Previamente… Módulo 3 - Tarea 5

GeoJSON y coropletas

GeoJSON es un formato basado en JSON para representar simples elementos geográficos junto con atributos no espaciales. Este formato es ampliamente usado en entornos web porque es ligero y fácil de procesar.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [102.0, 0.5]
      },
      "properties": {
        "prop0": "value0"
      }
    }
  ]
}
```

La función siguiente convierte un ESRI shapefile a un GeoJSON

Input[1]:

```
import shapefile
from json import dumps

def shape2json(fname, outfile):
    reader = shapefile.Reader(fname)
    fields = reader.fields[1:]
    field_names = [field[0] for field in fields]
    data = []
    for sr in reader.shapeRecords():
        atr = dict(zip(field_names, sr.record))
        geom = sr.shape.__geo_interface__
        data.append(dict(type="Feature", geometry=geom, properties=atr))
    keys = ['NUTS_NAME']
    for b in data:
        b['properties']['NUTS_NAME'] = b['properties']['NUTS_NAME'].strip('\u0000')
    with open(outfile, "w") as geojson:
        geojson.write(dumps({"type": "FeatureCollection",
                             "features": data}, indent=2) + "\n")
```

Para usarla, solo necesitas introducir el directorio del archivo de entrada y salida.

Input[3]:

```
input_data = "datos/nuts2/NUTS_RG_01M_2016_4326_LEVL_2.shp"
output_data = "datos/nuts2.json"

shape2json(input_data, output_data)
```

GeoJSON en mapas

Folium tiene un buen soporte para crear mapas con GeoJSON.

Solo necesitamos la localización del archivo.

Input[4]:

```
import folium
mapa = folium.Map(zoom_start=3,location=[41.6563,-0.8811])
folium.GeoJson("datos/nuts2.json").add_to(mapa)
```

Output[4]:

```
<folium.features.GeoJson at 0x234bea71608>
```

Folium crea un archive HTML con todo el código de JavaScript necesario y los datos en GeoJSON.

De esta manera podemos resusar los mapas.

La siguiente función guarda este HTML y lo visualiza en Jupyter.

Input[5]:

```
def embed_map(m, name):
    from IPython.display import IFrame
    file_name = 'datos/'+name + '.html'
    m.save(file_name)
    return IFrame(file_name, width=900,height=350)
```

Solo necesitamos usarlo

Input[6]:

```
embed_map(mapa, "nuts2")
```

Output[6]:

Module 3 – Task 6

GEOJSON AND CHOROPLETH

Coropletas con Folium

Folium te permite crear mapas de coropletas en una manera muy flexible. Primero necesitamos los datos.

Input[3]:

```
import pandas as pd
file = 'datos/animalEurostatNuts2_corrected.xlsx'
data = pd.read_excel(file, sheet_name='Data', index_col=0)
data.head(5)
```

Output[3]:

NUTS	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	...	2010	2011	2012	2013	2014
EU	86785.24	86785.24	86785.24	86785.24	86785.24	86785.24	86785.24	86785.24	86785.24	86785.24	...	86785.24	86785.24	86785.24	86785.24	86785.24
BE	3105.50	3099.60	3084.20	3161.10	3158.70	3070.80	2978.40	2984.40	2970.40	3041.60	...	2592.63	2560.32	2484.27	2432.53	2477.24
BE1	0.30	0.50	0.50	0.50	0.40	0.40	0.40	0.30	0.40	0.40	...	0.24	0.25	0.56	0.59	0.79
BE10	0.30	0.50	0.50	0.50	0.40	0.40	0.40	0.30	0.40	0.40	...	0.24	0.25	0.56	0.59	0.79
BE2	1655.20	1661.40	1637.20	1685.50	1678.50	1613.10	1556.80	1554.40	1536.20	1558.10	...	1303.87	1302.25	1269.41	1255.40	1299.98

5 rows × 29 columns

Después necesitamos una rampa de color.

Asumamos que los datos están entre 0 y 4000.

Input[4]:

```
from branca.colormap import linear

colormap = linear.YlGn_09.scale(0, 4000)
colormap
```

Output[4]:

Y ahora usamos la rampa de color para asociar cada NUTS con un color.

Input[5]:

```
color_dict = data['1991'].map(colormap)
color_dict.get('ES11')
```

Output[5]:

```
'#e1f4a9ff'
```

Y para asegurar que dependiendo de cada geometría del NUTS_IS se les da el color correcto. Usaremos el color rojo para indicar la ausencia de datos.

Input[6]:

```
def computar_color(feature):
    return  {
        'fillColor' : color_dict.get(feature["properties"]["NUTS_ID"], '#ff0000'),
        'fillOpacity': 1
    }
```

Ahora visualizaremos la información aplicando el color.

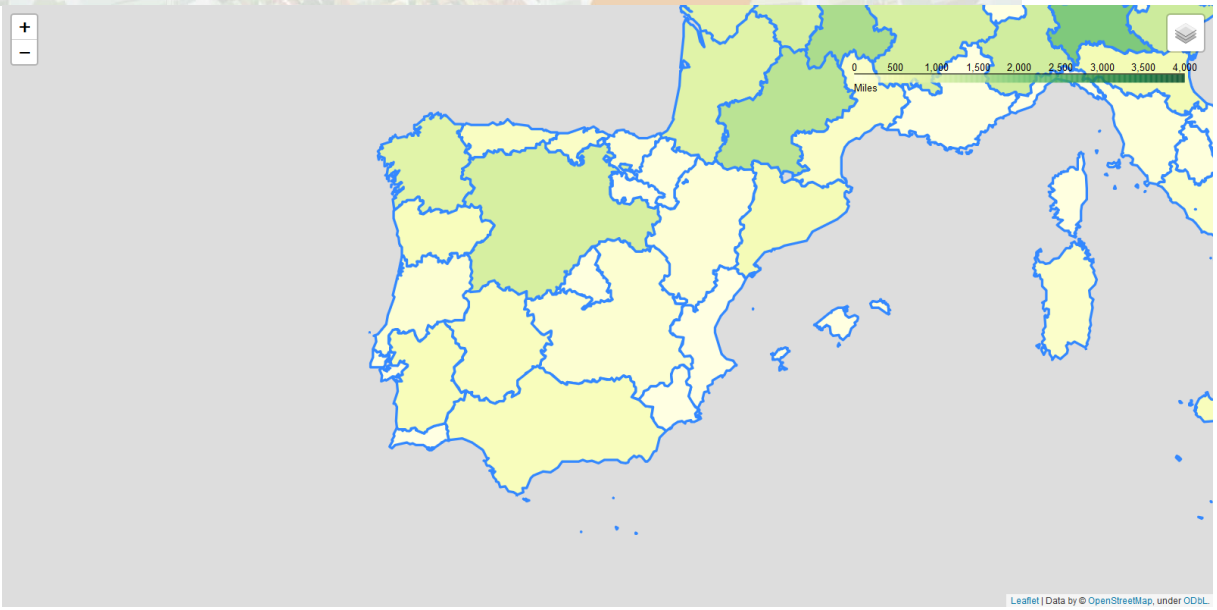
Input[7]:

```
import folium
mapa = folium.Map(zoom_start=3)
folium.GeoJson(
    "datos/nuts2.json",
    name = "Datos 1991",
    style_function = computar_color
).add_to(mapa)
folium.LayerControl().add_to(mapa)
colormap.caption = "Miles"
mapa.add_child(colormap)
mapa.fit_bounds([[43.7483377142, 3.03948408368],
                 [35.946850084, -9.39288367353]])
```

Input[8]:

```
def embed_map(m, name):
    from IPython.display import IFrame
    file_name = 'datos/'+name + '.html'
    m.save(file_name)
    return IFrame(file_name, width=900,height=350)
embed_map(mapa, "choropleth1")
```

Output[8]:



Podemos visualizar en capas datos de diferentes años.

Para hacer esto tienes que cambiar el código.

Input[9]:

```
def computar_color_columna(mapColor):
    def computar(feature):
        return  {
            'fillColor' : mapColor.get(feature["properties"]["NUTS_ID"], '#ff0000'),
            'fillOpacity': 1
        }
    return computar
```

Ahora creamos múltiples capas.

Input[10]:

```
mapa = folium.Map(zoom_start=3)
for column in ['1991', '2001', '2011', '2019']:
    folium.GeoJson("datos/nuts2.json",
        name = "Datos " + column,
        style_function = computar_color_columna(data[column].map(colormap))
    ).add_to(mapa)
folium.LayerControl().add_to(mapa)
colormap.caption = "Miles"
mapa.add_child(colormap)
mapa.fit_bounds([[43.7483377142, 3.03948408368],
                 [35.946850084, -9.39288367353]])
```

Input[11]:

```
embed_map(mapa, "choropleth2")
```

Output[11]: