

# **Programação II**

## **Mapas**

### **Parte 2**

Hugo Pacheco

DCC/FCUP  
20/21

# Mapas

- Última aula (*geopandas* + *contextily* + *matplotlib*):
  - Desenhar conjuntos de dados com coordenadas geográficas associadas
  - Desenhar (projeções do) mapa Mundo
- Esta aula (*geopandas* + *contextily* + *matplotlib*):
  - Combinar gráficos *matplotlib* “normais” com mapas
  - Mais exemplos

# 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 (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')

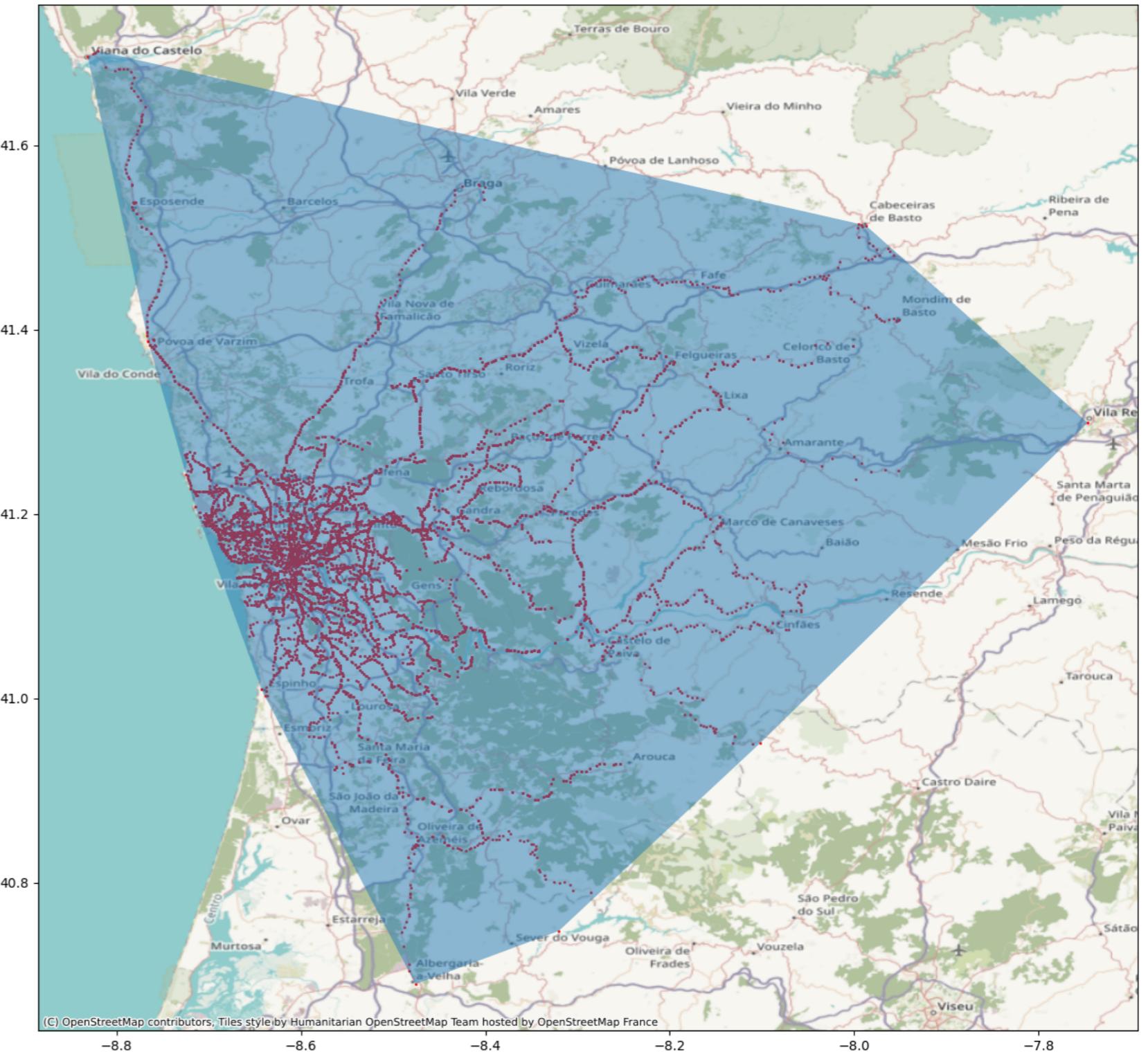
hull = gdf.unary_union.convex_hull
hulls = gpd.GeoSeries([hull])
hulls.plot(ax=ax,alpha=0.5)

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

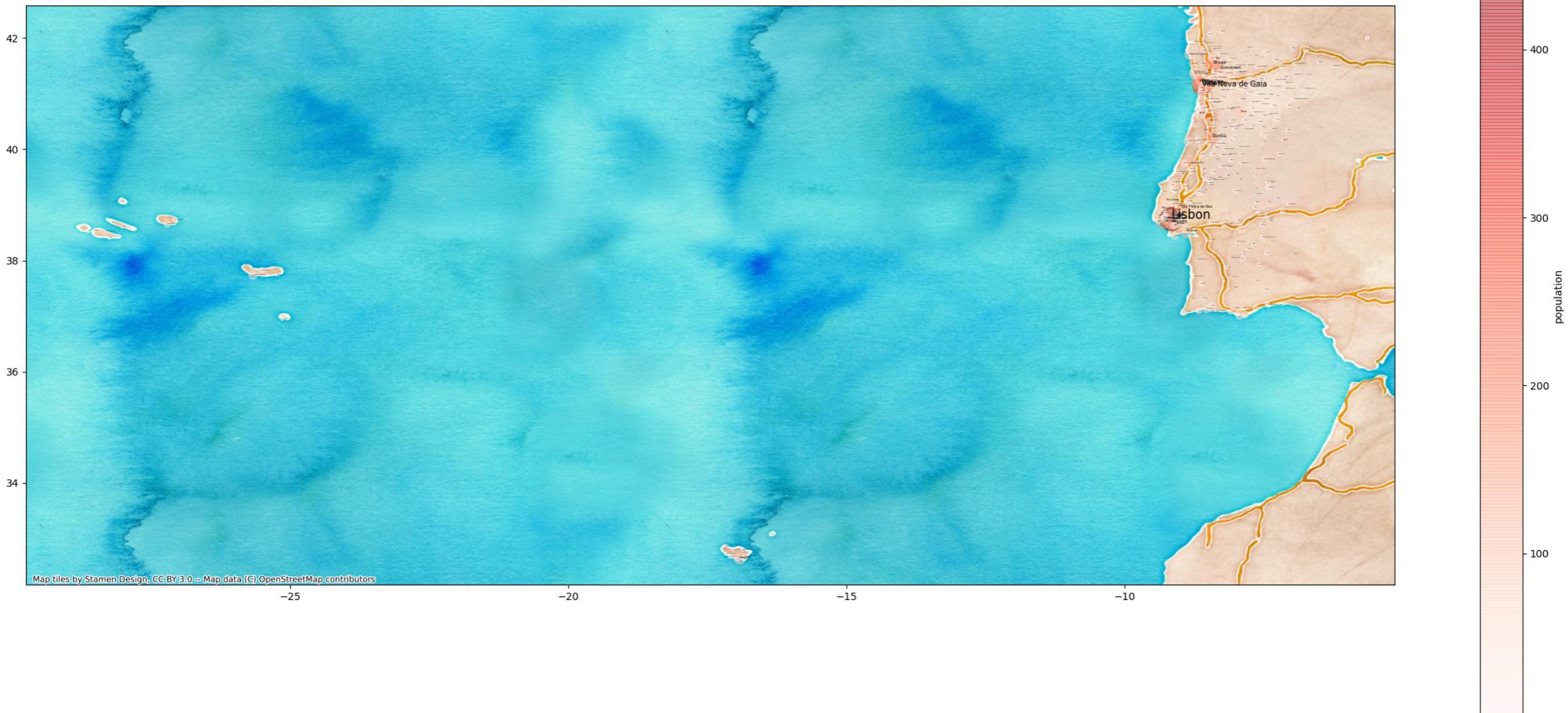
```
data = pd.read_csv('pt.csv')
data.fillna(1000,inplace=True)
lons = data['lng']
lats = data['lat']
pops = data['population'] / 1000
