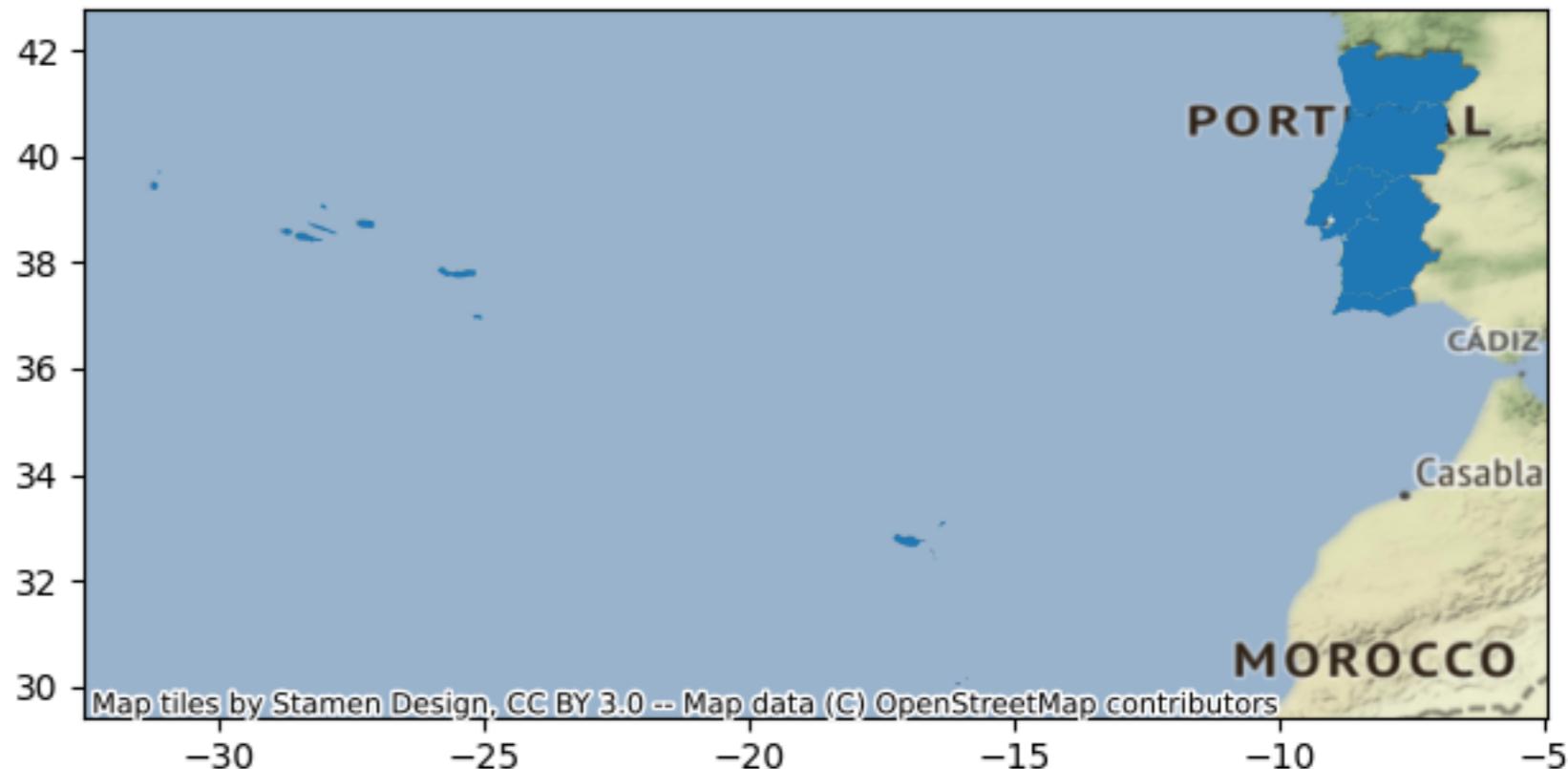
- Podemos desenhar facilmente um mapa de background com o *contextily*
- **Importante:** garantir que o CRS dos dados e do mapa são iguais
 - CRS = Coordinate Reference System
