Introdução a Análise de Dados com Python

Joyce Teixeira - jvt@cesar.school

Pandas

C.e.S.A.R school


PANDAS LOVE DATA SCIENCE
imgflip.com

Pandas

- O Pandas é uma biblioteca de código aberto em Python projetada para análise e manipulação de dados.
- Pandas é amplamente utilizado em ciência de dados, análise financeira, economia e outras áreas devido à sua eficácia em lidar com dados tabulares.
- Pandas simplifica a leitura, limpeza, transformação e análise de dados, economizando tempo e tornando a análise de dados mais acessível.

Pandas

- Existem dois tipos principais de estruturas de dados no pandas:
  - **Series** é uma estrutura de dados unidimensional que pode conter dados de diferentes tipos. É semelhante a uma coluna em uma planilha ou a um array NumPy.
  - **DataFrames** são estruturas bidimensionais semelhantes a tabelas de banco de dados. Eles são a estrutura de dados mais usada no Pandas e consistem em colunas com nomes e tipos de dados.

Series

Series

- Características principais
  - Homogeneidade: Todas as informações em uma Series compartilham o mesmo tipo de dado, o que a torna eficaz para armazenar dados de uma única variável.
  - Rótulos de índice: Cada elemento na Series é associado a um rótulo de índice, permitindo um acesso fácil e intuitivo aos dados.
  - Facilidade de manipulação: As Series oferecem métodos e funções poderosas para realizar operações, filtragens e transformações em seus dados.
  - Dados faltantes: As Series podem acomodar dados faltantes, identificados como NaN (Not a Number).
  - Flexibilidade: Apesar da homogeneidade, as Series podem armazenar dados de diferentes tipos (inteiros, flutuantes, strings, datas, etc.). Essa flexibilidade permite que você trabalhe com diversos tipos de dados em um mesmo conjunto.

Hands-on Tour!

- Criando uma Series

```
[1]  # Importando bibliotecas
     import pandas as pd

     # Criando uma Series (registros de 7 dias)
     avistamentos = pd.Series([10, 15, 8, 20, 12, 18, 9])
     print(avistamentos)
```

```
0    10
1    15
2     8
3    20
4    12
5    18
6     9
dtype: int64
```

C.e.S.A.R school

Series

```
# Selecionando apenas os valores
print(avistamentos.values)

# Selecionando apenas os índices
print(avistamentos.index)
```

```
[10 15  8 20 12 18  9]
RangeIndex(start=0, stop=7, step=1)
```

Series

- Como ao criar a Series não demos um índice específico o pandas usou os inteiros positivos crescentes como padrão.

```
[1]  # Importando bibliotecas
     import pandas as pd

     # Criando uma Series (registros de 7 dias)
     avistamentos = pd.Series([10, 15, 8, 20, 12, 18, 9])
     print(avistamentos)
```

```
0    10
1    15
2     8
3    20
4    12
5    18
6     9
dtype: int64
```

C.E.S.A.R school

Series

- Pode ser conveniente atribuirmos um índice diferente do padrão.
- Poderíamos atribuir nomes ao index:

```
# Criando índices personalizados
avistamentos = pd.Series([10, 15, 8, 20, 12, 18, 9], index=["Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado", "Domingo"])
print(avistamentos)
```

```
Segunda    10
Terça      15
Quarta      8
Quinta     20
Sexta      12
Sábado     18
Domingo     9
dtype: int64
```

Series

- O index nos ajuda para referenciar um determinado valor, ele nos permite acessar os valores pelo seu rótulo:

```
# Acessando apenas o registro da Sexta
print(avistamentos["Sexta"])
```

```
12
```

Series

- Selecionando os 5 primeiros registros com head() e os 5 últimos com tail()

```
# Selecionando os registros que representam os 5 primeiros dias
primeiros_dias = avistamentos.head()
print(primeiros_dias)
```

```
Segunda    10
Terça      15
Quarta      8
Quinta     20
Sexta      12
dtype: int64
```

```
# Mostrando os últimos 5 registros
ultimos_dias = avistamentos.tail()
print(ultimos_dias)
```

```
Quarta      8
Quinta     20
Sexta      12
Sábado     18
Domingo     9
dtype: int64
```

Series

- Outra facilidade proporcionada pela estrutura são seus métodos que fornecem informações estatísticas sobre os valores:
  - média: mean()
  - desvio padrão: std()

```
# Calculando a média e o desvio padrão de avistamentos em uma semana
media_avistamentos = avistamentos.mean()
std_avistamentos = avistamentos.std()
print(f"Média de avistamentos na primeira semana: {media_avistamentos:.2f}")
print(f"Desvio padrão de avistamentos na primeira semana: {std_avistamentos:.2f}")
```

```
Média de avistamentos na primeira semana: 13.14
Desvio padrão de avistamentos na primeira semana: 4.63
```

Series

- O método .describe() em uma Series do Pandas fornece estatísticas descritivas resumidas sobre os dados contidos na Series. Ele calcula várias medidas estatísticas que podem ajudar a entender a distribuição dos valores. As estatísticas incluídas na saída do .describe() são as seguintes:
  - count: O número de elementos na Series (ou seja, o tamanho da amostra).
  - mean: A média dos valores na Series.
  - std: O desvio padrão, que mede o quanto os valores tendem a se afastar da média. Valores maiores indicam maior dispersão nos dados.
  - min: O valor mínimo na Series.
  - 25%: O primeiro quartil (Q1) ou 25º percentil. Isso representa o valor abaixo do qual 25% dos dados estão.
  - 50%: A mediana, que é o valor do meio quando os dados são ordenados. Também é chamada de segundo quartil (Q2) ou mediana.
  - 75%: O terceiro quartil (Q3) ou 75º percentil. Isso representa o valor abaixo do qual 75% dos dados estão.
  - max: O valor máximo na Series.

Series

```
# Observando estatísticas descritivas sobre os dados
print(avistamentos.describe())
```

```
count     7.000000
mean     13.142857
std       4.634241
min       8.000000
25%       9.500000
50%      12.000000
75%      16.500000
max      20.000000
dtype: float64
```

Series

- Lidando com valores NaN

```
# Importando numpy para inserir valores NaN
import numpy as np

# Simulando a criação de Séries com valores NaN
avistamentos = pd.Series([10, 15, np.nan, 20, np.nan, 18, 9], index=["Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado", "Domingo"])

# Exiba a Series original
print("Series original:")
print(avistamentos)

# Preencha os valores NaN com zero
avistamentos_completos = avistamentos.fillna(0)

# Exiba a Series com os valores NaN preenchidos com zero
print("\nSeries com valores NaN preenchidos:")
print(avistamentos_completos)
```

```
Series original:
Segunda    10.0
Terça      15.0
Quarta      NaN
Quinta     20.0
Sexta       NaN
Sábado     18.0
Domingo     9.0
dtype: float64

Series com valores NaN preenchidos:
Segunda    10.0
Terça      15.0
Quarta      0.0
Quinta     20.0
Sexta       0.0
Sábado     18.0
Domingo     9.0
dtype: float64
```

Pandas + numpy

- A estrutura é flexível o suficiente para aplicarmos algumas expressões matemáticas e funções matemáticas do numpy diretamente:

```
print(avistamentos_completos ** 2)
print(np.log(avistamentos_completos))
```

```
Segunda    100.0
Terça      225.0
Quarta       0.0
Quinta     400.0
Sexta        0.0
Sábado     324.0
Domingo     81.0
dtype: float64
Segunda    2.302585
Terça      2.708050
Quarta          -inf
Quinta     2.995732
Sexta           -inf
Sábado     2.890372
Domingo    2.197225
dtype: float64
```

Series

- Seleção

```
# Selecionando dias com avistamentos acima de 15
dias_significativos = avistamentos[avistamentos > 15]
print("Dias com avistamentos significativos:")
print(dias_significativos)
```

```
Dias com avistamentos significativos:
Quinta    20.0
Sábado    18.0
dtype: float64
```

DataFrame

DataFrame

- Os DataFrames são uma estrutura de dados bidimensional no Pandas.
- Consistem em linhas e colunas, permitindo o armazenamento de dados em formato de tabela.
- Cada coluna em um DataFrame é uma Series.
- Os DataFrames também possuem rótulos de índice para linhas e nomes de coluna.
- São especialmente úteis para a manipulação de dados tabulares, como planilhas ou bancos de dados.

