FICHA TRABALHO 2

MODALIDADE:	Aprendizagem +	Não aplicável	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Visualização de dados em Python	CÓDIGO UFCD:	10809
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Introdução a biblioteca Geopandas
- Instalação e configuração da biblioteca

Introdução ao Geopandas

A biblioteca Geopandas é uma extensão da biblioteca pandas criada para facilitar a manipulação de dados georreferenciados.

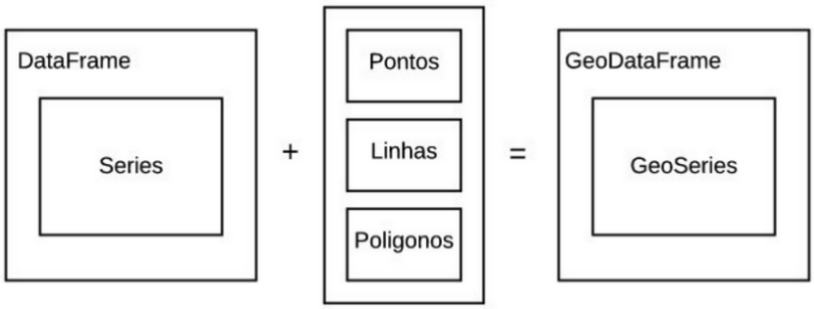
Mas o que são dados georreferenciados?

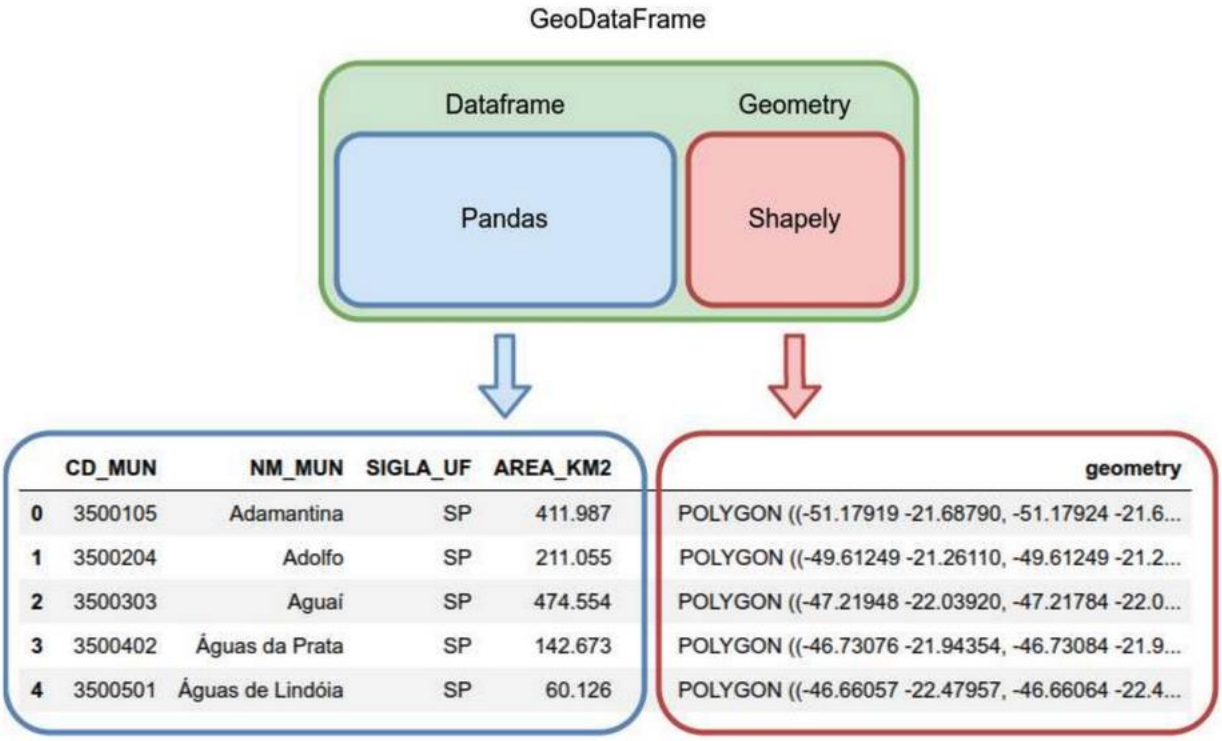
Também conhecidos como dados espaciais, geográficos ou geoespaciais, são dados que possuem as informações para localizar objetos ou entidades na superfície da Terra.

