Índice

2. Tipos de Dados Geoespaciais

Objetivo: Aprender a manipular arquivos geoespaciais (shapefiles, GeoJSON, raster) e transformá-los para análise.

2.1 Trabalhando com Dados Vetoriais

Os dados vetoriais são representações espaciais baseadas em pontos, linhas e polígonos. Esses dados são amplamente utilizados em aplicações geoespaciais para modelar características do mundo real, como limites políticos, rodovias, rios e infraestrutura urbana.

Um dos formatos mais comuns para armazenar dados vetoriais é o **Shapefile (.shp)**, desenvolvido pela ESRI. Ele é composto por múltiplos arquivos auxiliares (como pode ser visto no diretorio datasets/Brasil_UF):

- **.shp**: Armazena a geometria dos objetos espaciais.
- **.shx**: Índice espacial das geometrias.
- **.dbf**: Tabela de atributos contendo informações descritivas sobre os objetos.
- **.prj**: Define o sistema de coordenadas do arquivo.

Neste notebook, vamos carregar um shapefile contendo os limites das unidades federativas do Brasil e explorar sua estrutura.

In [15]:
```
import geopandas as gpd

#Usando ShapeFile
BASE_DIR = "datasets"
shapefile_path = f"{BASE_DIR}/RJ_2023/RJ_Municipios_2023.shp"  # Shapefile das U
gdf_shp = gpd.read_file(shapefile_path)

gdf_shp.head()
```

Out[15]:

	CD_MUN	NM_MUN	CD_RGI	NM_RGI	CD_RGINT	NM_RGINT	CD_UF	NM_UF	CD
0	3300100	Angra dos Reis	330002	Angra dos Reis	3301	Rio de Janeiro	33	Rio de Janeiro	
1	3300159	Aperibé	330012	Santo Antônio de Pádua	3304	Campos dos Goytacazes	33	Rio de Janeiro	
2	3300209	Araruama	330013	Cabo Frio	3305	Macaé - Rio das Ostras - Cabo Frio	33	Rio de Janeiro	
3	3300225	Areal	330007	Petrópolis	3303	Petrópolis	33	Rio de Janeiro	
4	3300233	Armação dos Búzios	330013	Cabo Frio	3305	Macaé - Rio das Ostras - Cabo Frio	33	Rio de Janeiro	

Outro formato bastante utilizado é o **GeoJSON (.geojson)**, que é baseado em JSON e amplamente adotado devido à sua compatibilidade com aplicações web. A maior diferença entre esses dataframes e um dataframe comum é a coluna geometry, que contém objetos espaciais do tipo POLYGON ou MULTIPOLYGON que representam áreas geográficas com coordenadas de latitude e longitude.

In [16]:
```
#Usando GeoJson
geojson_path = f"{BASE_DIR}/br_states.json"
gdf = gpd.read_file(geojson_path)

gdf.head()
```

Out[16]:

	id	FID_Export	SIGLA	Total	Homens	Mulheres	Urbana	Rural	TX_Alfab
0	AC	0	AC	557526	280983	276543	370267	187259	65.154545
1	AL	1	AL	2822621	1378942	1443679	1919739	902882	59.364356
2	AM	2	AM	2812557	1414367	1398190	2107222	705335	72.250000
3	AP	3	AP	477032	239453	237579	424683	52349	83.650000
4	BA	4	BA	13070250	6462033	6608217	8772348	4297902	71.163373

Exibir informações gerais do GeoDataFrame

In [17]:
```
gdf_shp.info()
print("\n-----\n")
gdf_shp.crs # Verificar a projeção cartográfica
```

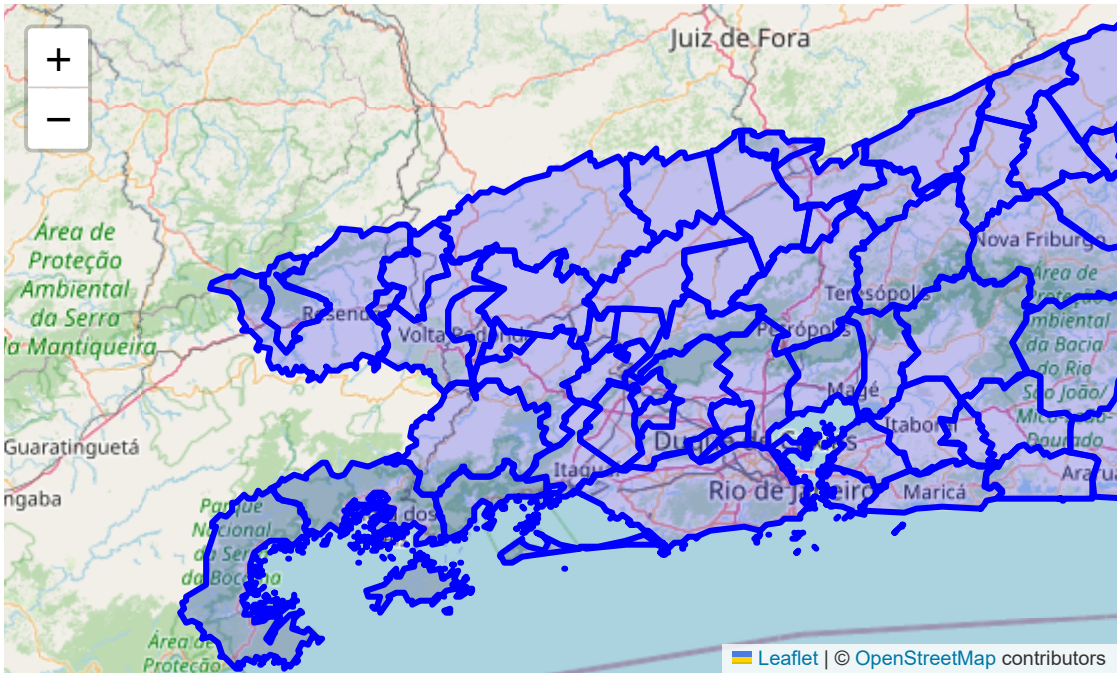
```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 92 entries, 0 to 91
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   CD_MUN      92 non-null     object  
 1   NM_MUN      92 non-null     object  
 2   CD_RGI      92 non-null     object  
 3   NM_RGI      92 non-null     object  
 4   CD_RGINT    92 non-null     object  
 5   NM_RGINT    92 non-null     object  
 6   CD_UF       92 non-null     object  
 7   NM_UF       92 non-null     object  
 8   CD_REGIAO   92 non-null     object  
 9   NM_REGIAO   92 non-null     object  
 10  CD_CONCURB  49 non-null     object  
 11  NM_CONCURB  49 non-null     object  
 12  AREA_KM2    92 non-null     float64 
 13  geometry    92 non-null     geometry
dtypes: float64(1), geometry(1), object(12)
memory usage: 10.2+ KB

-----
```

Out[17]:
```
<Geographic 2D CRS: EPSG:4674>
Name: SIRGAS 2000
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: Latin America - Central America and South America - onshore and offshor
e. Brazil - onshore and offshore.
- bounds: (-122.19, -59.87, -25.28, 32.72)
Datum: Sistema de Referencia Geocentrico para las AmericaS 2000
- Ellipsoid: GRS 1980
- Prime Meridian: Greenwich
```

Visualizacao do shapefile usando Folium

In [18]:
```
import folium

# Mapa centralizado no Brasil
mapa = folium.Map(location=[-22.9068, -43.1729], zoom_start=8)

# Adicionando os limites das unidades federativas como o GeoJson
for _, row in gdf_shp.iterrows():
    folium.GeoJson(
        row["geometry"],
        name=row["NM_MUN"],
        tooltip=str(row["AREA_KM2"]) + " km²",
        style_function=lambda x: {"color": "blue"},
    ).add_to(mapa)


mapa
```

Out[18]:



2.2 Trabalhando com Dados Raster

Os dados raster representam superfícies contínuas e são compostos por uma grade regular de células (pixels). Eles são usados para representar imagens de satélite, modelos

digitais de elevação (MDE) e outras informações geoespaciais. Cada pixel contém um valor que representa uma característica da superfície.

Como os dados raster funcionam como um bitmap (ou seja, uma matriz de valores organizados em células/pixels), eles permitem representar informações contínuas sobre superfícies geográficas. O formato TIFF (Tagged Image File Format), especialmente na sua variação GeoTIFF, é amplamente utilizado para armazenar dados raster geoespaciais. Ele já vem com metadados embutidos, o que facilita a extração de informações espaciais e a integração com ferramentas de análise geográfica.

In [19]:
```
import rasterio


arquivo_raster = f"{BASE_DIR}/World_Temp_geotiff/data/TEMP.tif"

# Carrega um arquivo raster da temperatura global
with rasterio.open(arquivo_raster) as src:
    print(f"Formato: {src.driver}")
    print(f"Dimensões (Linhas, Colunas): {src.height}, {src.width}")
    print(f"Número de bandas: {src.count}")
    print(f"Sistema de Coordenadas: {src.crs}")
```

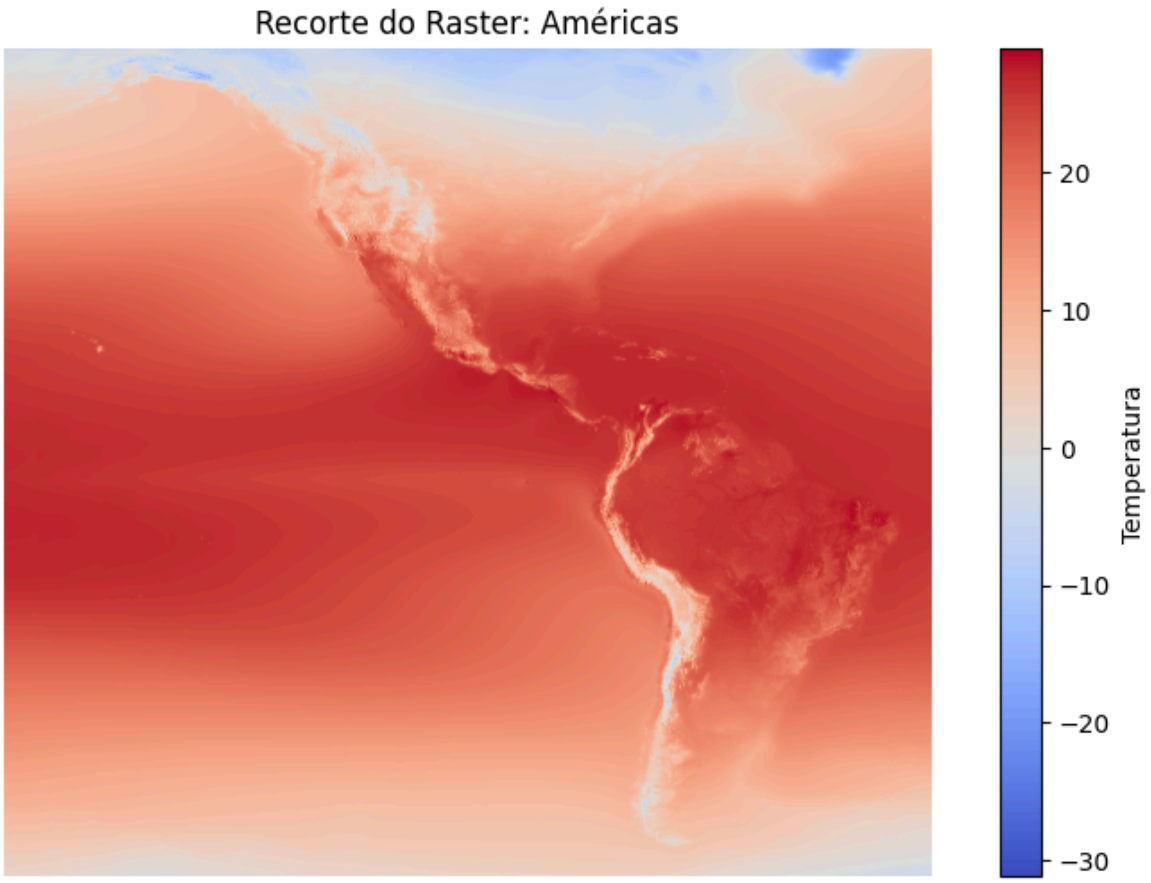
```
Formato: GTiff
Dimensões (Linhas, Colunas): 15000, 43200
Número de bandas: 1
Sistema de Coordenadas: EPSG:4326
```

Extraindo e Visualizando os Dados

Os dados raster geralmente possuem múltiplas bandas (camadas). Vamos visualizar a primeira banda da imagem:

In [20]:
```
import rasterio
from rasterio.windows import from_bounds
import matplotlib.pyplot as plt

# Define os limites aproximados das Américas (lon/lat)
# América do Norte até América do Sul
lon_min, lon_max = -170, -30  # de Alaska até o Nordeste do Brasil
lat_min, lat_max = -60, 80    # da Patagônia até o Ártico


with rasterio.open(arquivo_raster) as src:
    window = from_bounds(
        lon_min, lat_min, lon_max, lat_max,
        transform=src.transform
    )
    americas = src.read(1, window=window)
    americas_transform = src.window_transform(window)

# Plota o recorte das Américas
plt.figure(figsize=(10, 6))
plt.imshow(americas, cmap='coolwarm')
plt.colorbar(label="Temperatura")
plt.title("Recorte do Raster: Américas")
```

```
plt.axis('off')
plt.show()
```



Estatísticas Básicas do Raster

A partir dessa matriz ja é possivel extrair algumas informacoes que podem ser interessentes

In [21]:
```
print(f"Mínima Temperatura: {np.min(raster_data)}")
print(f"Máxima Temperatura: {np.max(raster_data)}")
print(f"Temperatura Média: {np.mean(raster_data)}")
print(f"Desvio Padrao: {np.std(raster_data)}")
```

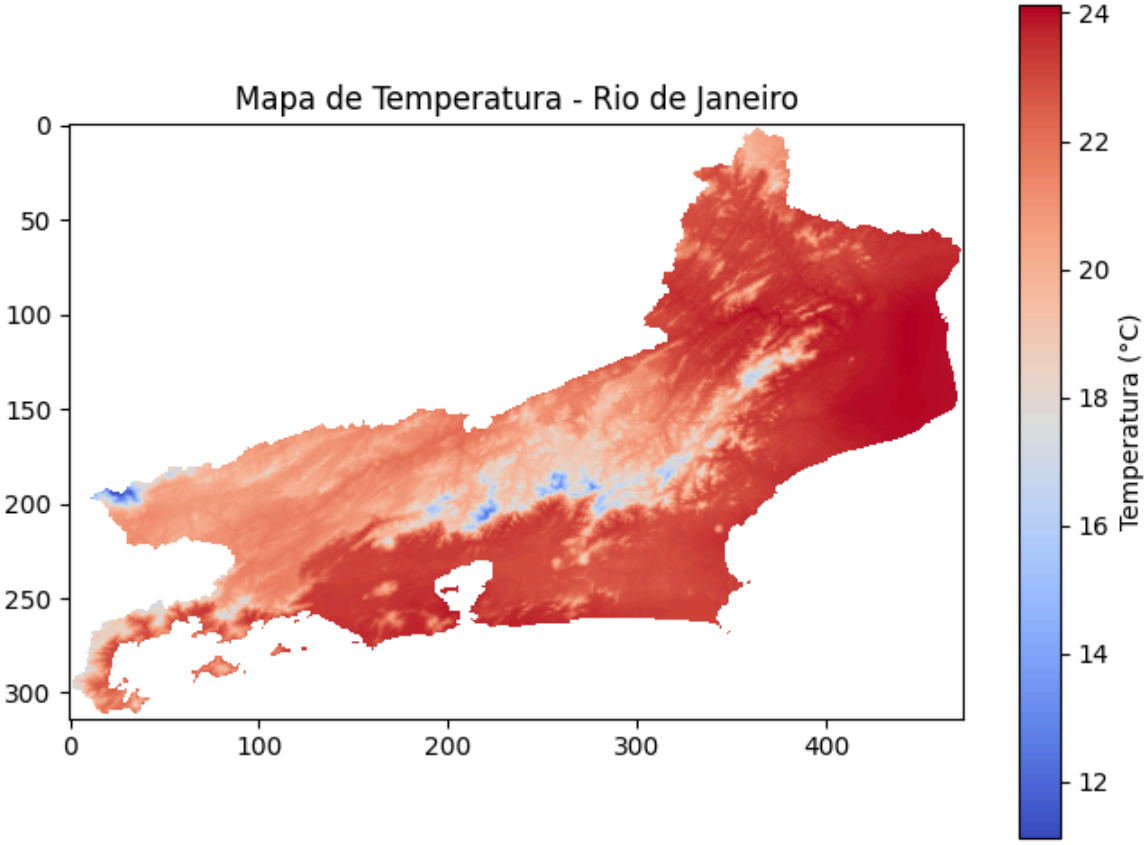
```
Mínima Temperatura: 0.25
Máxima Temperatura: 31.0625
Temperatura Média: 23.23480796813965
Desvio Padrao: 3.050732374191284
```

Recorte de Raster Usando um Shapefile (Rio de Janeiro)

É comum precisar recortar um raster para uma área de interesse usando um shapefile (usaremos o shp do Rio utilizado na Secao 1.1). Podemos fazer isso com rasterio.mask:

In [22]:
```
from rasterio.mask import mask

# Carregar o shapefile
rio = gpd.read_file(shapefile_path)

# Abrir o raster e recortar com base na geometria do shapefile
with rasterio.open(arquivo_raster) as src:
    rio = rio.to_crs(src.crs)  # Garantir que ambos têm o mesmo sistema de coord
```

```
    out_image, out_transform = mask(src, rio.geometry, crop=True, filled=True, n
    out_meta = src.meta.copy()
    out_meta.update({
        "driver": "GTiff",
        "height": out_image.shape[1],
        "width": out_image.shape[2],
        "transform": out_transform,
        "nodata": np.nan
    })

# Substituir zeros por NaN
out_image = out_image.astype('float32')
out_image[out_image == 0] = np.nan

# Exibir o raster recortado (Mapa de Temperatura)
plt.figure(figsize=(8, 6))
plt.imshow(out_image[0], cmap="coolwarm")
plt.colorbar(label="Temperatura (°C)")
plt.title("Mapa de Temperatura - Rio de Janeiro")
plt.show()
```

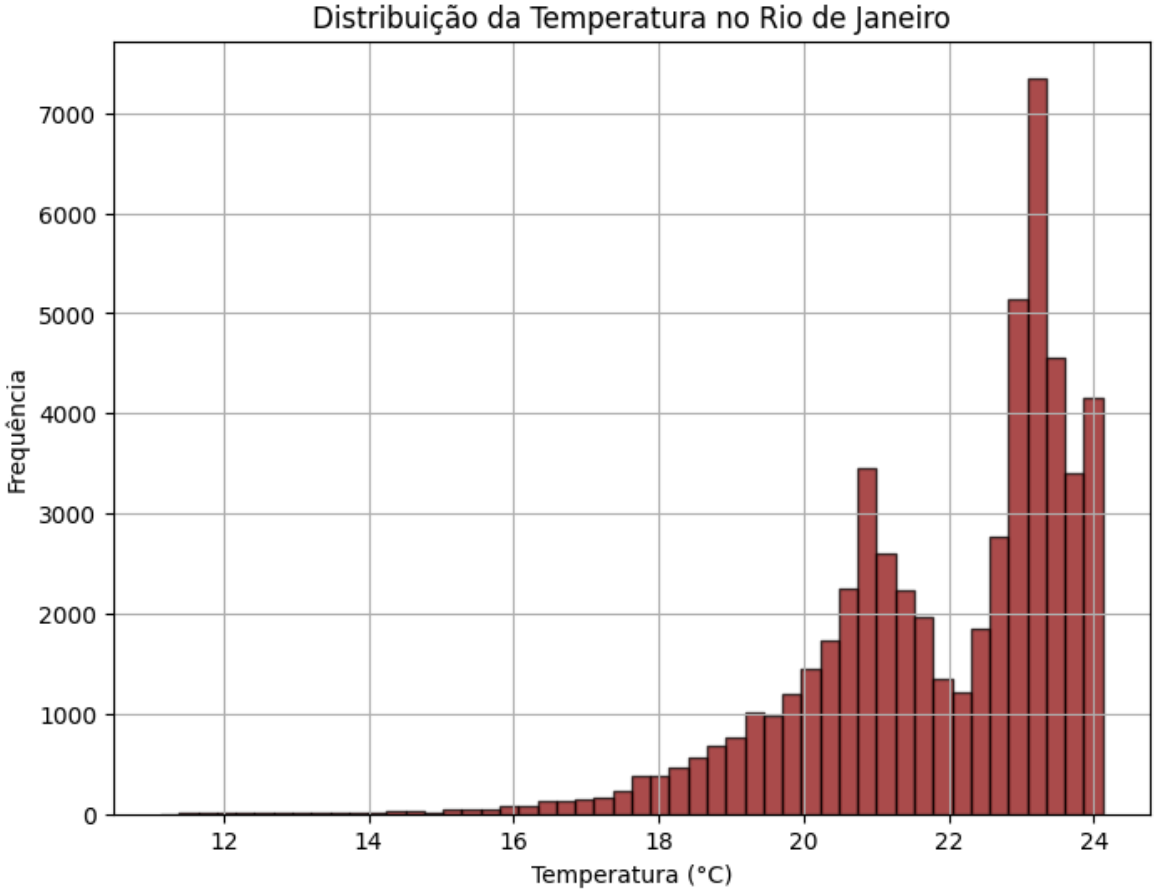


In [23]:
```
import numpy as np

# Filtrar apenas os valores de temperatura válidos
valid_pixels = out_image[0][out_image[0] > 0]

# Cálculo das estatísticas
temp_min = np.min(valid_pixels)
temp_max = np.max(valid_pixels)
temp_mean = np.mean(valid_pixels)
temp_std = np.std(valid_pixels)

print(f"Estatísticas de Temperatura do Rio de Janeiro:")
print(f"Temperatura mínima: {temp_min:.2f} °C")
```

```
print(f"Temperatura máxima: {temp_max:.2f} °C")
print(f"Temperatura média: {temp_mean:.2f} °C")
print(f"Desvio padrão: {temp_std:.2f} °C")
```

```
Estatísticas de Temperatura do Rio de Janeiro:
Temperatura mínima: 11.12 °C
Temperatura máxima: 24.12 °C
Temperatura média: 21.90 °C
Desvio padrão: 1.79 °C
```

In [24]:
```
plt.figure(figsize=(8, 6))
plt.hist(valid_pixels, bins=50, color="darkred", alpha=0.7, edgecolor="black")
plt.xlabel("Temperatura (°C)")
plt.ylabel("Frequência")
plt.title("Distribuição da Temperatura no Rio de Janeiro")
plt.grid(True)
plt.show()
```



Salvando um Novo Raster

In [25]:
```
with rasterio.open(
    "novo_raster.tif",
    "w",
    driver="GTiff",
    height=out_image.shape[1],
    width=out_image.shape[2],
    count=1,
    dtype=out_image.dtype,
    crs=src.crs,
    transform=out_transform,
) as dest:
    dest.write(out_image[0], 1)
```

2.3 Raster ou Vetorial

Nesta seção, vamos entender a diferença entre os dois formatos mais usados em geociência:

- **Raster**: representações contínuas em grade de pixels (como imagens)
- **Vetorial**: representações geométricas precisas (pontos, linhas, polígonos)

Além da explicação teórica, vamos visualizar a diferença usando dados reais.

📊 Comparativo entre Raster e Vetorial

Tipo de dado	Representação	Estrutura	Exemplo
Raster	Grade de pixels (imagem)	Matriz regular (grid)	Temperatura, altitude, satélite
Vetorial	Pontos, linhas e polígonos	Tabelado + geometria	Localização de imóveis, limites de bairros