DataFrame

- Criaremos um DataFrame que possui valores de diferentes tipos, usando um dicionário como entrada dos dados.

```
import pandas as pd

# Dados dos restaurantes
dados = {'Restaurante': ["Mamma Mia", "Sabor Oriental", "La Pizzeria", "Chef Estrelado", "Cantina Aconchego", "Pizza Point"],
         'Culinária': ["Italiana", "Chinesa", "Italiana", "Francesa", "Italiana", "Italiana"],
         'Avaliação': [4.8, 4.5, 4.9, 4.2, 4.7, 3.9],
         'Preço Médio': [40, 25, 35, 150, 30, 25]}

# Criando um DataFrame
df = pd.DataFrame(dados)
print(df)
```

```
         Restaurante Culinária  Avaliação  Preço Médio
0          Mamma Mia  Italiana        4.8           40
1     Sabor Oriental   Chinesa        4.5           25
2        La Pizzeria  Italiana        4.9           35
3     Chef Estrelado  Francesa        4.2          150
4  Cantina Aconchego  Italiana        4.7           30
5        Pizza Point  Italiana        3.9           25
```

DataFrame

- Os tipos de dados que compõem as colunas podem ser verificados por um método próprio:

```
# Verificando os tipos de dados que compõem as colunas

print(df.dtypes)
```

```
Restaurante     object
Culinária       object
Avaliação      float64
Preço Médio      int64
dtype: object
```

DataFrame

- A quantidade de linhas e colunas pode ser visualizada com o método .shape

```
# Verificando linhas e colunas

print(df.shape)
```

```
(6, 4)
```

DataFrame

- É possível acessar a lista de colunas de forma bem intuitiva:

```
# Acessando a lista de colunas
print(df.columns)
```

```
Index(['Restaurante', 'Culinária', 'Avaliação', 'Preço Médio'], dtype='object')
```

DataFrame

- Selecionando os 5 primeiros registros com head() e os 5 últimos com tail()

```
# Exibindo as 5 primeiras linhas do DataFrame
print(df.head())
```

```
         Restaurante Culinária  Avaliação  Preço Médio
0          Mamma Mia  Italiana        4.8           40
1     Sabor Oriental   Chinesa        4.5           25
2        La Pizzeria  Italiana        4.9           35
3     Chef Estrelado  Francesa        4.2          150
4  Cantina Aconchego  Italiana        4.7           30
```

```
# Exibindo as 5 últimas linhas do DataFrame
print(df.tail())
```

```
         Restaurante Culinária  Avaliação  Preço Médio
1     Sabor Oriental   Chinesa        4.5           25
2        La Pizzeria  Italiana        4.9           35
3     Chef Estrelado  Francesa        4.2          150
4  Cantina Aconchego  Italiana        4.7           30
5        Pizza Point  Italiana        3.9           25
```

DataFrame

- Os nomes das colunas podem ser usadas para acessar seus valores:

```
# Acessando uma coluna a partir do seu nome
print(df["Restaurante"])
```

```
0            Mamma Mia
1       Sabor Oriental
2          La Pizzeria
3       Chef Estrelado
4    Cantina Aconchego
5          Pizza Point
Name: Restaurante, dtype: object
```

DataFrame

- describe() também é uma boa forma de verificar resumidamente a disposição estatística dos dados numéricos:

```
# Verificando a disposição estatística dos dados numéricos
print(df.describe())
```

```
       Avaliação  Preço Médio
count   6.000000     6.000000
mean    4.500000    50.833333
std     0.384708    48.930222
min     3.900000    25.000000
25%     4.275000    26.250000
50%     4.600000    32.500000
75%     4.775000    38.750000
max     4.900000   150.000000
```

DataFrame

- Outra facilidade proporcionada pela estrutura são seus métodos que fornecem informações estatísticas sobre os valores:
  - média: mean()
  - desvio padrão: std()

```
# Calculando a média das avaliações
media_avaliacoes = df['Avaliação'].mean()
# Calculando o desvio padrão das avaliações
std_avaliacoes = df['Avaliação'].std()
print(media_avaliacoes)
print(std_avaliacoes)
```

```
4.5
0.38470768123342697
```

DataFrame

- Para selecionar de acordo com critérios condicionais, se usa o que se chama de Boolean Indexing.
  - EX: Selecionar apenas as linhas em que o valor da coluna Avaliação seja acima da média de avaliações.

```
# Filtrando restaurantes com avaliações acima da média
restaurantes_acima_media = df[df['Avaliação'] > media_avaliacoes]
print(restaurantes_acima_media)
```

```
         Restaurante Culinária  Avaliação  Preço Médio
0          Mamma Mia  Italiana        4.8           40
2       La Pizzeria   Italiana        4.9           35
4  Cantina Aconchego  Italiana        4.7           30
```

DataFrame

- Este tipo de indexação também possibilita checar condições de múltiplas colunas.
- Diferentemente do que estamos habituados em Python, aqui se usam operadores bitwise, ou seja, &, |, ~ ao invés de and, or, not, respectivamente.
    - EX: Suponha que além de df["Avaliação"] > média queiramos que o valor da coluna Preço Médio não seja maior que 32:

```
# Filtrando restaurantes com avaliações acima da média e preço médio abaixo de R$32,00
restaurantes_acima_media = df[(df['Avaliação'] > media_avaliacoes) & (df['Preço Médio'] < 32)]
print(restaurantes_acima_media)
```

```
         Restaurante Culinária  Avaliação  Preço Médio
4  Cantina Aconchego  Italiana        4.7           30
```

DataFrame

- Muitas vezes é necessário selecionarmos valores específicos de um DataFrame, seja uma linha ou uma célula específica.

DataFrame

```
# Acessando as informações do restaurante Chef Estrelado
print(df[df['Restaurante'] == 'Chef Estrelado'])
```

```
      Restaurante Culinária  Avaliação  Preço Médio
3  Chef Estrelado  Francesa        4.2          150
```

```
# Queremos exibir apenas o preço médio do restaurante acessado
print(df[df['Restaurante'] == 'Chef Estrelado']['Preço Médio'])
print(df[df['Restaurante'] == 'Chef Estrelado']['Preço Médio'].values)
```

```
3    150
Name: Preço Médio, dtype: int64
[150]
```

```
# Encontrando o preço do restaurante "Chef Estrelado"
preco_chef_estrelado = df[df['Restaurante'] == 'Chef Estrelado']['Preço Médio'].values[0]
print(preco_chef_estrelado)
```

```
150
```

DataFrame

- Outra tarefa comum aplicada em DataFrames é ordená-los por determinada coluna:

```
# Classificando o menu por avaliação
print(df.sort_values(by='Avaliação', ascending=False))
```

```
         Restaurante Culinária  Avaliação  Preço Médio
2        La Pizzeria  Italiana        4.9           35
0          Mamma Mia  Italiana        4.8           40
4  Cantina Aconchego  Italiana        4.7           30
1     Sabor Oriental   Chinesa        4.5           25
3     Chef Estrelado  Francesa        4.2          150
5        Pizza Point  Italiana        3.9           25
```

DataFrame

- Usar o método sort_values não modifica o nosso DataFrame original:

```
# Verificando o dataframe após o sort_values
print(df)
```

```
         Restaurante Culinária  Avaliação  Preço Médio
0          Mamma Mia  Italiana        4.8           40
1     Sabor Oriental   Chinesa        4.5           25
2        La Pizzeria  Italiana        4.9           35
3     Chef Estrelado  Francesa        4.2          150
4  Cantina Aconchego  Italiana        4.7           30
5        Pizza Point  Italiana        3.9           25
```

DataFrame

- Selecionando com o uso de métodos estatísticos.

```
# Encontrando o restaurante com a avaliação mais alta
melhor_restaurante = df[df['Avaliação'] == df['Avaliação'].max()]
print(melhor_restaurante)
```

```
   Restaurante Culinária  Avaliação  Preço Médio
2  La Pizzeria  Italiana        4.9           35
```

```
# Exibindo apenas o nome
melhor_restaurante = df[df['Avaliação'] == df['Avaliação'].max()]['Restaurante'].values[0]
print(melhor_restaurante)
```

```
La Pizzeria
```

Leitura de Dados

Leitura de Dados

- O pandas nos fornece uma série de funcionalidades de leitura de dados, para os mais diversos formatos estruturais de dados.
- Experimente a *autocomplete* de pd.read_<Ctrl + espaço>, entre eles estão:
  - pd.read_csv(): Lê dados de arquivos CSV.
  - pd.read_excel(): Lê dados de planilhas do Excel.
  - pd.read_json(): Lê dados de arquivos JSON.
  - pd.read_sql(): Lê dados de bancos de dados SQL.
  - pd.read_html(): Lê dados de tabelas HTML em páginas da web.
  - entre outros...

Ler CSV

- Arquivo obtido em: http://dados.recife.pe.gov.br/
  - Ficha dos Imóveis da Prefeitura do Recife

Ler CSV

```
# Importando a partir de uma biblioteca
from google.colab import files

uploaded = files.upload()
```

Escolher arquivos | Nenhum arquivo escolhido | Cancel upload

```
# Importando a partir de uma biblioteca
from google.colab import files

uploaded = files.upload()
```

Escolher arquivos imoveis.csv

- **imoveis.csv**(text/csv) - 1641988 bytes, last modified: 18/10/2023 - 100% done

```
Saving imoveis.csv to imoveis.csv
```

C.E.S.A.R school

Ler CSV

```
# Lendo arquivo csv
df_csv = pd.read_csv("imoveis.csv", sep=";")
df_csv
```

	ultimaAtualizacaoCadastral	situacao	sequencialImovel	proprietarioPrincipal	natureza	inscricaoImobiliaria
0	02/12/2015	Ativo	1031171	MUNICIPIO DO RECIFE	Territorial	1.1410.190.02.0025.0000-6
1	29/11/2015	Ativo	1032984	MUNICIPIO DO RECIFE	Territorial	1.1410.140.01.0730.0000-0
2	29/11/2015	Ativo	1049160	MUNICIPIO DO RECIFE	Predial	1.1426.030.01.0170.0000-3
3	29/11/2015	Ativo	1050834	MUNICIPIO DO RECIFE	Predial	1.1426.043.01.0379.0000-4
4	29/11/2015	Ativo	1063391	MUNICIPIO DO RECIFE	Territorial	1.1445.155.02.0670.0000-4
...	...	...	...	...	...	...
3090	29/11/2015	Ativo	7777400	MUNICIPIO DO RECIFE	Predial	6.1710.069.02.0251.0319-2

Ler CSV

```
# Para acessar o Google Drive, você pode montá-lo usando o 'google.colab'
from google.colab import drive

# Montar o Google Drive
drive.mount('/content/drive')
```

```
# Agora que o Google Drive está montado, você pode acessar os arquivos
caminho_arquivo = '/content/drive/My Drive/imoveis.csv'
df_drive = pd.read_csv(caminho_arquivo, sep=";")
df_drive.head()
```

	ultimaAtualizacaoCadastral	situacao	sequencialImovel	proprietarioPrincipal	natureza	inscricaoImobiliaria	enderecoImovelNomeLogradouro	enderecoImovelN
0	02/12/2015	Ativo	1031171	MUNICIPIO DO RECIFE	Territorial	1.1410.190.02.0025.0000-6	RUA PAULINO GOMES DE SOUZA Q-PRACA	
1	29/11/2015	Ativo	1032984	MUNICIPIO DO RECIFE	Territorial	1.1410.140.01.0730.0000-0	RUA DO FUTURO Q-AREA L-D	
2	29/11/2015	Ativo	1049160	MUNICIPIO DO RECIFE	Predial	1.1426.030.01.0170.0000-3	RUA DR JOAQUIM ARRUDA FALCAO	
3	29/11/2015	Ativo	1050834	MUNICIPIO DO RECIFE	Predial	1.1426.043.01.0379.0000-4	RUA SACADURA CABRAL	
4	29/11/2015	Ativo	1063391	MUNICIPIO DO RECIFE	Territorial	1.1445.155.02.0670.0000-4	RUA CEL VIRGILIO DE MEDEIROS Q-A L-12	

5 rows × 63 columns

C.E.S.A.R school

Leitura de Arquivos

- Para visualizar sucintamente as primeiras linhas de um DataFrame existe o método **head()**

```
In [25]: df.head()
```

Out[25]:

	ultimaAtualizacaoCadastral	situacao	sequencialImovel	proprietarioPrincipal	natureza	inscricaoImobiliaria	enderecoImov
0	02/12/2015	Ativo	1031171	MUNICIPIO DO RECIFE	Territorial	1.1410.190.02.0025.0000-6	RUA PAULINC
1	29/11/2015	Ativo	1032984	MUNICIPIO DO RECIFE	Territorial	1.1410.140.01.0730.0000-0	RUA DO F
2	29/11/2015	Ativo	1049160	MUNICIPIO DO RECIFE	Predial	1.1426.030.01.0170.0000-3	RUA DF
3	29/11/2015	Ativo	1050834	MUNICIPIO DO RECIFE	Predial	1.1426.043.01.0379.0000-4	RUA S
4	29/11/2015	Ativo	1063391	MUNICIPIO DO RECIFE	Territorial	1.1445.155.02.0670.0000-4	RUA CEL VIR

5 rows × 63 columns

Leitura de Arquivos

- Por padrão head() exibe as 5 primeiras linhas, mas isso pode ser alterado:

```
In [26]: df.head(n=10)
```

Out[26]:

	ultimaAtualizacaoCadastral	situacao	sequencialImovel	proprietarioPrincipal	natureza	inscricaoImobiliaria	enderecoImov
0	02/12/2015	Ativo	1031171	MUNICIPIO DO RECIFE	Territorial	1.1410.190.02.0025.0000-6	RUA PAULINC
1	29/11/2015	Ativo	1032984	MUNICIPIO DO RECIFE	Territorial	1.1410.140.01.0730.0000-0	RUA DO F
2	29/11/2015	Ativo	1049160	MUNICIPIO DO RECIFE	Predial	1.1426.030.01.0170.0000-3	RUA DF
3	29/11/2015	Ativo	1050834	MUNICIPIO DO RECIFE	Predial	1.1426.043.01.0379.0000-4	RUA S
4	29/11/2015	Ativo	1063391	MUNICIPIO DO RECIFE	Territorial	1.1445.155.02.0670.0000-4	RUA CEL VIR
5	28/11/2015	Ativo	1063405	MUNICIPIO DO RECIFE	Predial	1.1445.155.02.0777.0000-9	RUA CEL VIR
6	29/11/2015	Ativo	1065114	MUNICIPIO DO RECIFE	Predial	1.1445.183.03.0012.0000-7	
7	29/11/2015	Ativo	1070312	MUNICIPIO DO RECIFE	Territorial	1.1445.714.77.0001.0000-8	AV NORTE I
8	29/11/2015	Ativo	1081039	MUNICIPIO DO RECIFE	Predial	1.1450.140.04.0075.0000-4	
9	29/11/2015	Ativo	1087517	MUNICIPIO DO RECIFE	Predial	1.1450.290.02.0110.0000-8	

10 rows × 63 columns

Leitura de Arquivos

- O tail exibe por padrão as últimas 5 linhas do DataFrame:

```
In [27]: df.tail()
```

Out[27]:

	ultimaAtualizacaoCadastral	situacao	sequencialImovel	proprietarioPrincipal	natureza	inscricaoImobiliaria	enderecoI
3090	29/11/2015	Ativo	7777400	MUNICIPIO DO RECIFE	Predial	6.1710.069.02.0251.0319-2	
3091	29/11/2015	Ativo	7777418	MUNICIPIO DO RECIFE	Predial	6.1710.069.02.0251.0320-6	
3092	29/11/2015	Ativo	7777426	MUNICIPIO DO RECIFE	Predial	6.1710.069.02.0251.0321-4	
3093	29/11/2015	Ativo	7780141	MUNICIPIO DO RECIFE	Territorial	4.1540.067.01.0010.0001-5	
3094	28/11/2015	Ativo	7780249	MUNICIPIO DO RECIFE	Predial	2.1185.015.10.0357.0001-6	

5 rows × 63 columns

Manipulação de Dados

Manipulação de Dados

- Podemos verificar quais os bairros que estão no nosso conjunto de dados.
  - O método unique() retorna uma lista ou série com todos os valores únicos, sem contar a frequência

```
# Verificando quais bairros estão no conjunto de dados

df_imoveis = df_drive
print(df_imoveis["enderecoImovelBairro"].unique())
```

```
['GRACAS' 'JAQUEIRA' 'ESPINHEIRO' 'SANTO AMARO' 'RECIFE' 'DERBY'
 'SOLEDADE' 'BOA VISTA' 'SANTO ANTONIO' 'SAO JOSE' 'ILHA JOANA BEZERRA'
 'CABANGA' 'COELHOS' 'ILHA DO LEITE' 'PASSARINHO' 'BREJO DE BEBERIBE'
 'BEBERIBE' 'CAJUEIRO' 'LINHA DO TIRO' 'PORTO DA MADEIRA'
 'ALTO SANTA TEREZINHA' 'AGUA FRIA' 'BOMBA DO HEMETERIO' 'FUNDAO'
 'CAMPINA DO BARRETO' 'ARRUDA' 'PEIXINHOS' 'CAMPO GRANDE'
 'ALTO JOSE DO PINHO' 'ENCRUZILHADA' 'HIPODROMO' 'ROSARINHO'
 'BREJO DA GUABIRABA' 'NOVA DESCOBERTA' 'VASCO DA GAMA' 'MACAXEIRA'
 'APIPUCOS' 'ALTO JOSE BONIFACIO' 'MORRO DA CONCEICAO' 'CASA AMARELA'
 'ALTO DO MANDU' 'MONTEIRO' 'CASA FORTE' 'POCO' 'PARNAMIRIM' 'SANTANA'
 'IPUTINGA' 'CORDEIRO' 'TORRE' 'MADALENA' 'ENGENHO DO MEIO' 'TORROES'
 'ZUMBI' 'CAXANGA' 'VARZEA' 'COHAB' 'ILHA DO RETIRO' 'MUSTARDINHA'
 'AFOGADOS' 'SAN MARTIN' 'ESTANCIA' 'MANGUEIRA' 'AREIAS' 'JIQUIA' 'CURADO'
 'JARDIM SAO PAULO' 'TEJIPIO' 'IBURA' 'SANCHO' 'BARRO' 'IMBIRIBEIRA'
 'PINA' 'BRASILIA TEIMOSA' 'IPSEP' 'BOA VIAGEM' 'JORDAO' 'GUABIRABA'
 'DOIS UNIDOS' 'CORREGO DO JENIPAPO' 'TAMARINEIRA']
```

Manipulação de Dados

- Podemos verificar quais os bairros que estão no nosso conjunto de dados.
  - O método nunique() retorna um valor inteiro, que é o número de valores únicos

```
# Verificando quais bairros estão no conjunto de dados
print(df_imoveis["enderecoImovelBairro"].nunique())
```

```
80
```

Manipulação de Dados

- Podemos verificar a homogeneidade da nossa amostra em relação aos bairros.
- Para isso vamos contar valores utilizando outro método disponível, o **value_counts**()

```
# Verificando a homogeneidade da amostra em relação aos bairros
print(df_imoveis["enderecoImovelBairro"].value_counts())
```

```
PINA                741
ARRUDA              454
BOA VIAGEM          389
LINHA DO TIRO       286
BREJO DA GUABIRABA  221
                   ...
CAJUEIRO              1
PASSARINHO            1
COELHOS               1
CABANGA               1
TAMARINEIRA           1
Name: enderecoImovelBairro, Length: 80, dtype: int64
```

Manipulação de Dados

- Os valores contados também podem ser normalizados para expressar porcentagens:

```
# Valores normalizados

# Verificando a homogeneidade da amostra em relação aos bairros
print(df_imoveis["enderecoImovelBairro"].value_counts(normalize=True))
```

```
PINA                  0.239418
ARRUDA                0.146688
BOA VIAGEM            0.125687
LINHA DO TIRO         0.092407
BREJO DA GUABIRABA    0.071405
                          ...
CAJUEIRO              0.000323
PASSARINHO            0.000323
COELHOS               0.000323
CABANGA               0.000323
TAMARINEIRA           0.000323
Name: enderecoImovelBairro, Length: 80, dtype: float64
```

Manipulação de Dados

- Agrupar os dados se baseando em certos critérios é outro processo que o pandas facilita bastante com o **groupby()**.

Manipulação de Dados

```
# Agrupar dados
# groupby("enderecoImovelBairro") agrupa os dados com base nos valores únicos encontrados na coluna "enderecoImovelBairro".
# Isso significa que ele criará um grupo para cada bairro encontrado nessa coluna.

# .mean() calcula a média de todas as colunas numéricas para cada grupo.
# Portanto, o resultado será um novo DataFrame onde as colunas numéricas são as médias dos valores para cada bairro.
df_imoveis_bairros = df_imoveis.groupby("enderecoImovelBairro").mean()
df_imoveis_bairros
```

..	v0ValorTestadaFicticia	aliquotaImposto	distrito	setor	quadra	face	lote	sequencial	longitude	latitude
...	3703.061034	0.002155	5.000000	1667.758621	176.793103	2.206897	218.482759	5.354091e+06	-34.909095	-8.079632
...	1052.533571	0.000000	2.000000	1308.928571	65.357143	2.571429	396.642857	2.671348e+06	-34.895341	-8.021164
...	2495.460000	0.012500	3.000000	1355.000000	135.000000	2.000000	133.000000	3.196011e+06	NaN	NaN
...	602.300000	0.000000	3.000000	1270.000000	59.000000	3.000000	256.500000	3.140412e+06	NaN	NaN
...	5374.780000	0.000000	2.000000	1371.000000	65.000000	3.000000	178.000000	2.319802e+06	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
...	4191.028333	0.000000	4.000000	1481.666667	28.000000	4.166667	313.500000	4.680386e+06	-34.912065	-8.045192
...	1433.927500	0.000000	4.083333	1521.083333	50.500000	7.083333	147.166667	4.414627e+06	-34.932818	-8.060780
...	821.564167	0.001250	4.000000	2032.083333	73.000000	1.791667	313.125000	4.890936e+06	-34.954326	-8.038687
...	705.658889	0.000000	3.000000	1234.444444	127.111111	14.222222	325.444444	4.049152e+06	NaN	NaN
...	1854.576667	0.002000	4.000000	1536.666667	85.666667	1.333333	422.333333	4.366892e+06	-34.917579	-8.051997

C.e.S.A.R school

Manipulação de Dados

- Para extrairmos dados de uma coluna basta acessá-lo normalmente:

```
# Extrair dados de uma coluna

df_imoveis_bairros["areaTotalConstruidaMultipla"].sort_values()
```

```
enderecoImovelBairro
AFOGADOS               25.000000
AGUA FRIA             190.000000
IPUTINGA              199.450000
AREIAS                240.000000
BREJO DE BEBERIBE     273.178947
                          ...
TORRE                        NaN
TORROES                      NaN
VARZEA                       NaN
VASCO DA GAMA                NaN
ZUMBI                        NaN
```

Tratando dados Incompletos

- Por muitos motivos podem haver dados incompletos no dataset.
- O pandas simplifica a remoção de quaisquer linhas ou colunas que possuem um valor NaN.
  - dropna() retorna as linhas que não contém um NaN.

```
# Tratando dados incompletos
print(df_imoveis.shape)
df_imoveis_new = df_imoveis.dropna()
print(df_imoveis_new.shape)
```

```
(3095, 63)
(704, 63)
```

```
# Removendo apenas se houver valores NaN nas colunas latitude e longitude
df_imoveis_new = df_imoveis.dropna(subset=["latitude", "longitude"])
print(df_imoveis_new.shape)
```

```
(1645, 63)
```

Exercícios

- Entre site http://dados.recife.pe.gov.br/ e faça o download do arquivo: Resultado Final dos Alunos - 2022
- 1. Verifique quais os bairros possuem escolas no conjunto de dados.
- 2. Descubra o número de alunos por bairro.
- 3. Descubra o número de escolas por bairro.
- 4. Descubra o número de alunos aprovados por bairro.
- 5. Descubra o número de alunos reprovados por frequência que estudam pela MANHÃ em cada escola.
- 6. Descubra o número de alunos reprovados por frequência que estudam pela NOITE em cada Bairro.

Visualização de Dados

Visualização de Dados

- Métodos para visualização de dados:
  - Matplotlib
  - Seaborn
  - Pandas
  - Ploty
  - Entre outros.

Visualização de Dados

- Iremos ver as funcionalidades básicas da biblioteca Matplotlib
  - Biblioteca para visualização de gráficos onde é possível se ter mais liberdade no conteúdo e possibilidades de visualização.

Visualização de Dados - Matplotlib

- Existem uma série de parâmetros que podem ser setados no momento da plotagem de gráficos
- Os básicos serão apresentados a seguir
- Para se aprofundar mais, acesse: matplotlib.pyplot

Visualização de Dados - Matplotlib

- Primeiro passo, importar biblioteca:

```
] # Importando a biblioteca
  import matplotlib.pyplot as plt
```

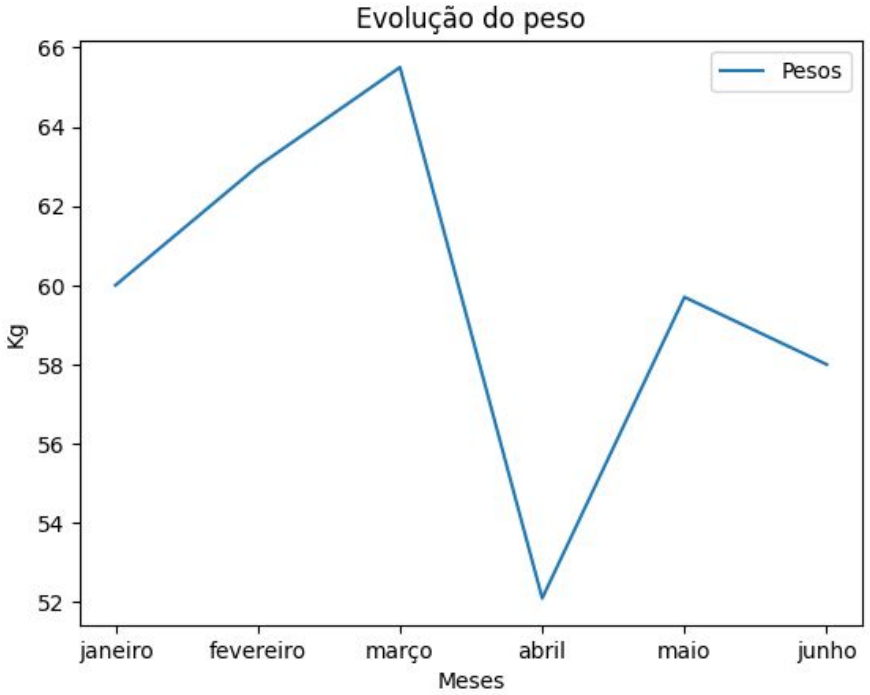
Visualização de Dados - Matplotlib

- Gráficos tipo plot()
  - Gráfico de linha
    - São os gráficos mais simples

```
# Definindo variáveis
x = ['janeiro', 'fevereiro', 'março', 'abril', 'maio', 'junho']
y = [60, 63, 65.5, 52.1, 59.7, 58]

# Atribuindo um título ao gráfico
plt.title('Evolução do peso')
plt.xlabel('Meses')
plt.ylabel('Kg')

# Atribuindo uma legenda
plt.plot(x, y, label = 'Pesos')
plt.legend()

#Exibindo o gráfico gerado
plt.show()
```

Visualização de Dados - Matplotlib

- Gráficos tipo plot()
  - Gráfico de linha
    - São os gráficos mais simples

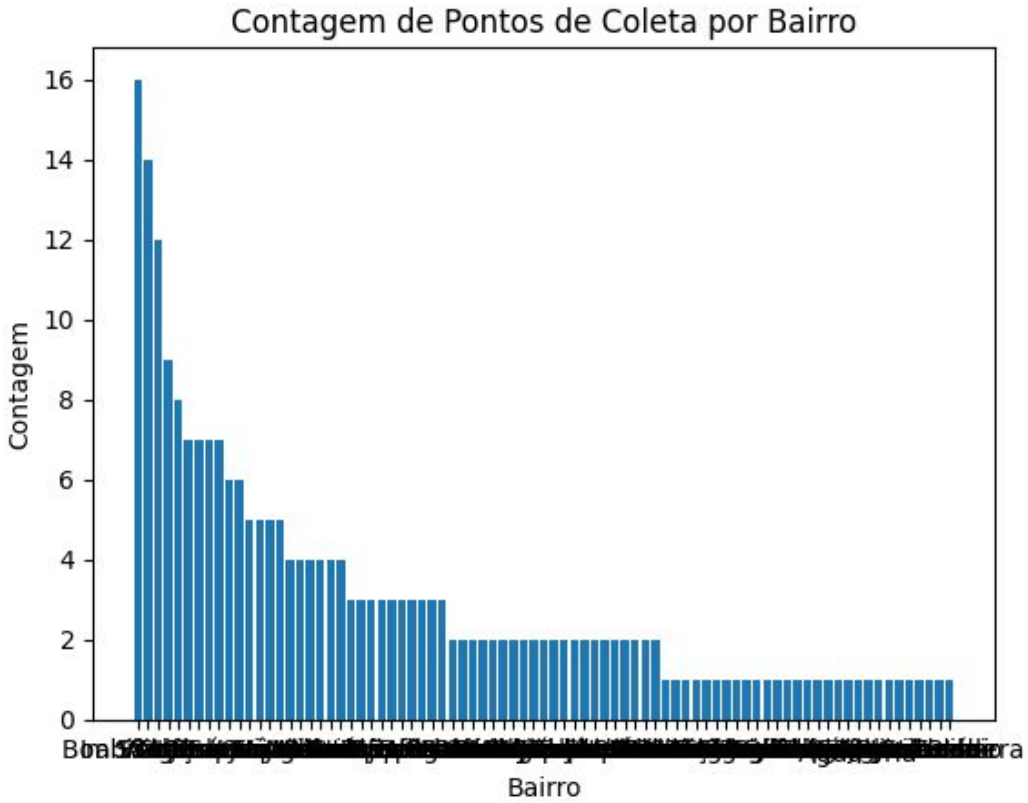


Visualização de Dados - Matplotlib

- Gráficos tipo bar()
  - Gráficos de barra
    - Plotar gráfico de barras que apresentem o número de pontos de coleta por bairro:

```
# Carregando csv
link = "http://dados.recife.pe.gov.br/dataset/10a009f2-f9bb-457b-8f78-d8f0dc3ced37/resource/ef521704-6960-4ef1-8f98-a60db4a0d79b/download/pontos_coleta.csv"
df_coletas = pd.read_csv(link, sep=";")

# Crie um gráfico de barras
plt.bar(df_coletas["bairro"].value_counts().index, df_coletas["bairro"].value_counts().values)

# Adicione um título ao gráfico
plt.title("Contagem de Pontos de Coleta por Bairro")

# Rotule os eixos
plt.xlabel("Bairro")
plt.ylabel("Contagem")

# Mostre o gráfico
plt.show()
```

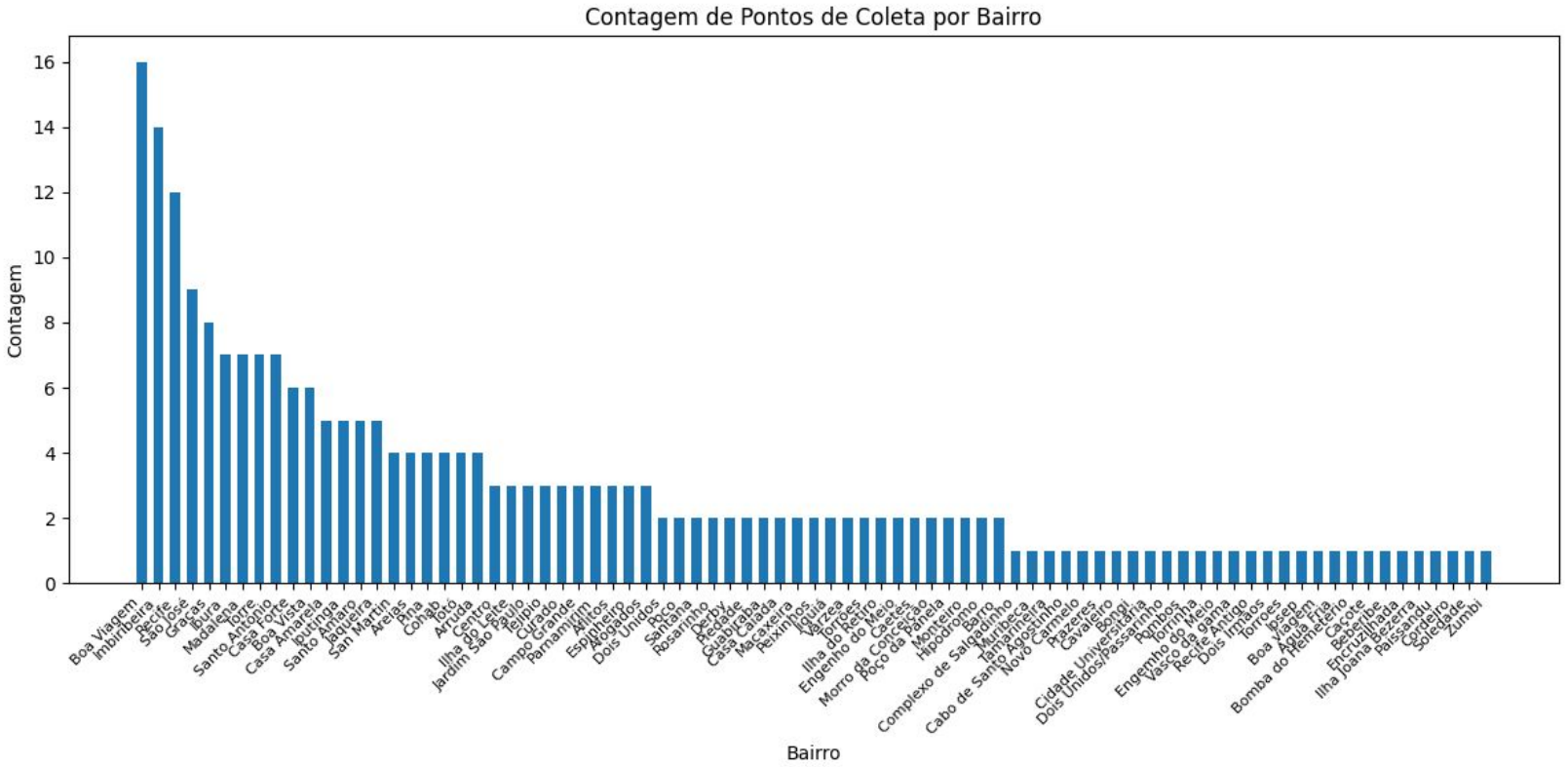
Visualização de Dados - Matplotlib



Visualização de Dados - Matplotlib

```
# Aprimorando a visualização

# Crie um gráfico de barras
plt.figure(figsize=(12, 6))  # Defina um tamanho de figura mais largo
plt.bar(df_coletas["bairro"].value_counts().index, df_coletas["bairro"].value_counts().values, width=0.6)  # Aumente a largura das barras

# Ajuste a inclinação dos rótulos do eixo x e diminua o tamanho da fonte
plt.xticks(rotation=45, ha='right', fontsize=8)

# Adicione um título ao gráfico
plt.title("Contagem de Pontos de Coleta por Bairro")

# Rotule os eixos
plt.xlabel("Bairro")
plt.ylabel("Contagem")

# Mostre o gráfico
plt.tight_layout()  # Ajuste o layout para evitar cortes nos rótulos
plt.show()
```

Visualização de Dados - Matplotlib

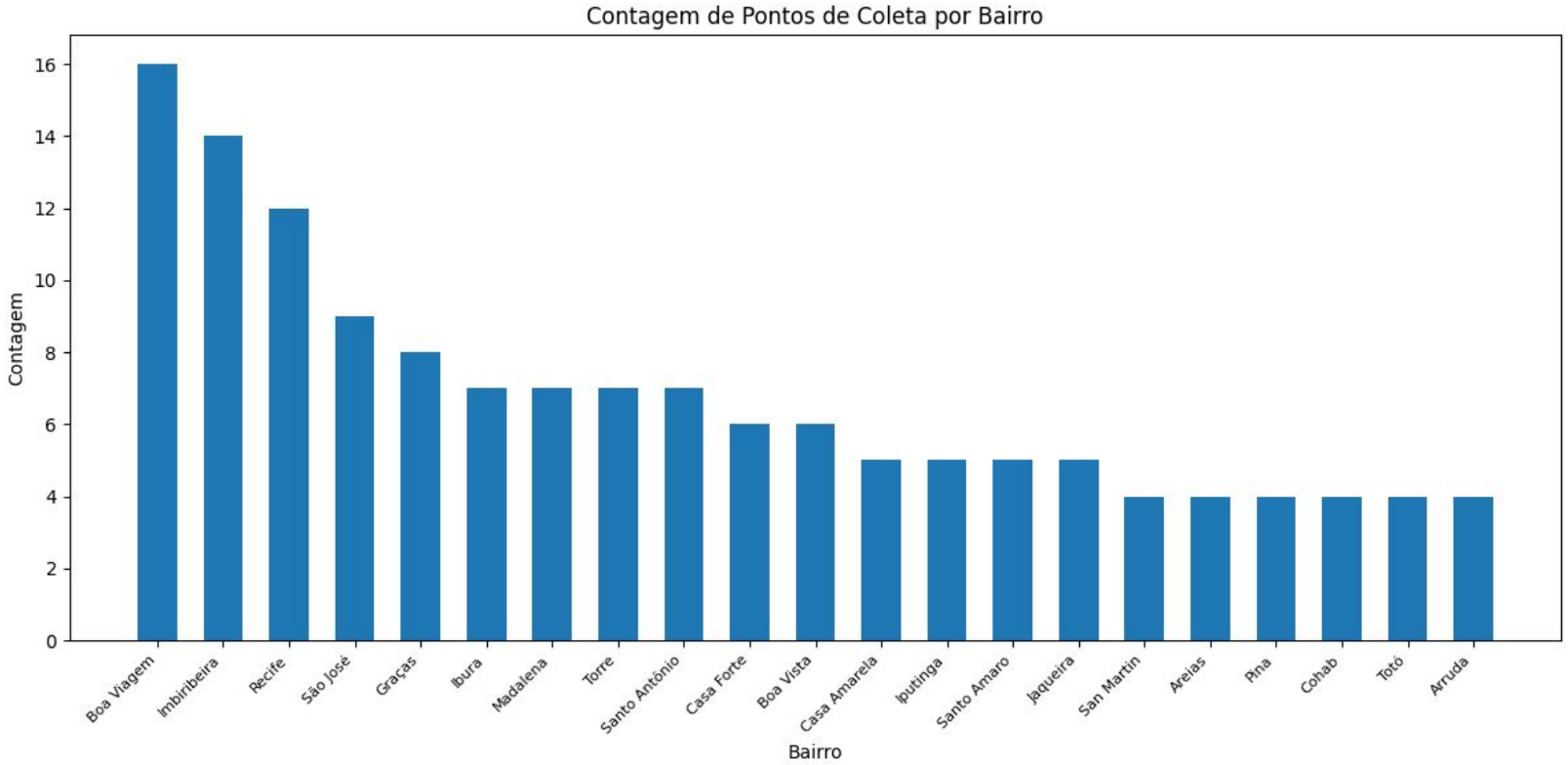


Visualização de Dados - Matplotlib

- Gráficos tipo bar()
  - Gráficos de barra
    - Plotar apenas os bairros que tenham 4 ou mais pontos de coleta.

```
# Plotar apenas os bairros que tenham 4 ou mais pontos de coleta.
df_bairros_muitos_pontos_coleta = df_coletas["bairro"].value_counts()[df_coletas["bairro"].value_counts() >= 4]

# Crie um gráfico de barras
plt.figure(figsize=(12, 6))  # Defina um tamanho de figura mais largo
plt.bar(df_bairros_muitos_pontos_coleta.index, df_bairros_muitos_pontos_coleta.values, width=0.6)  # Aumente a largura das barras

# Ajuste a inclinação dos rótulos do eixo x e diminua o tamanho da fonte
plt.xticks(rotation=45, ha='right', fontsize=8)

# Adicione um título ao gráfico
plt.title("Contagem de Pontos de Coleta por Bairro")

# Rotule os eixos
plt.xlabel("Bairro")
plt.ylabel("Contagem")

# Mostre o gráfico
plt.tight_layout()  # Ajuste o layout para evitar cortes nos rótulos
plt.show()
```

Visualização de Dados - Matplotlib



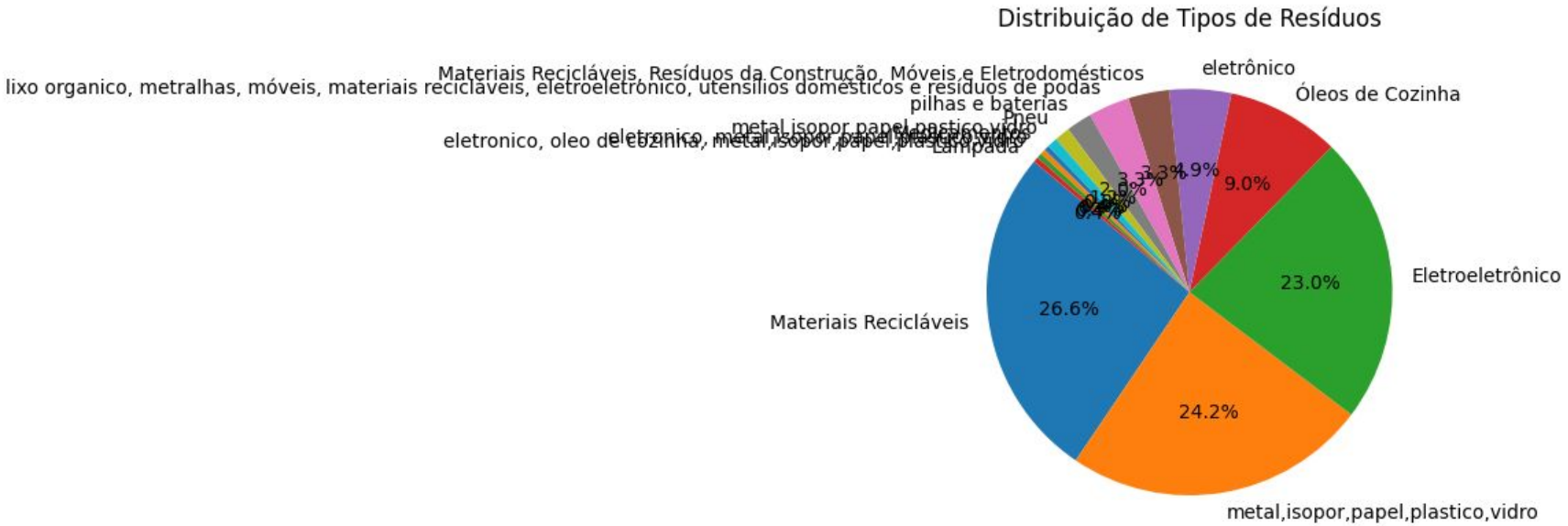
Visualização de Dados - Matplotlib

- Gráficos tipo pie()
  - Gráficos de pizza
    - Plotar gráfico de pizza que mostra a contagem e distribuição dos tipos de resíduos que são colhidos.

```
# Crie o gráfico de pizza
plt.pie(df_coletas["tiporesiduo"].value_counts().values, labels=df_coletas["tiporesiduo"].value_counts().index, autopct='%1.1f%%', startangle=140)

# Adicione um título
plt.title("Distribuição de Tipos de Resíduos")

# Mostre o gráfico
plt.show()
```

Visualização de Dados - Matplotlib

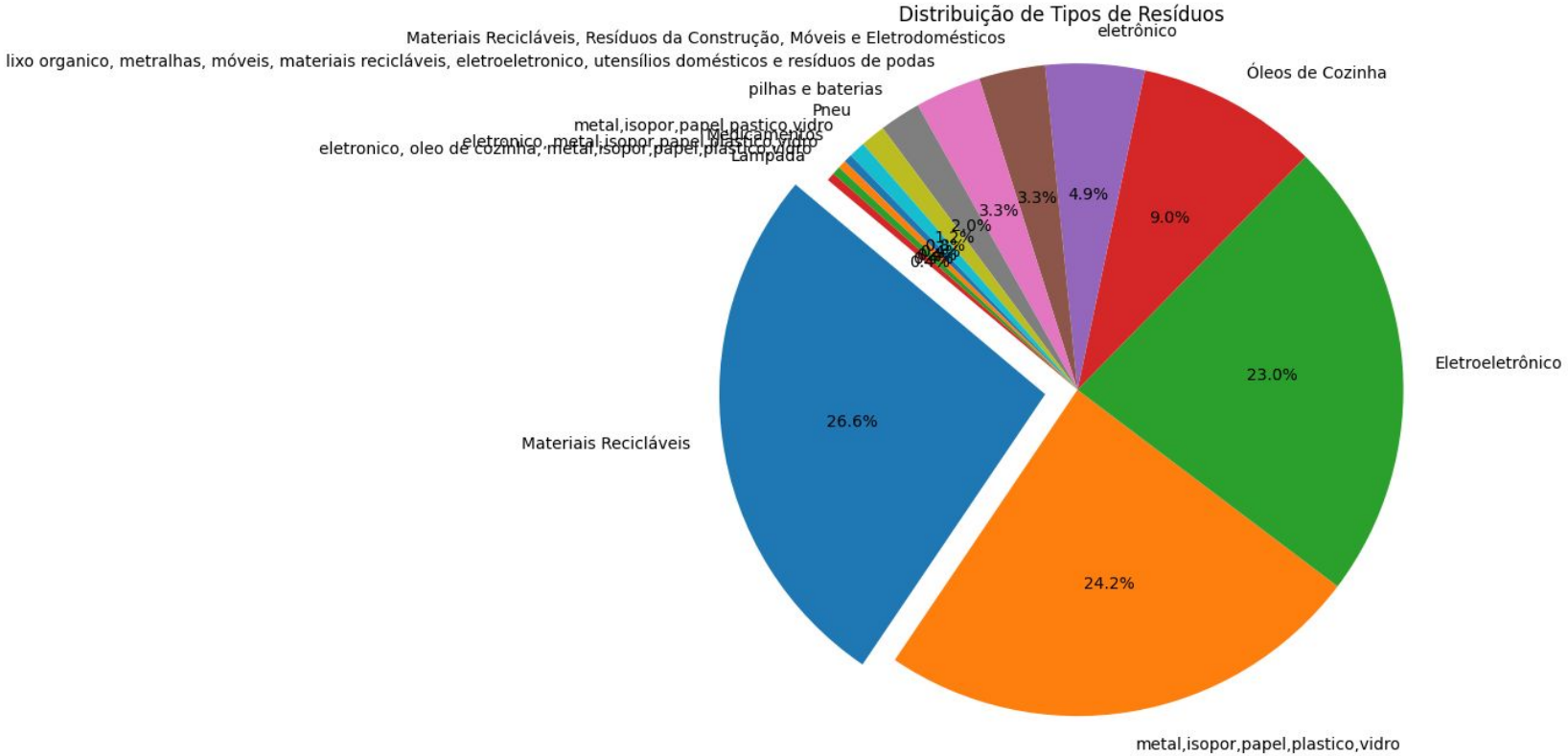
- Gráficos tipo pie()
  - Gráficos de pizza
    - Plotar gráfico de pizza que mostra a contagem e distribuição dos tipos de resíduos que são colhidos.



Visualização de Dados - Matplotlib

```
# Aprimorando o gráfico de pizza

import numpy as np
# Crie o gráfico de pizza
plt.figure(figsize=(8, 8))  # Defina o tamanho da figura

# Explodir a primeira fatia (destacar)
tupla_tamanho = (0,) * (df_coletas["tiporesiduo"].value_counts().values.size - 1)
explode = (0.1,) + tupla_tamanho

# Crie o gráfico de pizza
plt.pie(df_coletas["tiporesiduo"].value_counts().values, labels=df_coletas["tiporesiduo"].value_counts().index,
        explode=explode, autopct='%1.1f%%', startangle=140)

# Adicione um título
plt.title("Distribuição de Tipos de Resíduos")

# Mostre o gráfico
plt.axis('equal')  # Para manter o aspecto de círculo
plt.show()
```

Visualização de Dados - Matplotlib

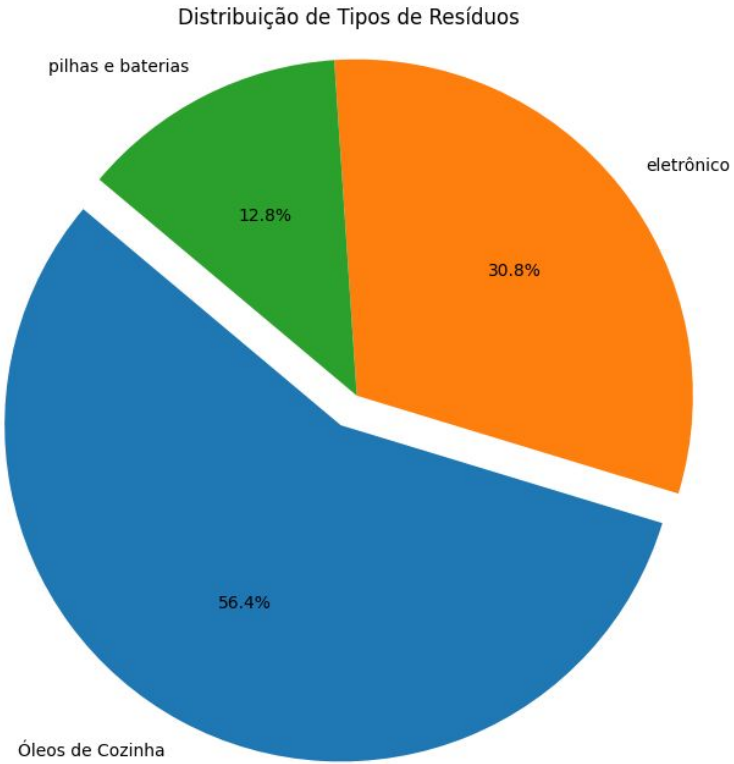


Visualização de Dados - Matplotlib

- Gráficos tipo pie()
  - Gráficos de pizza
    - Plotar gráfico de pizza que mostra a contagem e distribuição dos tipos de resíduos: Óleos de cozinha, pilhas e baterias e eletrônicos.

```
# Exibir um gráfico de pizza que mostra a contagem de distribuição dos
# tipos de resíduos: Óleos de cozinha, pilhas e baterias e eletrônicos.
lista_residuos = ["eletrônico", "pilhas e baterias", "Óleos de Cozinha"]
verifica = df_coletas["tiporesiduo"].isin(lista_residuos)

# Crie o gráfico de pizza
plt.figure(figsize=(8, 8))  # Defina o tamanho da figura

# Explodir a primeira fatia (destacar)
tupla_tamanho = (0,) * (df_coletas[verifica]["tiporesiduo"].value_counts().values.size - 1)
explode = (0.1,) + tupla_tamanho

# Crie o gráfico de pizza
plt.pie(df_coletas[verifica]["tiporesiduo"].value_counts().values,
        labels=df_coletas[verifica]["tiporesiduo"].value_counts().index, explode=explode, autopct='%1.1f%%', startangle=140)

# Adicione um título
plt.title("Distribuição de Tipos de Resíduos")

# Mostre o gráfico
plt.axis('equal')  # Para manter o aspecto de círculo
plt.show()
```

Visualização de Dados - Matplotlib



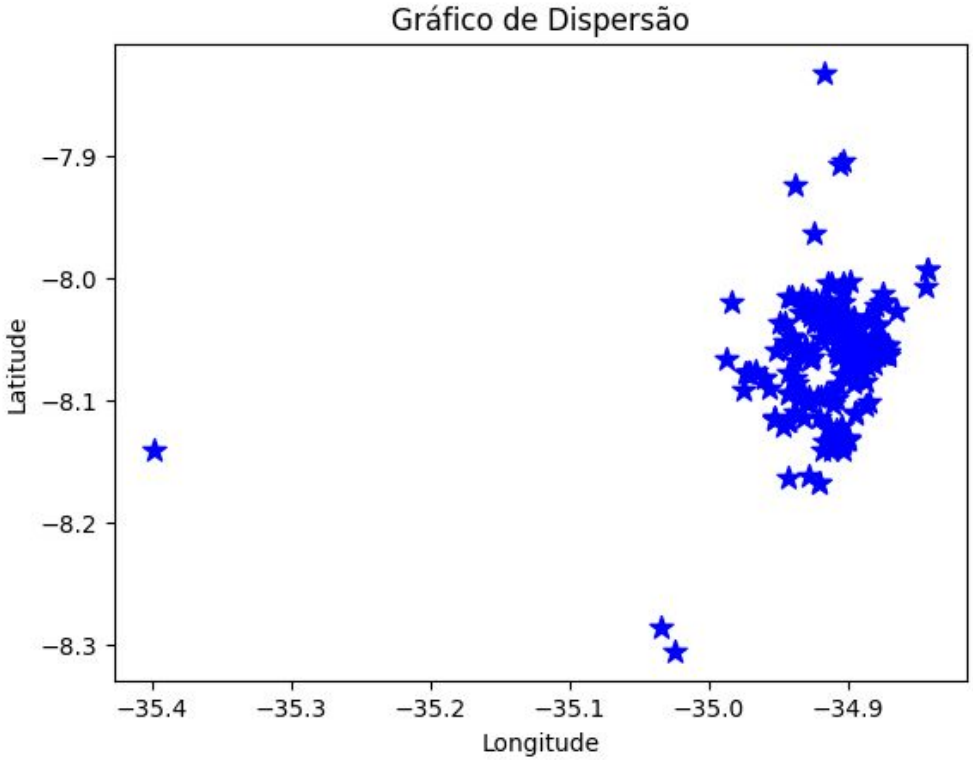
Visualização de Dados - Matplotlib

- Gráficos tipo scatter()
  - Gráfico de dispersão

```
# Criando um gráfico
plt.scatter(df_coletas['longitude'], df_coletas['latitude'], label = 'Pontos', color = 'b', marker = '*', s = 100)

plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Gráfico de Dispersão')

plt.show()
```

Visualização de Dados - Matplotlib

- Gráficos tipo scatter()
  - Gráficos de dispersão



Exercício

- Lista de Exercício 02 disponível