 - Pode haver alguma distorção de perspetiva consoante o CRS

```
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)

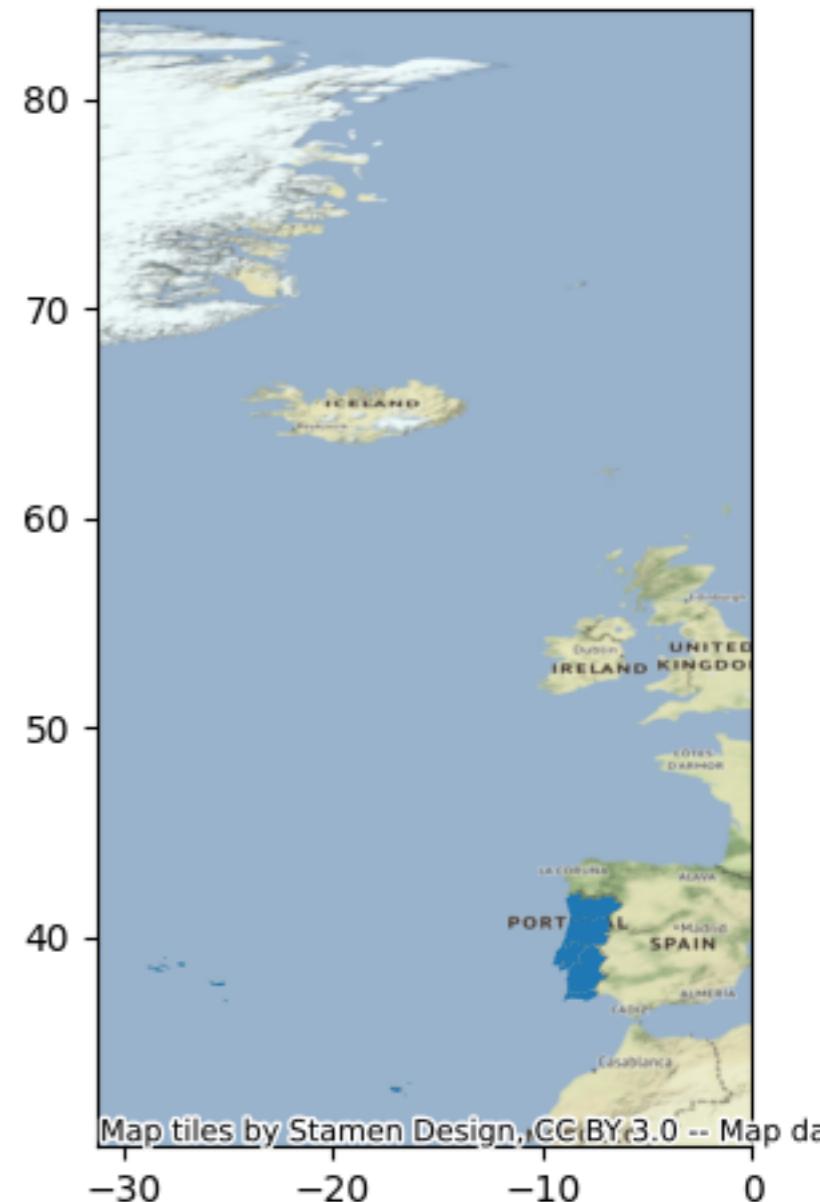
- Podemos alterar o zoom do mapa, controlando os eixos do gráfico
- **Relembrar:** é um gráfico matplotlib normal

```
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:

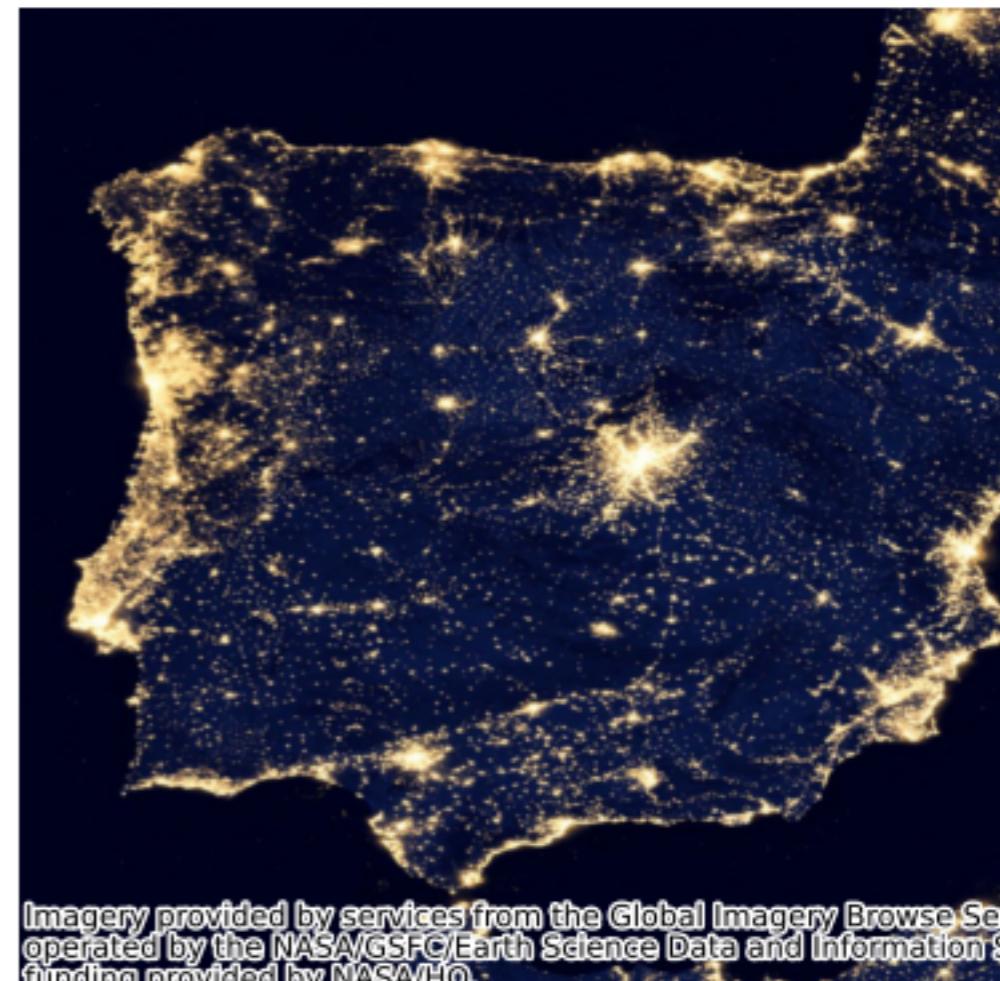
```
>>> ctx.providers.keys()
dict_keys(['OpenStreetMap', 'OpenSeaMap', 'OpenPtMap',
'OpenTopoMap', 'OpenRailwayMap', 'OpenFireMap',
'SafeCast', 'Thunderforest', 'OpenMapSurfer', 'Hydda',
'MapBox', 'Stamen', 'Esri', 'OpenWeatherMap', 'HERE',
'FreeMapSK', 'MtbMap', 'CartoDB', 'HikeBike', 'BasemapAT',
'nImaps', '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()
```

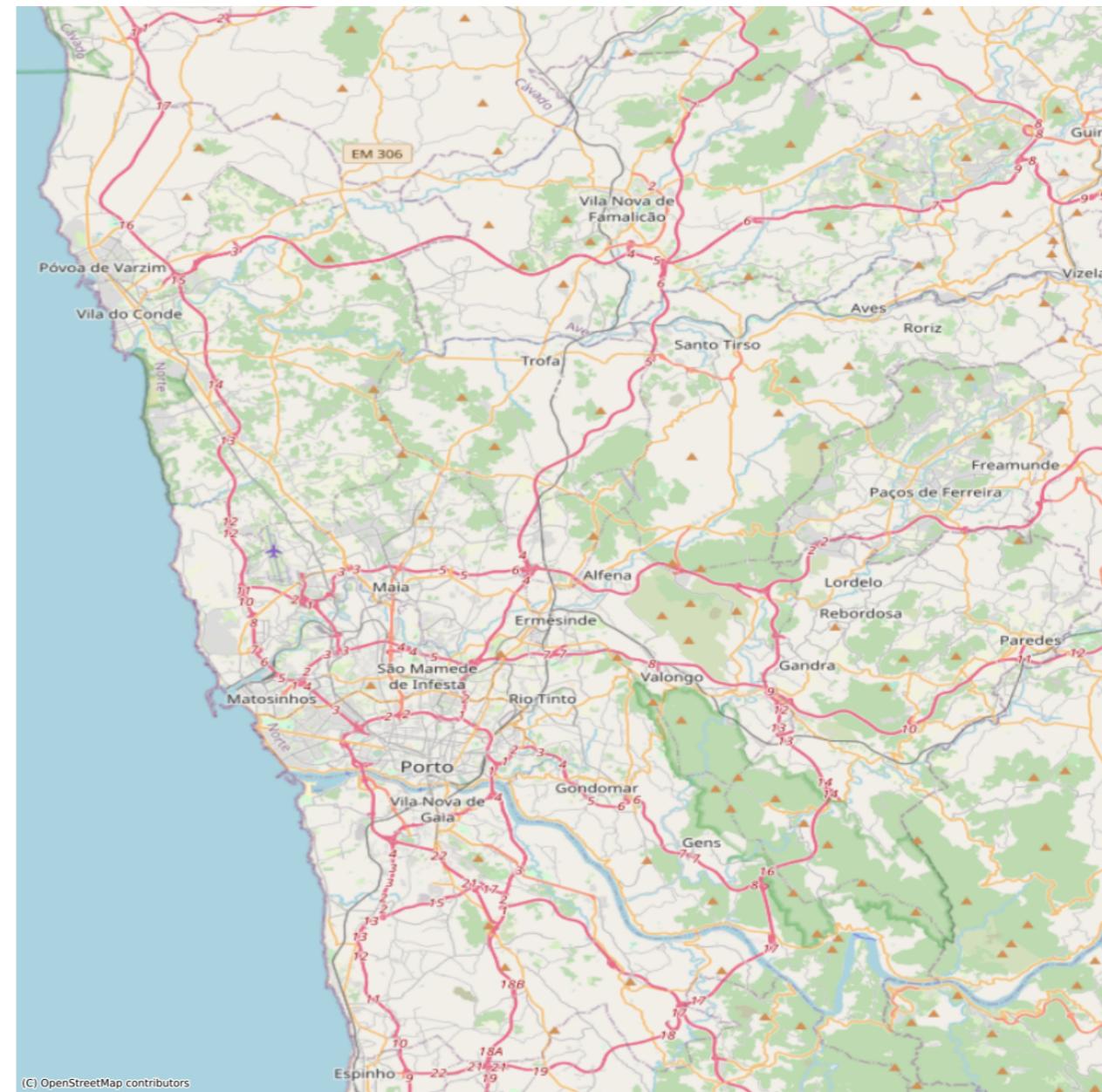


Contextily (mapas)

- 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

```
# Porto alargado
fig,ax = plt.subplots(figsize=(15,15))
ax.set_xlim(-8.9,-7.9)
ax.set_ylim(40.7,41.7)
ax.axis('off')

# múltiplos mapas
colors = ctx.providers.Stamen.Watercolor
labels = ctx.providers.Stamen.TonerLabels
ctx.add_basemap(ax,crs=4326,source=colors)
ctx.add_basemap(ax,crs=4326,source=labels)

plt.show()
```



GeoPandas (criação)