mpop = data['population'].mean()
maxpop = data['population'].max()

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

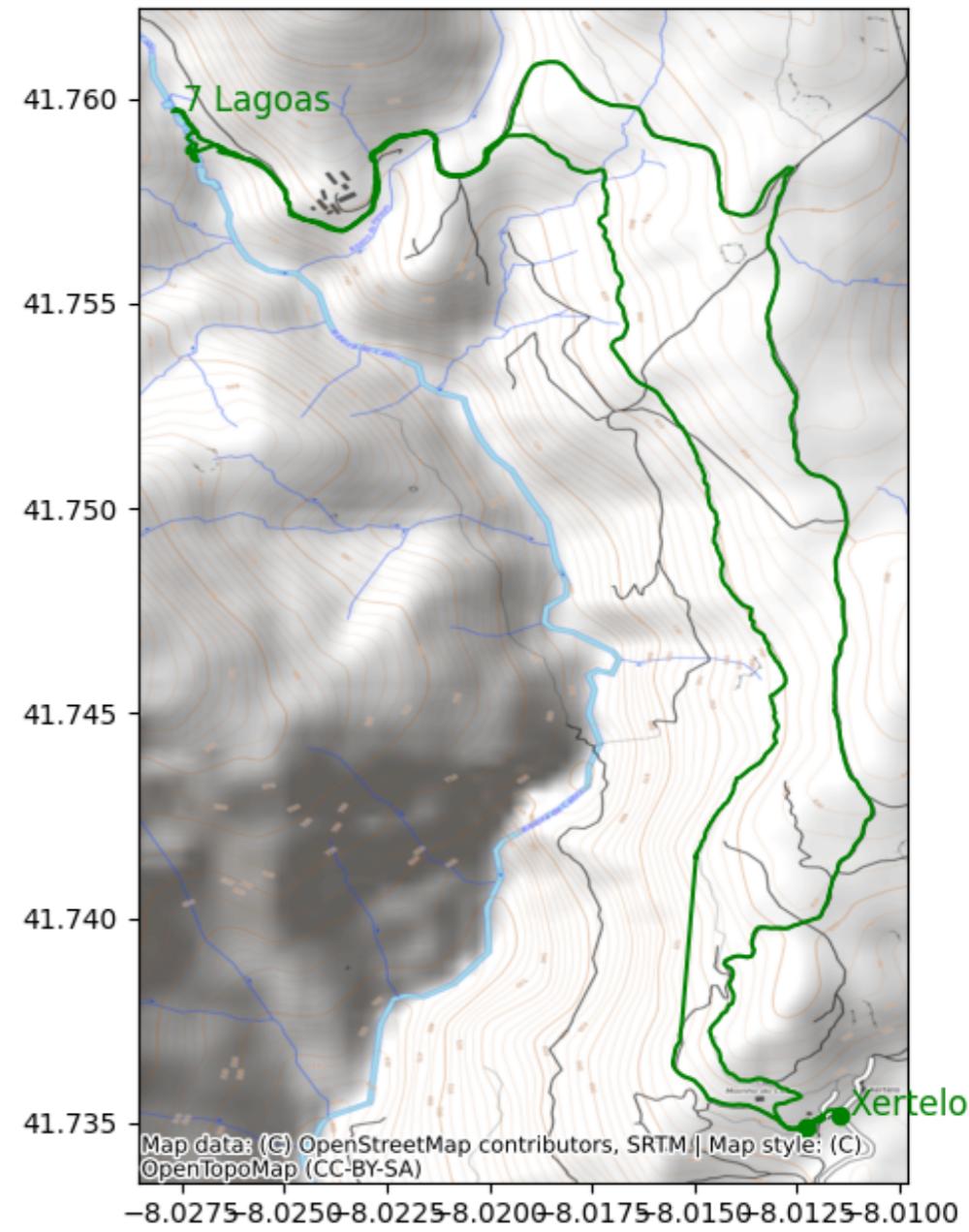
- E.g., desenhar Trilho das 7 Lagoas no Gerês, disponível em formato KML [aqui](#)

```
import fiona
gpd.io.file.fiona.drvsupport.supported_drivers['KML']
] = 'rw'
trilho1 = gpd.read_file('Trilho-das-7-Lagoas.kml',
driver='KML')

fig,ax = plt.subplots(figsize=(5,10))
trilho1.plot(ax=ax,color='green')
topo = ctx.providers.OpenTopoMap
ctx.add_basemap(ax,crs=4326,source=topo)

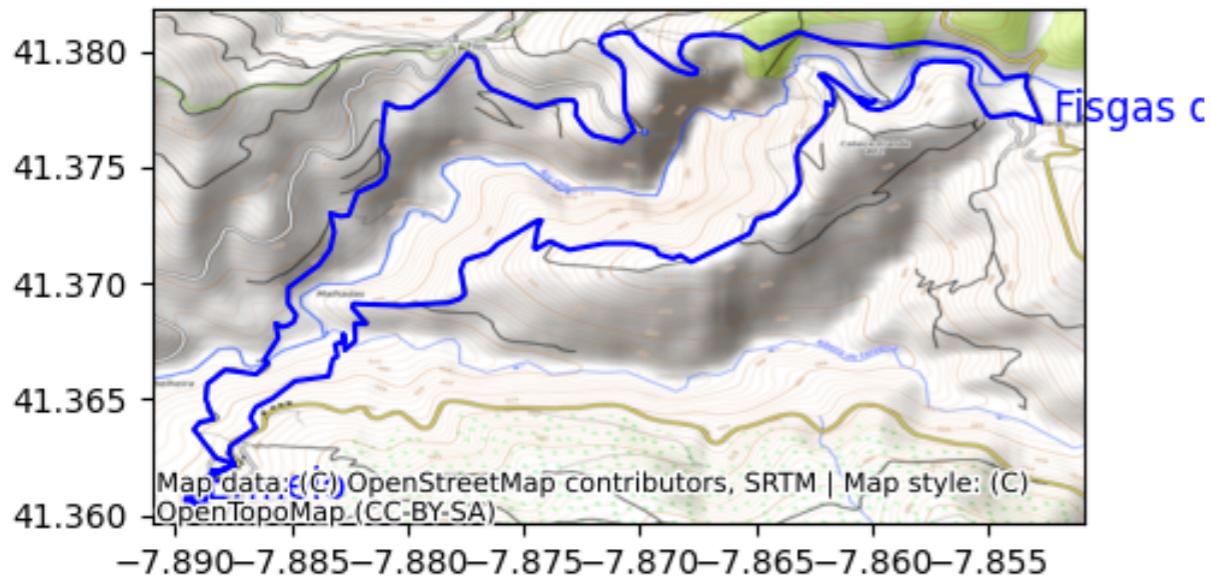
xertelo = trilho1['geometry'][0]
lagoasX = float('inf'); lagoasY = float('inf')
for x,y,z in trilho1['geometry'][2].coords:
    if x < lagoasX: lagoasX=x; lagoasY=y

plt.text(xertelo.x,xertelo.y,
'Xertelo',fontsize=12,color='green');
plt.text(lagoasX,lagoasY, ' 7
Lagoas',fontsize=12,color='green');
plt.show()
```



# Mapas + Matplotlib

- E.g., desenhar trilho das Fisgas do Ermelo no Gerês, disponível em formato KML [aqui](#)



```
trilho2 = gpd.read_file('Fisgas-de-Ermelo.kml', driver='KML')

fig,ax = plt.subplots(figsize=(10,5))
trilho2.plot(ax=ax,color='blue')

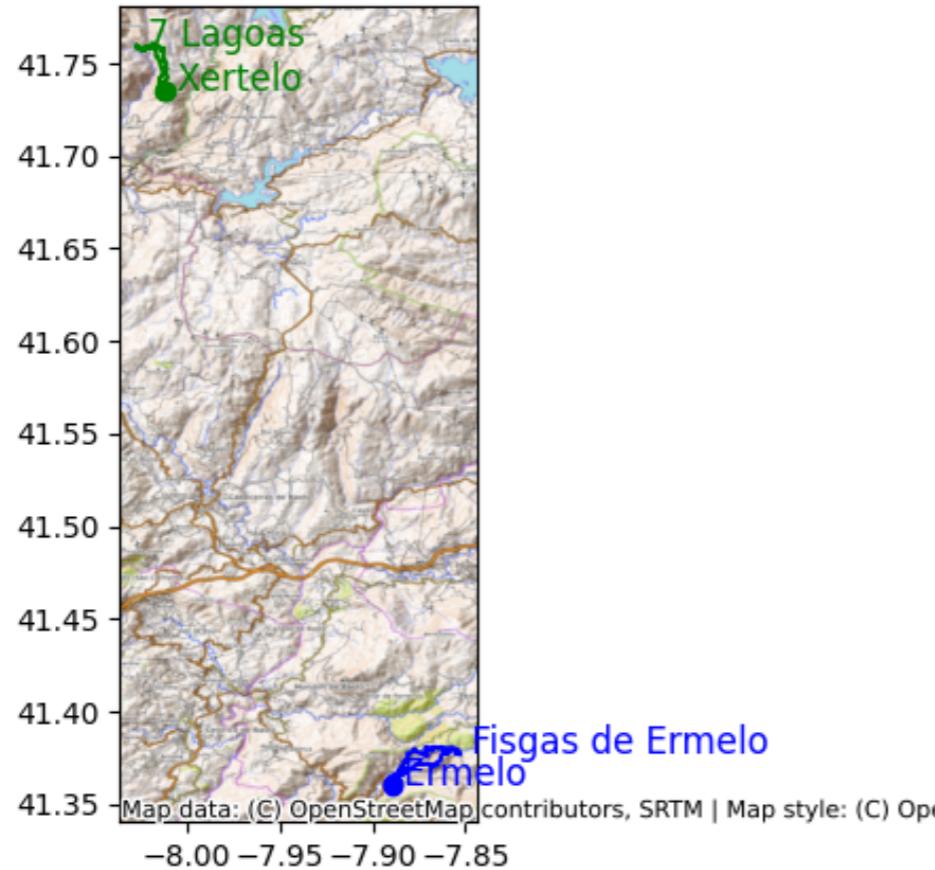
topo = ctx.providers.OpenTopoMap
ctx.add_basemap(ax,crs=4326,source=topo)

ermelo = trilho2['geometry'][2]
fisgasX = float('-inf'); fisgasY = float('-inf')
for x,y,z in trilho2['geometry'][0].coords:
    if x > fisgasX: fisgasX=x; fisgasY=y

plt.text(ermelo.x,ermelo.y,'Ermelo',fontsize=12,color='blue')
plt.text(fisgasX,fisgasY,' Fisgas de Ermelo',fontsize=12,color='blue')
plt.show()
```

# Mapas + Matplotlib

- E.g., combinar os dois trilhos com seleção e zoom in/out



```
from matplotlib.widgets import CheckButtons
lagoas = True; fisgas = True
rect = plt.axes([0.7,0.9,0.1,0.08])
button = CheckButtons(rect,
    ('Lagoas','Fisgas'),(lagoas,fisgas))
trilhos = pd.concat([trilho,trilho2])
b0 = -7.451,41.5,-7.452,41.6
b12 = trilhos.total_bounds
b1 = trilho.total_bounds
b2 = trilho2.total_bounds
def zoom():
    if lagoas:
        if fisgas: bb = b12
        else: bb = b1
    else:
        if fisgas: bb = b2
        else: bb = b0
    minx,miny,maxx,maxy = bb
    ax.set_xlim(minx,maxx)
    ax.set_ylim(miny,maxy)
    plt.draw()
def reage(label):
    global lagoas, fisgas
    if label=='Lagoas': lagoas = not lagoas
    elif label=='Fisgas': fisgas = not fisgas
    zoom()
button.on_clicked(reage)
```

# 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](#)
- 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()
```

# 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](#)
- 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()
```

# 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](#)