Essas informações são representadas principalmente pelos dados de:

- Latitude
- Longitude

Para fazer isto, o GeoPandas utiliza estruturas de dados geométricas implementadas pela biblioteca Shapely dentro das Series e dos DataFrames. Com esta adição, duas estruturas de dados surgem, as GeoSeries e os GeoDataFrames.



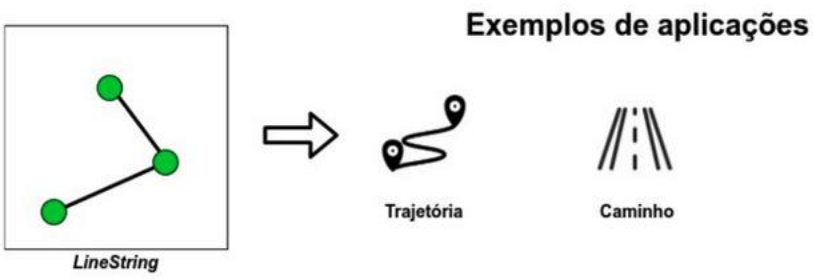


	CD_MUN	NM_MUN	SIGLA_UF	AREA_KM2	geometry
0	3500105	Adamantina	SP	411.987	POLYGON ((-51.17919 -21.68790, -51.17924 -21.6...
1	3500204	Adolfo	SP	211.055	POLYGON ((-49.61249 -21.26110, -49.61249 -21.2...
2	3500303	Aguaí	SP	474.554	POLYGON ((-47.21948 -22.03920, -47.21784 -22.0...
3	3500402	Águas da Prata	SP	142.673	POLYGON ((-46.73076 -21.94354, -46.73084 -21.9...
4	3500501	Águas de Lindóia	SP	60.126	POLYGON ((-46.66057 -22.47957, -46.66064 -22.4...

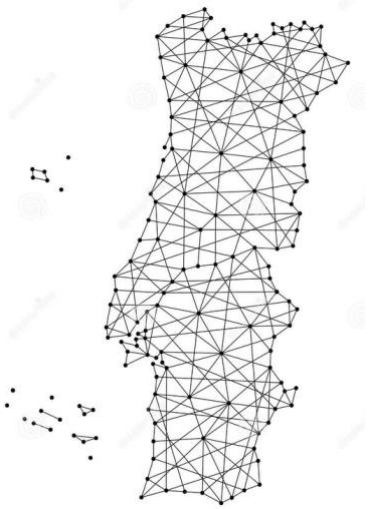
Com base nas informações de geometria que são fornecidos nos ficheiros, podemos analisar:

- **Pontos**: criar pontos de interesse, assinalar localizações entre outras situações;

- **Traçar linhas**: delinear trajetórias, caminhos entre outras situações;



- Criar polígonos: com a combinação dos elementos anteriores (pontos e polígonos), podemos criar divisões, relatórios geográficos ou até mesmo projetar mapas;



Instalação biblioteca Geopandas

Para instalar o Geopandas deve abrir o terminal do Visual Studio Code e digitar a instrução:

- **Windows:** pip install geopandas
- **MacOS:** pip3 install geopandas

Ficheiro Shapefile com os polígonos do mapa de Portugal

Na página da **Direção-Geral do Território** é disponibilizado um documento chamado "Carta Administrativa Oficial de Portugal (CAOP)" que permite descarregar as informações dos dados territoriais do nosso país. Hiperligação: https://www.dgterritorio.gov.pt/cartografia/cartografia-tematica/caop

Nesta página, podemos vamos utilizar os dados no ano de 2022 e vamos descarregar o ficheiro na seguinte hiperligação: https://geo2.dgterritorio.gov.pt/caop/CAOP_Continente_2022-shp.zip e extrair a pasta na ficha_2 para conseguir importar os ficheiros necessários para o laboratório do jupyter.

Ficheiros Shapefiles (.shp)

Um shapefile (ficheiros .shp) é um formato de armazenamento de dados de vetor para armazenar a posição, a forma e os atributos de configurações geográficas. É armazenado como um conjunto de ficheiros relacionados entre si e contém uma classe de configurações. Os shapefiles normalmente contêm muitos dados associados e foi historicamente utilizado em aplicações de desktop GIS - Sistema de Informação Geográfica.

Os dados são armazenados normalmente em ficheiros .zip que contêm pelo menos os ficheiros .shp, .shx, .dbf e .prj do shapefile.

Após a instalação, para poder usar a nova biblioteca, basta importar a biblioteca no início do ficheiro do programa:

```
#importar a biblioteca do geopandas
import geopandas
```

Mas, no código não queremos chamar sempre por todo o nome da biblioteca.

Como tal, **vamos criar um nome mais curto** que vai **representar tudo o nome da biblioteca**, ajudando na hora da invocação. ***A frente*** do nome da biblioteca podem colocar a palavra reservada ***as*** com o nome alternativo a frente:

Exemplo: **import** nome_biblioteca_a_usar **as** nome

```
#importar a biblioteca do geopandas
import geopandas as gpd
```

Leitura de um ficheiro com dados espaciais - Shapefiles (.shp)

Para ler ficheiros .shp (shapefiles), deve criar uma variável e usar a biblioteca do geopandas (acrónimo do gdp) com a função **read_file()**, tal como demonstra a seguinte imagem:

```
# Ler o ficheiro do shapefile
freguesias = gpd.read_file("Cont_Freg_CAOP2022.shp")
```

Tutorial GeoPandas

Vamos começar por fazer a criação de uma pasta com o nome “ficha_2” e dentro desta a importação dos ficheiros que forma anexados no moodle. De seguida, faça a importação das 3 biblioteca:

```
#importar bibliotecas geopandas e matplotlib
import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd
```

Para ler ficheiros .shp (shapefiles), deve criar uma variável e usar a biblioteca do geopandas (acrónimo do gdp) com a função **read_file()**, tal como demonstra a seguinte imagem:

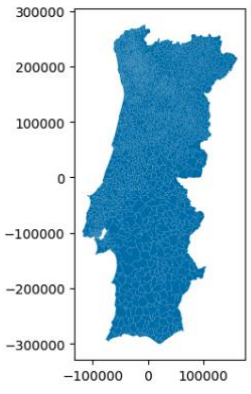
```
# Ler o ficheiro do shapefile
freguesias = gpd.read_file("continente/Cont_Freg_CAOP2022.shp")
```

```
freguesias.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 2882 entries, 0 to 2881
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   DICOFRE     2882 non-null   object
 1   Freguesia   2882 non-null   object
 2   Concelho    2882 non-null   object
 3   Distrito    2882 non-null   object
 4   Area_ha     2882 non-null   float64
 5   Des_Simpli  2882 non-null   object
 6   geometry    2882 non-null   geometry
dtypes: float64(1), geometry(1), object(5)
memory usage: 157.7+ KB
```

Ou então pedir para mostrar o que carregou em formato visual:

```
freguesias.plot()
```



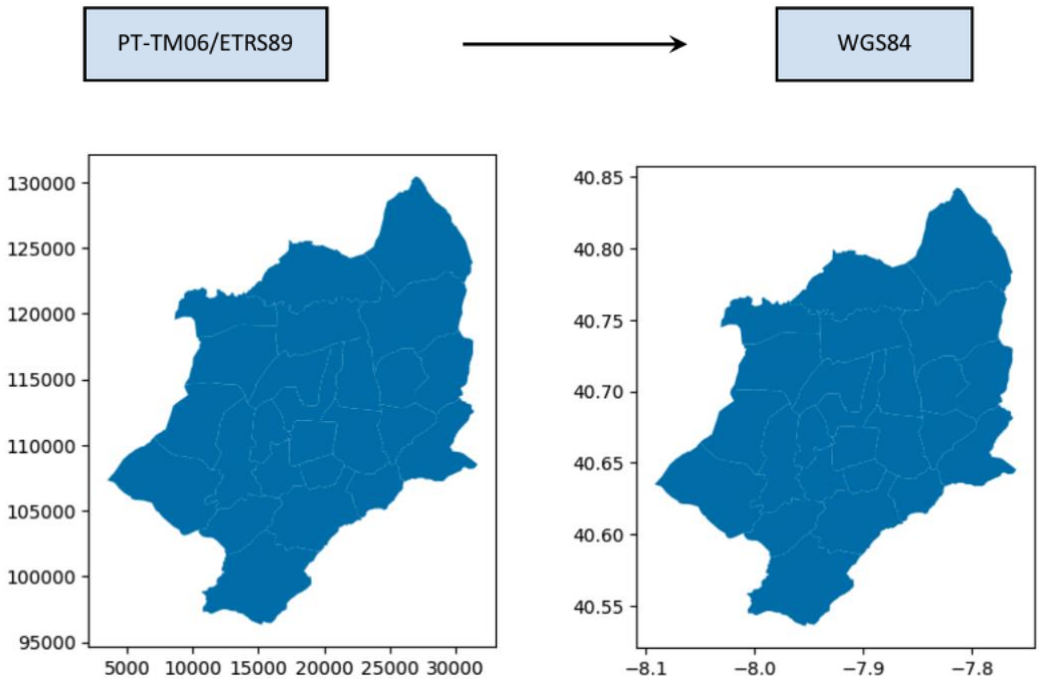
Converter Modelo CRS PT-TM06/ETRS89 para WGS84 (latitude/longitude)

Por defeito, quando pedimos para mostrar o mapa de polígonos com a função plot(), este mostra medidas no modelo **CRS** PT-TM06/ETRS89.

Um **sistema de referência de coordenadas (CRS)** define, com a ajuda de coordenadas, como o mapa bidimensional projetado relaciona com locais reais na terra, ou seja, é um sistema de coordenadas retangulares.

No entanto, podemos pedir ao programa para mudar o modelo **CRS** para **WGS84** que utiliza as coordenadas GPS com latitude e longitude (caso queria aplicar estes polígonos em mapas web), ou seja, é um sistema de coordenadas geográficas.

```
# Converter CRS para WGS84 (latitude/longitude) se quiser usar em mapas web
freguesias = freguesias.to_crs(epsg=4326)
#mostrar novamente o mapa
freguesias.plot()
```



Filtrar a informação do ficheiro – Campos Distrito e Concelho

No ficheiro que carregou, podemos observar que temos as colunas Distrito e Concelho. Se desejar pode pedir os dados ao utilizar com a função input() e escrever o nome do Distrito e/ou Concelho:

```
# Solicitar ao utilizador o distrito e o concelho pretendido
distrito = input("Digite o nome do distrito: ")
concelho = input("Digite o nome do concelho: ")

freguesia_viseu = freguesias.query(' Distrito == "' + distrito + '" & Concelho == "' + concelho + '" ')
freguesia_viseu
```

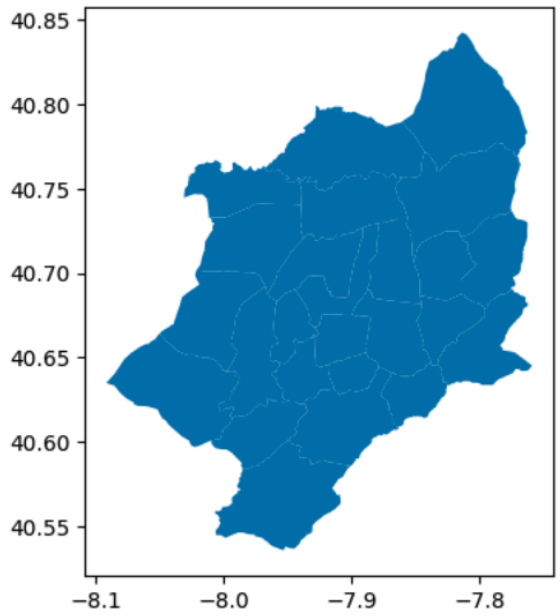
```
Digite o nome do distrito:  Viseu
Digite o nome do concelho:  Viseu
```

	DICOFRE	Freguesia	Concelho	Distrito	Area_ha	Des_Simpli	geometry
2817	182301	Abraveses	Viseu	Viseu	1222.72	Abraveses	POLYGON ((20774.471 113518.55, 20774.332 11350...
2818	182304	Bodiosa	Viseu	Viseu	2539.50	Bodiosa	POLYGON ((16379.576 118889.398, 16376.516 1188...
2819	182305	Calde	Viseu	Viseu	3505.68	Calde	POLYGON ((23023.287 124581.33, 23569.828 12392...
2820	182306	Campo	Viseu	Viseu	1624.04	Campo	POLYGON ((15982.012 113503.934, 15965.952 1134...
2821	182307	Cavernães	Viseu	Viseu	1314.39	Cavernães	POLYGON ((26853.378 117120.3, 26859.569 117099...
2822	182310	Cota	Viseu	Viseu	4154.75	Cota	POLYGON ((27464.082 130011.166,