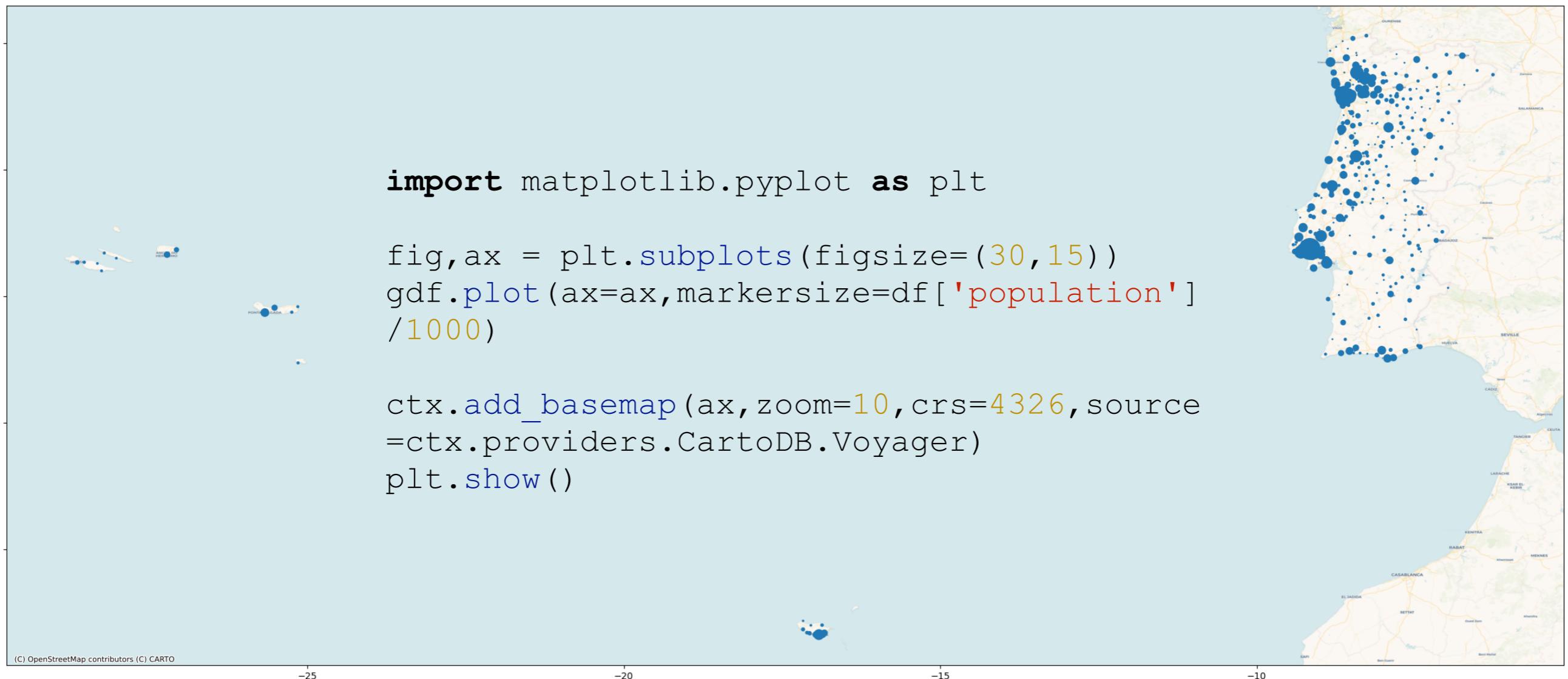
- Podemos criar um novo *GeoDataFrame* a partir de um *DataFrame*, acrescentando uma coluna *geometry*
- E.g., lista de cidades de Portugal disponível [aqui](#) em formato CSV
- Exportar para ficheiro GeoJSON

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



Mapas (sobreposição)

- 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 (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')

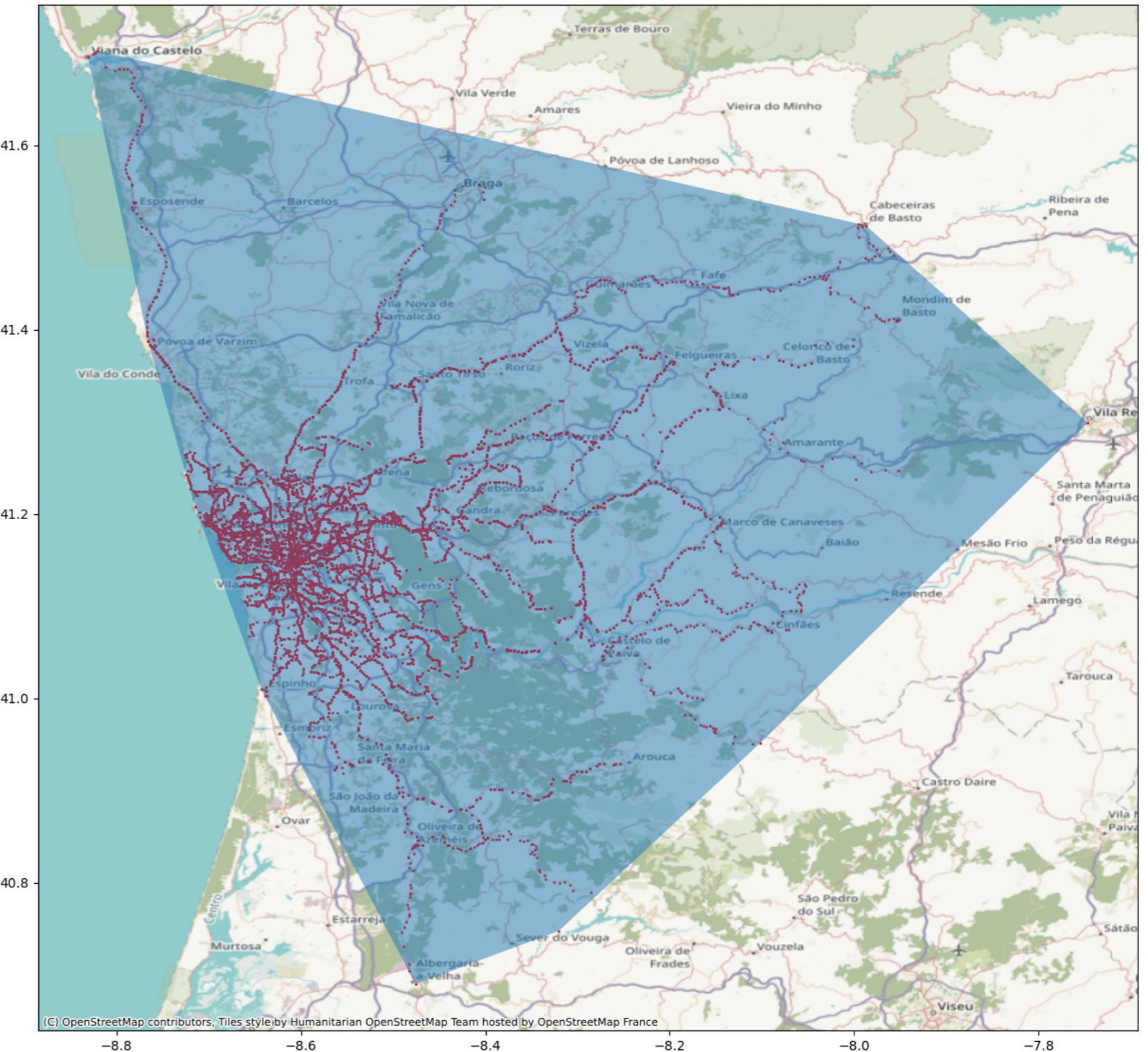
hull = gdf.unary_union.convex_hull
hulls = gpd.GeoSeries([hull])
hulls.plot(ax=ax,alpha=0.5)

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



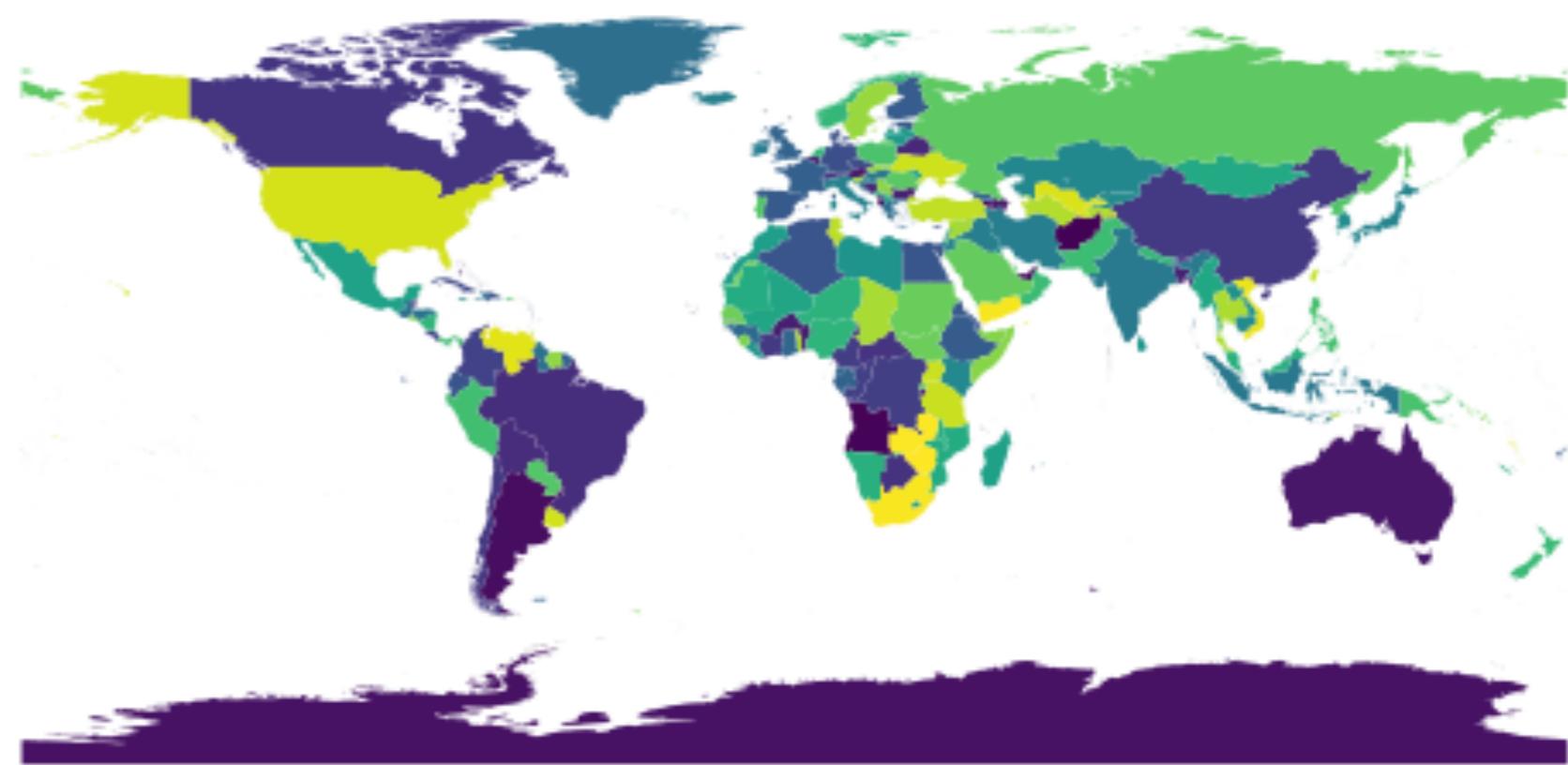
GeoPandas (plot cores)

- Podemos desenhar mapas de cores, os chamados *choropeth 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)

#mortes

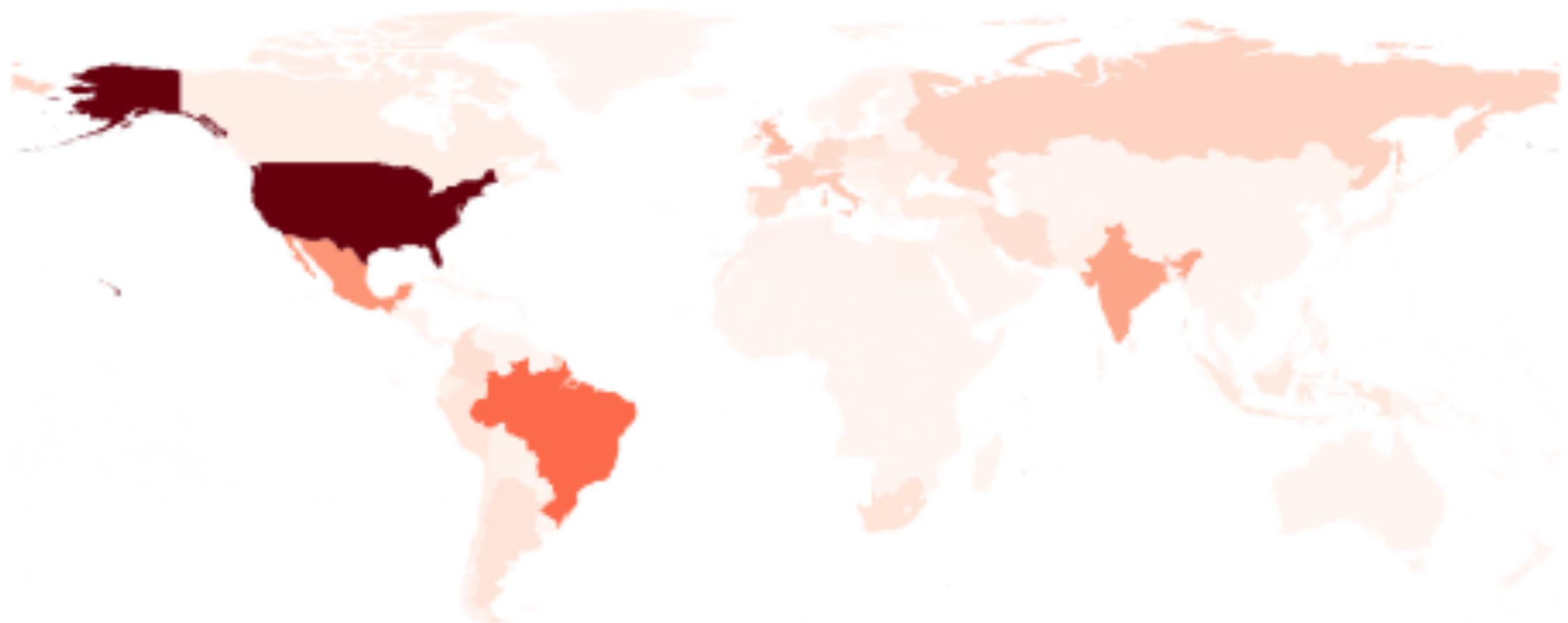
- E.g., dados COVID-19 mundiais partilhados pela OMS + códigos de países disponíveis aqui

```
# 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)

#mortes

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



GeoPandas (plot cores)

#mortes diárias

- E.g., mapa de países de todo o mundo disponível [aqui](#) + dados COVID-19 mundiais partilhados pela [OMS](#) + códigos de países disponíveis [aqui](#)
- Carregar e cruzar dados, como na aula anterior
- Ordenar datas por dias desde o dia 0

```
gdf3 = gpd.read_file("countries.geojson")
gdf3 = gdf3.rename(columns={'ADMIN':'Country'})
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'})
gdf32 = pd.merge(gdf3,codes,how='left')
df = pd.read_csv("WHO-COVID-19-global-data.csv")
df = df.rename(columns={'Country_code':'ISO_A2'})
gdf = pd.merge(gdf32,df,how='inner'); gdf = gdf.dropna()
gdf['Date_reported'] = pd.to_datetime(gdf['Date_reported'])
gdf['day'] = matplotlib.dates.date2num(gdf['Date_reported'])
minday = gdf['day'].min()
gdf['day'] = gdf['day'] - minday
minday = 0
maxday = gdf['day'].max()
```

GeoPandas (plot cores)

#mortes diárias

- E.g., mapa de países de todo o mundo disponível [aqui](#) + dados COVID-19 mundiais partilhados pela [OMS](#) + códigos de países disponíveis [aqui](#)
- Mapa de cores
- Slider temporal: número de mortes por dia

```
_ , ax = plt.subplots(figsize=(30,10))
def draw_day(day):
    gdf[gdf['day'] ==
day].plot(ax=ax, column='New_deaths', cmap='Reds')
    ax.axis('off')
draw_day(maxday)
rect = plt.axes([0.3,0.9,0.5,0.04])
slider =
Slider(rect, 'Day', minday, maxday, valinit=maxday, valstep=1)
def reage(day):
    ax.clear()
    draw_day(day)
slider.on_changed(reage)
plt.show()
```

GeoPandas (plot cores)

#mortes diárias

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

Day

417

