map1

January 28, 2022

1 Maps

inspired by

https://towardsdatascience.com/a-complete-guide-to-an-interactive-geographical-map-using-python-f4c5197e23e0

https://gisco-services.ec.europa.eu/distribution/v2/countries/

https://pypi.org/project/pyshp/

https://towardsdatascience.com/mapping-geograph-data-in-python-610a963d2d7f

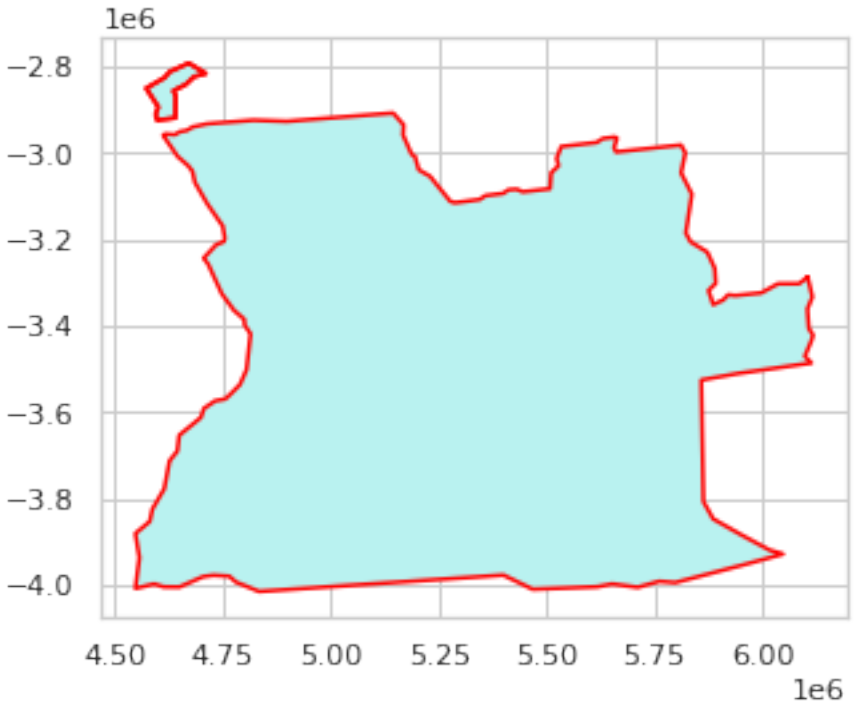
[ ]:
```
import numpy as np
import pandas as pd
import shapefile as shp
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid", palette="pastel", color_codes=True)
sns.mpl.rc("figure", figsize=(10,6))
%matplotlib inline
```

[ ]:
```
shp_path = r"./EU_SHP/CNTR_RG_20M_2020_3035.shp"
sf = shp.Reader(shp_path)
shapeRecs = sf.shapeRecords()

df_shapes = pd.DataFrame(columns = ['country_code', 'shape'])
```

[ ]:
```
country_id = 6
fillcolor = "c"
bordercolor = "red"
df_shapes = pd.DataFrame(columns = ['country_code', 'shape'])



country_shape = shapeRecs[country_id].shape
idx = []
idx = country_shape.parts
n_parts = len(idx)
```

```
n_points = len(country_shape.points)
idx.append(n_points)

plt.figure()
ax = plt.axes()
ax.set_aspect('equal')
for i in range(n_parts): #a country may be composed of several shapes
    p0 = idx[i]
    p1 = idx[i+1]
    if (p1-p0)>10:  #avoid small shapes
        shape_ex = sf.shape(country_id)
        seg=shape_ex.points[p0:p1]
        x_lon = np.zeros((len(seg),1))
        y_lat = np.zeros((len(seg),1))
        nn = len(seg) if i < n_parts else 0
        for ip in range(nn):
            x_lon[ip] = seg[ip][0]
            y_lat[ip] = seg[ip][1]
    plt.plot(x_lon,y_lat, c = bordercolor)
    plt.fill(x_lon,y_lat, fillcolor)

    print("adding to df_shapes", i)
    tmp= [(x_lon[i][0], y_lat[i][0]) for i in range(0, len(x_lon))]
    new_row = {'country_code': 'IT', 'shape': tmp}
    df_shapes = df_shapes.append(new_row, ignore_index = True)
```

```
adding to df_shapes 0
adding to df_shapes 1
adding to df_shapes 2
```



hello

[ ]:
```
data = pd.read_excel("EU_data.xlsx",sheet_name ="data",   usecols = "A:B")
xmin = min(data.x)
xmax = max(data.x)
xnorm = (data.x-xmin)/(xmax-xmin)
```

[ ]:
```
n_shapes = len(shapeRecs)
eunames =␣
 ↪['AL','AT','BE','BG','HR','CH','CY','CZ','DK','EE','FI','FR','DE','GR','HU','IE','IT','LV',
 ↪'GB']
#eunames = ['AT','IT','SI']
#eunames = ['IT', 'FR', 'ES']


fillcolor = "blue"
bordercolor = "white"

x = data

cmap = plt.get_cmap('brg')
colors = cmap(xnorm)
x_min, x_max = 22E5, 70E5
```

```
plt.figure(figsize = (10,10))
#ax = plt.axes()
ax.set_aspect('equal')
df_shapes = pd.DataFrame(columns = ['country_code', 'shape'])

for sh_id in range(n_shapes):
    country_code = shapeRecs[sh_id].record[0]
    if country_code in eunames:
        val = min(data[ data['ISO2'] == country_code ].x)
        country_shape = shapeRecs[sh_id].shape
        idx = []
        idx = country_shape.parts
        n_parts = len(idx)
        n_points = len(country_shape.points)
        idx.append(n_points)

        for i in range(n_parts): #a country may be composed of several shapes
            p0 = idx[i]
            p1 = idx[i+1]
            if (p1-p0)>6:  #avoid small shapes
                shape_ex = sf.shape(sh_id)
                seg=shape_ex.points[p0:p1]
                x_lon = np.zeros((len(seg),1))
                y_lat = np.zeros((len(seg),1))
                nn = len(seg) if i < n_parts else 0
                for ip in range(nn):
                    x_lon[ip] = seg[ip][0]
                    y_lat[ip] = seg[ip][1]

                if (min(x_lon[0]) > x_min) & (max(x_lon[0]) < x_max):
                    fillcolor = (val-xmin)/(xmax-xmin)
                    plt.plot(x_lon,y_lat, c = bordercolor, linewidth =.5, alpha␣
 ↪= 1)
                    plt.fill(x_lon,y_lat, color = cmap(fillcolor), alpha = 1)

                    #print("adding to df_shapes", i)
                    tmp= [(x_lon[i][0], y_lat[i][0]) for i in range(0,␣
 ↪len(x_lon))]
                    new_row = {'country_code': country_code, 'shape': tmp}
                    df_shapes = df_shapes.append(new_row, ignore_index = True)
plt.grid(False)
plt.axis('off')
plt.show
```

[ ]: <function matplotlib.pyplot.show(close=None, block=None)>