Depois disso, basta fazer a aplicação da função **plot()** na variável que filtrou as informações, do qual tem os dados de geometria:

```
freguesia_viseu.plot()
```

```
<Axes: >
```



Propriedade Centroid – Cores de fundo e traços

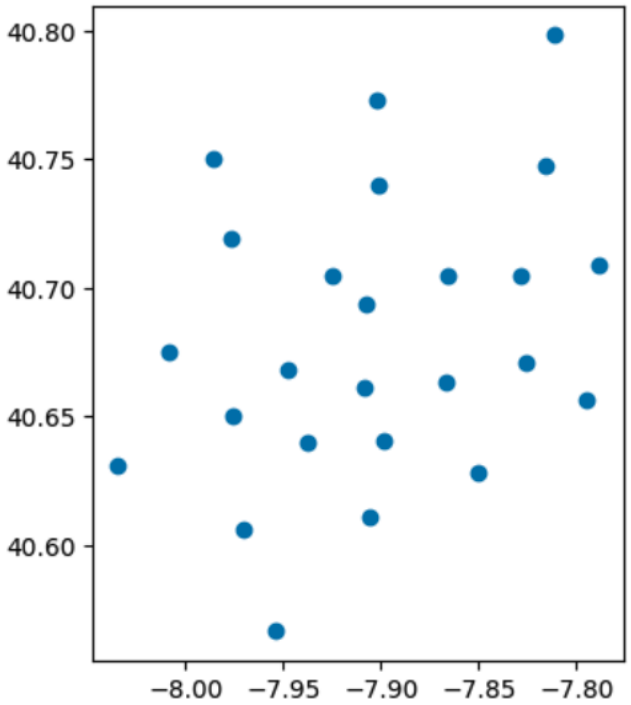
Depois de filtrar as informações por distrito e concelhos, podemos pedir para indicar os pontos que representam o centro geométrico de um polígono com a propriedade centroid. Este permite devolver a lista de informações dos pontos, tal como demonstra a seguinte imagem:

```
#pontos que representam o centro geométrico de um polígono
centroids_freguesias = freguesia_viseu.centroid
centroids_freguesias
```

```
2817    POINT (19030.209 113891.781)
2818    POINT (13245.447 116677.985)
2819    POINT (19510.353 122693.147)
2820    POINT (17612.566 115149.256)
2821     POINT (25738.26 115104.389)
2822    POINT (27145.937 125514.542)
2823    POINT (23887.587 106662.053)
2824     POINT (19546.26 119045.871)
2825      POINT (15160.51 99758.195)
```

Eventualmente, pode mostrar graficamente os pontos das localizações dos concelhos, pode usar a função plot() (mas ainda vão estar separados do mapa):

```
#pontos que representam o centro geométrico de um polígono
centroids_freguesias = freguesia_viseu.centroid
centroids_freguesias.plot()
```



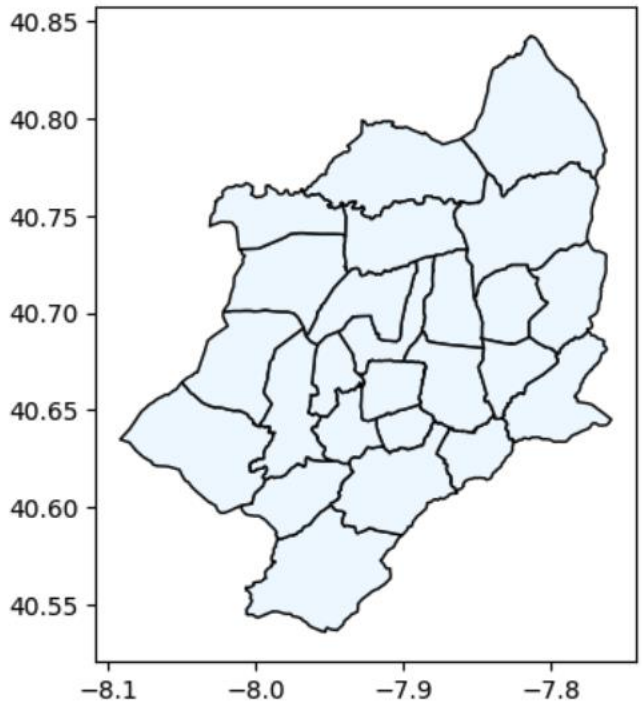
Função Plot() – Cores de fundo e traços

Na função plot() podemos indicar os parâmetros para indicar a cor de fundo dos polígonos e a cor dos traços a volta dos concelhos do distrito, mais especificamente:

- Parâmetro **color** → define a cor de fundo dos polígonos;
- Parâmetro **edgecolor** → define a cor dos traços dos polígonos;

```
#Definir cor de fundo e cor dos traços dos poligonos
freguesia_viseu.plot(color='aliceblue', edgecolor='black')
```

```
<Axes: >
```



No entanto, para além da configuração anterior, podemos combinar este novo mapa com os pontos centroids. Como tal, vamos seguir os seguintes passos:

Passo 1 – Vamos reaproveitar a instrução do plot() anterior e guardar esse código numa variável para depois emparelhar com os centroids. Crie uma nova variável chamada "**distrito**" e coloque o código:

```
#Combinar plot() personalizado do distrito com centroids
distrito = freguesia_viseu.plot(color='aliceblue', edgecolor='black')
```

Passo 2 – Combinar a variável **distrito** dentro variável **centroids_freguesias** (criado anteriormente) e pedir para fazer o plot dos centroids, em que vai necessitar de 3 parâmetros:

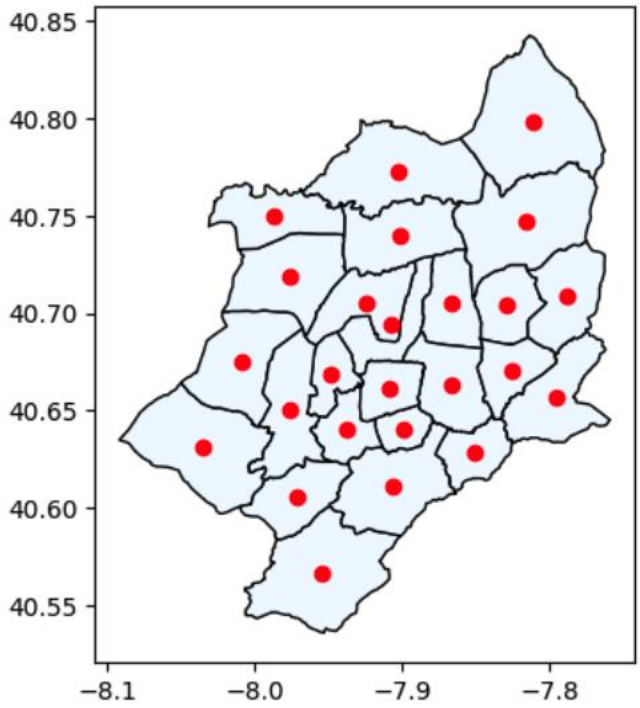
- Parâmetro **ax** → para interligar a variável distrito e combinar o mapa com os centroids;
- Parâmetro **marker** → define o formato do ponto (neste caso colocamos a letra o para simular um círculo;
- Parâmetro **color** → define a cor do ponto (neste caso indicamos a cor vermelho);

```
#combinar a variavel distrito na variavel centroids_freguesias (criado anteriormente)
#e pedir para fazer o plot dos centroids
centroids_freguesias.plot(ax=distrito, marker="o", color="red")
```

Combinação dos 2 passos e resultado:

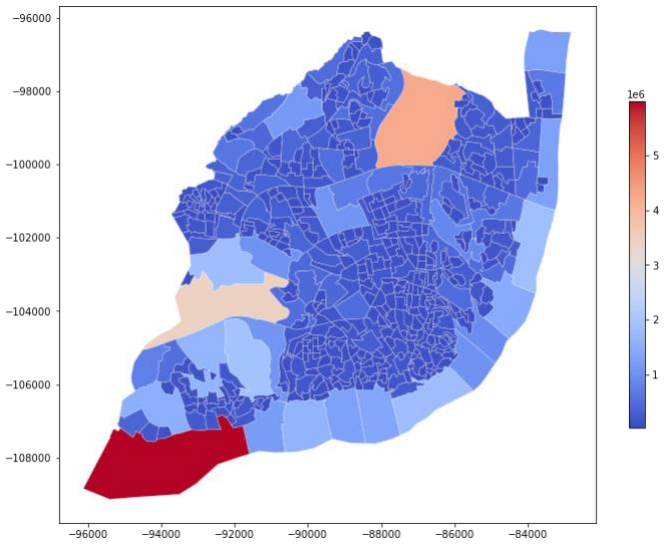
```
#Combinar plot() personalizado do distrito com centroids
distrito = freguesia_viseu.plot(color='aliceblue', edgecolor='black')
#combinar a variavel distrito na variavel centroids_freguesias (criado anteriormente)
#e pedir para fazer o plot dos centroids
centroids_freguesias.plot(ax=distrito, marker="o", color="red")
```

```
<Axes: >
```



Combinação de mapas e dados demográficos

Podemos combinar a utilização de dados demográficos dentro da nossa projeção e criar dados estatísticos visuais, como por exemplo, mostrar um mapa de cores com a densidade populacional e com variações de cores:



Para este objetivo, vamos criar este tipo de visualização para o concelho de viseu e mostrar a densidade populacional por visualização de gráficos.

Para obter a informação dados populacionais, temos o portal do IRN que realizar os Censos. A realização de censos envolve a recolha exaustiva de dados demográficos e de habitação a nível nacional, conduzida pelo Instituto Nacional de Estatística (INE).

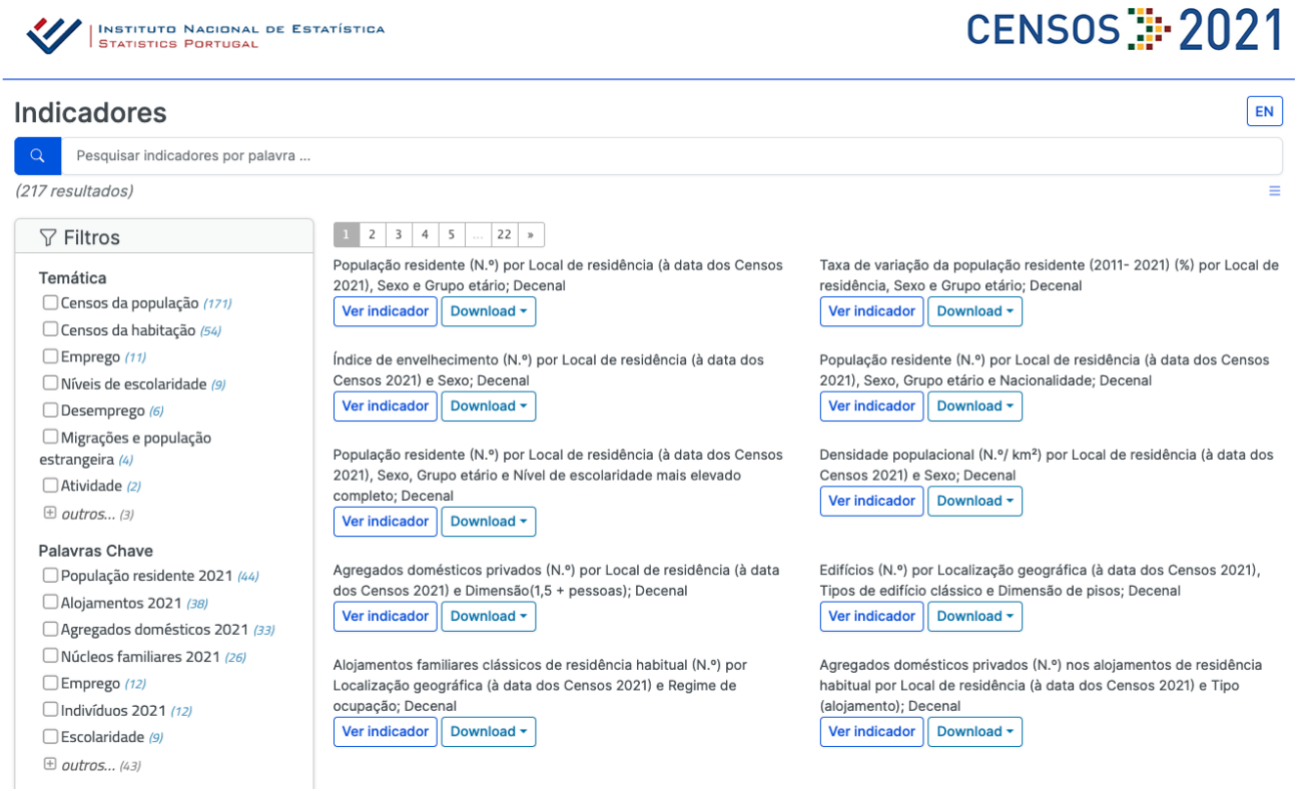
Estes dados são coletados através de um questionário aplicado a todos os indivíduos, famílias, alojamentos e edifícios, com o objetivo de obter uma "fotografia" completa do país num determinado momento.

A recolha pode ser realizada online, por telefone ou presencialmente, e é apoiada por outros organismos como os municípios e as juntas de freguesia.

Para aceder ao portal dos Censos, vamos clicar na seguinte hiperligação: https://censos.ine.pt/

Nesta página, vamos clicar na opção Censos 2021 – Indicadores / Base de dados:

Na página dos indicadores, temos imensas base de dados de informação, tal como demonstra a seguinte imagem:



Para este caso, vai ser utilizado o indicador “População residente (N.º) por Local de residência (à data dos Censos 2021), Sexo e Grupo etário; Decenal” que pode ser consultado na seguinte hiperligação: https://tabulador.ine.pt/censos2021/?lang=PT&c=0011609

Na página, pode descarregar os dados das estatísticas clicando no botão download e temos os ficheiros .xlsx, .ods e .json (neste caso não temos ficheiros .csv).

Para este contexto, foi extraída a informação do distrito de Viseu para obter apenas a população do distrito e respetivos concelhos (**ficheiro**: habitantes_dist_con_Viseu.csv)

Este ficheiro contém a mesma informação que o shapefile só que tem uma última coluna com o número de habitantes que foi extraído do ficheiro do IRN (só que no formato .csv).

A ideia será juntar (merge) a informação deste novo ficheiro com o trabalho que foi feito anteriormente.

Como tal, vamos fazer os seguintes passos:

Passo 1 – Importar o ficheiro **habitantes_dist_con_Viseu.csv** com o **delimitador** da , (virgula)

```
#carregar o ficheiro
data = pd.read_csv("habitantes_dist_con_Viseu.csv", sep=",")
```

Passo 2 – A variável freguesia_viseu tem os dados do distrito e concelhos de Viseu (dados do ficheiro shapefile). No entanto precisamos de fazer a junção da informação dos habitantes que acabamos de importar.

Assim sendo, vamos juntar os dataframes freguesia_viseu com data (e neste apenas selecionar a coluna Habitantes).

Quando fazemos a operação de junção temos de indicar o campo que está em comum nos 2 dataframes para saber emparelhar a informação e só depois o campo que queremos selecionar e guardar toda a informação na variável final:

```
#juntar dataframes freguesia_viseu com data
final = freguesia_viseu.merge(data[["Freguesia", "Habitantes"]], on = "Freguesia")
```

Passo 3 – Agora que já temos as informações todas juntas num único dataframe, vamos pedir para mostrar o gráfico plot() e o fator de projeção vai ser o campo dos Habitantes (pois o plot já vai buscar os dados da coluna “geometry” para desenhar o mapa). Como tal, deve invocar a variável que acabou de criar (final) e indicar 5 parâmetros:

- Parâmetro **column**: permite indicar qual é o campo para usar como base na cor de projeção, neste caso queremos a coluna 'Habitantes';

- Parâmetro **cmap**: permite indicar o esquema de cores a utilizar (podemos indicar 'OrRd' para colocar cores castanhas, ou pode configurar com outro esquema de cor. Como tal, deve consultar a seguinte página e ver qual o esquema a selecionar: https://matplotlib.org/stable/users/explain/colors/colormaps.html);

- Parâmetro **linewidth**: permite definir a grossura da linha que divide os concelhos (neste caso colocamos 0.8);

- Parâmetro **edgecolor**: permite definir o grau de transparência da cor em que 0 não coloca cor na linha divisória e 1 coloca cizento na totalidade (neste caso, vamos colocar '0.8');

- Parâmetro **legend**: permite construir a legenda lateral a informar os dados das cores (neste esquema de cores, quanto mais elevado o número de habitantes mais escuro fica);

```
# Plot do GeoDataFrame usando a coluna 'Habitantes' como base para a cor
final.plot(column='Habitantes', cmap='OrRd', linewidth=0.8, edgecolor='0.8', legend=True)
```

Para complementar a informação do gráfico, não esqueça de dar um título e mostrar o gráfico final:

```
# Adicionar um título ao gráfico
plt.title('Casos por Freguesia')

# Mostrar o gráfico
plt.show()
```

Resultado Final: