Objetivos

- Apresentar e utilizar o pacote **pandas**
- Como carregar uma base dados
- Como visualizar os dados
- Intuição de análise exploratória de dados

Introdução a análise de dados usando Pandas

Vamos começar pelo começo! Vamos escolher um dataset (conjunto de dados) para analisar.

Vamos utilizar um pacote do python capaz de trabalhar com tabelas de dados chamada **pandas**, para essas tabelas chamamos de ***dataframe***.

Veja mais em: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

Para instalar

- **pandas**: `pip install pandas`

LEIA A DOCUMENTAÇÃO: https://pandas.pydata.org/docs/index.html

```
import pandas as pd
```

Hello World! do mundo dos dados

Conjunto de dados `Dataset`

Para trabalhar com análise de dados precisamos de.... `DADOS` .

Podemos escolher qualquer base de dados disponivel na internet, ou até mesmo criar nosso proprio dataset.

Vamos simplificar essa etapa e começar analisando uma base pequena e muito famosa chamada **iris** que esta disponivel em:

ref: https://archive.ics.uci.edu/ml/datasets/Iris

Conheça outros datasets: https://archive.ics.uci.edu/ml/datasets.php

```
import pandas as pd

# Caminho do arquivo
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Lê e carrega o arquivo para a memória
df = pd.read_csv(url)


# Lê e carrega o arquivo para a memória
df = pd.read_csv(url, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])

# Verifica se há dados faltantes
dados_faltantes = df.isnull()  # ou df.isna()

# Conta os valores nulos em cada coluna
contagem_nulos_por_coluna = dados_faltantes.sum()

print("Dados faltantes por coluna:")
print(contagem_nulos_por_coluna)
```

```
Dados faltantes por coluna:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
class           0
dtype: int64
```

Conhecendo os dados

Conhecendo os dados

Essa etapa é muito importante, CONHECER OS DADOS!

Quanto mais você conhece a base de DADOS maior a possibilidade de extrair INFORMAÇÕES úteis para tomada de decisão.

```
df.head()
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Note que a primeira linha não contem os nomes das colunas ou `atributos` (variaveis) e sim, dados (valores).

Dependendo da base dados utilizada e como você carrega no pandas, os dados da primeira linha são importados como atributos.

Vamos adicionar um cabeçario ao nosso dataframe. Mas o que podemos adicionar???

Vamos dar uma olhada no repositório oficial onde dadas informações sobre o dataset e é dito que as variaveis são:

```
 Attribute Information:

 1. sepal length in cm
 2. sepal width in cm
 3. petal length in cm
 4. petal width in cm
 5. class:
 -- Iris Setosa
 -- Iris Versicolour
 -- Iris Virginica
```

```
# Define o nome das colunas
header = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
# Lê e carrega o arquivo para a memória
df = pd.read_csv(url, header=None, names=header)
```

```
# Retorna um trecho com as 5 primeiras linhas do dataframe
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

análisando o dataset

Agora que já carregamos o dataset corretamente, vamos começar a analisa-lo. o método `info()` é um bom ponto de partida para isso.

```
# Mostra informações sobre o dataframe em si
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
```

```
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
# exibe o shape (dimensoẽs) do dataframe
df.shape
```

```
(150, 5)
```

Desafio 1

Analisando as informações do dataset iris, responda:

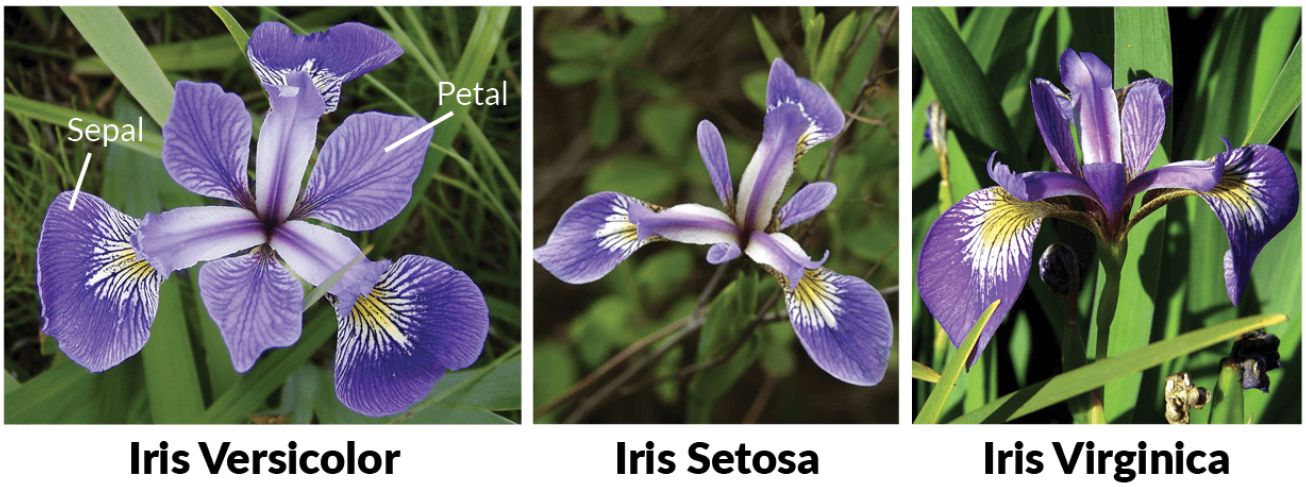
2. Quantos dados existem nesse dataset?
3. Qual a quantidade de atributos?
4. Existe valores faltantes?
5. De que tipo são os dados (dtype)?

```
R2: Existem 150 dados neste dataset;
R3: Há 5 atributos:  sepal_length, sepal_width, petal_length, petal_width e species;
R4: Não foram identificados dados nulos no dataset
R5: float64 e object;
```

Análisando dos dados mais a fundo

São 150 exemplares de flor de íris, pertencentes a três espécies diferentes: **setosa**, **versicolor** e **virginica**, sendo 50 amostras de cada espécie.

Os atributos de `largura e comprimento de sépala` e `largura e comprimento de pétala` de cada flor fooram medidos manualmente.



Resumo estatístico

O método `describe()` gera um resumo estatístico dos dados contidos em um DataFrame ou Series.

Retorna estatísticas descritivas como `média, desvio padrão, valor mínimo, quartis e valor máximo` para as colunas numéricas do DataFrame

```
df.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000

Note que o método ***describe()*** não exibe a coluna **species**, pois se trata de uma coluna não-numérica.

Apenas as colunas numéricas estão presentes, o atributo **species** indica rótulos - trata-se de dados categóricos.

max 7.900000 4.400000 6.900000 2.500000

```
# retorna a quantiade de classes da coluna

df.species.unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

Agrupando dados

O método `groupby` é usada para agrupar dados com base em valores específicos de uma ou mais colunas.

Permite realizar operações em grupos de dados e é muito útil e versatil para análise e agregação de dado.

```
# agrupamento por média

df.groupby('species').mean()
```

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	5.006	3.418	1.464	0.244
Iris-versicolor	5.936	2.770	4.260	1.326
Iris-virginica	6.588	2.974	5.552	2.026

```
#  Quantidade de cada categoria
df.groupby('species').size()
```

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Limpeza de Dados

Base de dados do mundo real podem conter diversos problemas, é muito comum o lidar com valores faltantes em um dataset.

Como exemplo, vamos usar o conjunto de dados Iris, mas introduzimos algumas 'imperfeições' para fins de demonstração.

Para introduzir valores faltantes, podemos usar o seguinte código:

```
 ## Gera dados faltante no dataset
 for col in df_iris.columns[:-1]:
     df_iris.loc[np.random.choice(df_iris.index, 5), col] = np.nan
```

```
import numpy as np

# cópia de df
df_iris = df

# Gera dados faltante no dataset
for col in df_iris.columns[:-1]:
    df_iris.loc[np.random.choice(df_iris.index, 5), col] = np.nan
```

```
df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 131 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  121 non-null    float64
 1   sepal_width   121 non-null    float64
 2   petal_length  122 non-null    float64
 3   petal_width   121 non-null    float64
 4   species       131 non-null    object 
dtypes: float64(4), object(1)
memory usage: 10.2+ KB
```

```
# Verifica os valores ausentes

df_iris.isnull().sum()
```

```
sepal_length    10
sepal_width     10
petal_length     9
petal_width     10
species          0
dtype: int64
```

Excluindo linhas

Podemos simplismente excluir as linhas ou colunas que contenham dados faltantes, basta usar a função dropna pandas, utilizando como parâmetros o axis = 1 para dizer que queremos deletar a coluna ou axis = 0 para linha e inplace = True para aplicarmos no dataset e não criarmos uma cópia deste:

- `axis=0` --> exclui linha
- `axis=1` --> exclui coluna

`Pense bemmmm!!!!` A decisão por qual vai excluir depende do problema que você está atacando...

```
df_iris.dropna(axis=0, inplace=True)
```

Preenchimento com valores

Você pode preencher os valores faltantes com médias, medianas, modas ou outros valores relevantes.

Isso ajuda a manter o tamanho do conjunto de dados, mas pode introduzir viés nos resultados.

`Pense bemmmm!!!!` A decisão por qual valor preencher depende do problema que você está atacando...

```
# Tratamento de valores faltantes: imputação média

for col in df_iris.columns[:-1]:  # a coluna de especie não entra
    mean_val = df_iris[col].mean()
    df_iris[col].fillna(mean_val, inplace=True)
```

```
df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 131 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  131 non-null    float64
 1   sepal_width   131 non-null    float64
 2   petal_length  131 non-null    float64
 3   petal_width   131 non-null    float64
 4   species       131 non-null    object 
dtypes: float64(4), object(1)
memory usage: 10.2+ KB
```

Analisando informações em gráficos

Uma análise gráfica pode ajudar a compreeder melhor os dados que estamos trabalhando....

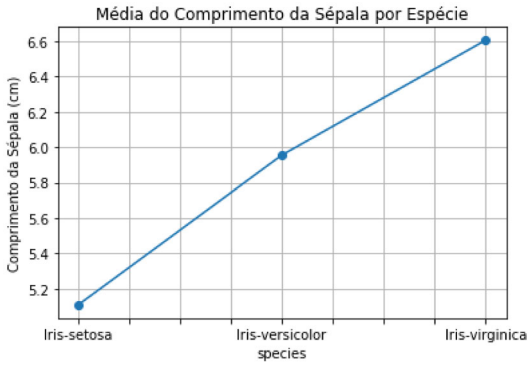
Vamos explorar diferentes tipos de visualizações que podem nos ajudar a entender melhor nossos dados.

Vamos usar o `matplotlib` e o `seaborn` para nos ajudar.

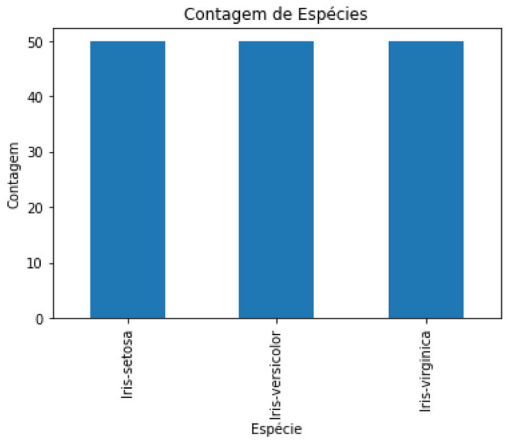
Se precisar instalar:

- pip install matplotlib seaborn

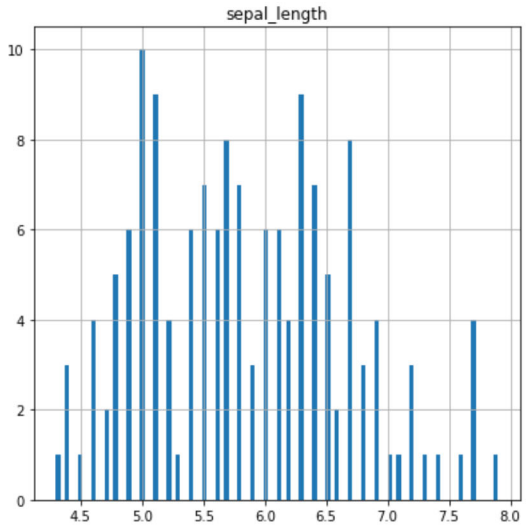
```
import matplotlib.pyplot as plt
import seaborn as sns


# Gráfico de linha para a média do comprimento da sépala de cada espécie

grouped = df_iris.groupby('species')['sepal_length'].mean()
grouped.plot(kind='line', marker='o')


plt.title('Média do Comprimento da Sépala por Espécie')
plt.ylabel('Comprimento da Sépala (cm)')
plt.grid(True)
plt.show()
```

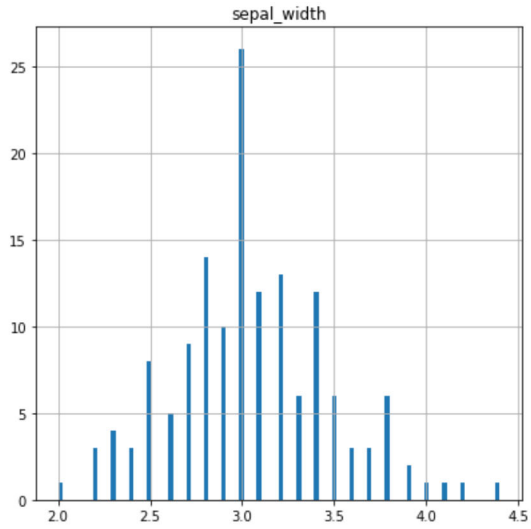


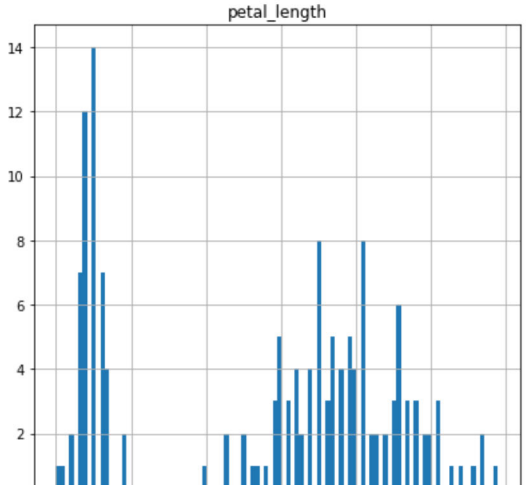

```
# Gráfico de Barras

species_count = df['species'].value_counts()
species_count.plot(kind='bar')

plt.title('Contagem de Espécies')
plt.xlabel('Espécie')
plt.ylabel('Contagem')
plt.show()
```

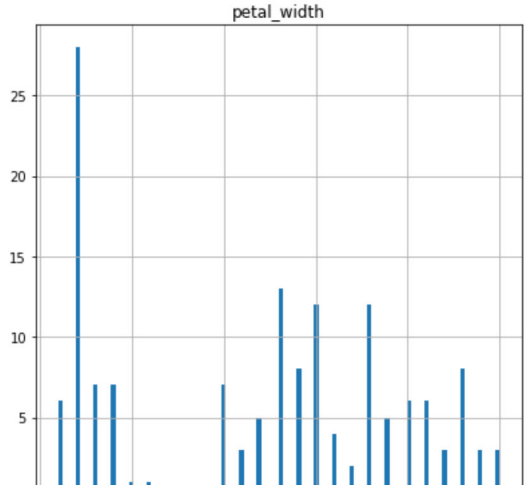



```
# Lembra de histograma, que exibe uma gráfico de frequência.
df.hist(bins=100, figsize=(15, 15))
plt.show()
```

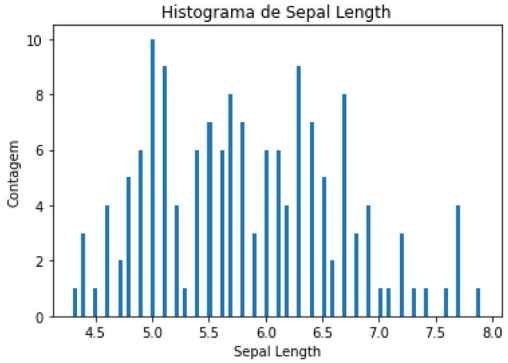




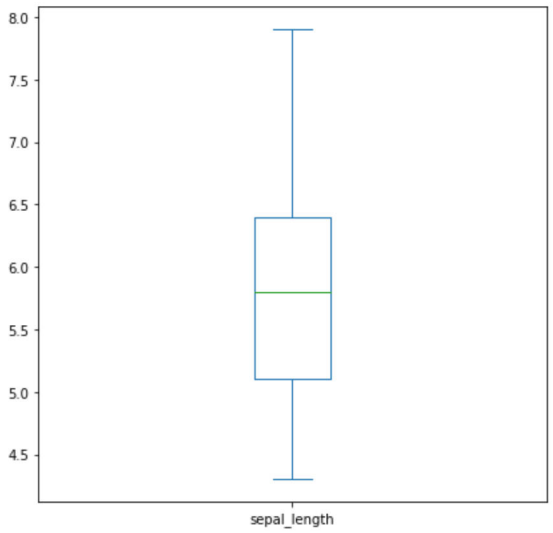


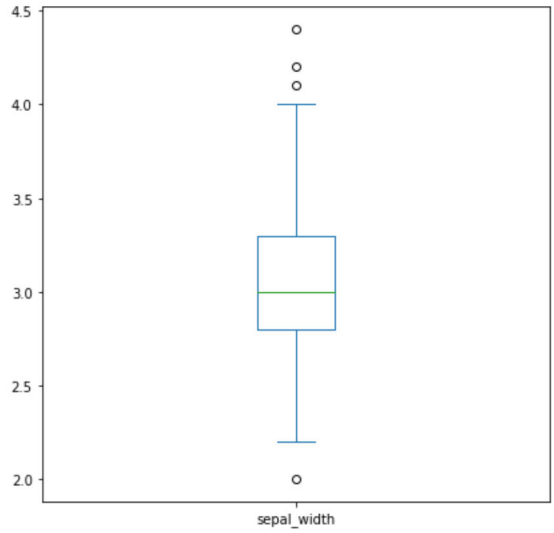


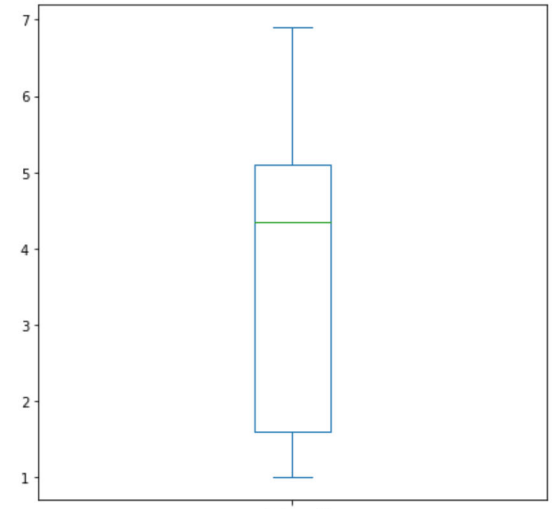

```
# Histograma apenas de um atributo

plt.hist(df['sepal_length'], bins=100)


plt.title('Histograma de Sepal Length')
plt.xlabel('Sepal Length')
plt.ylabel('Contagem')
plt.show()
```

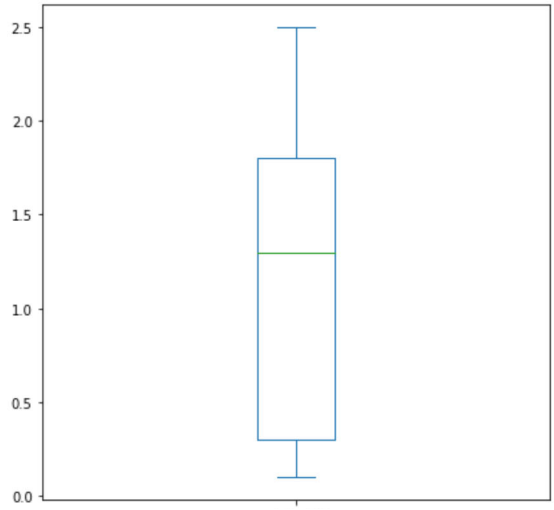



```
# box plot
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, figsize=(15, 15))
plt.show()
```

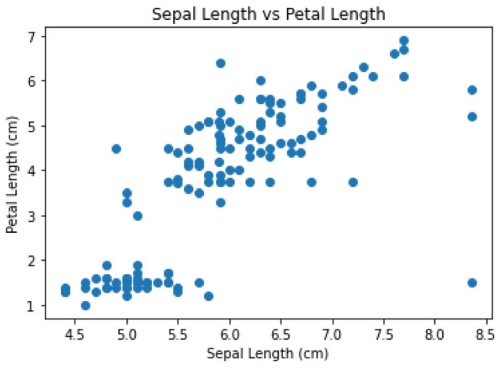






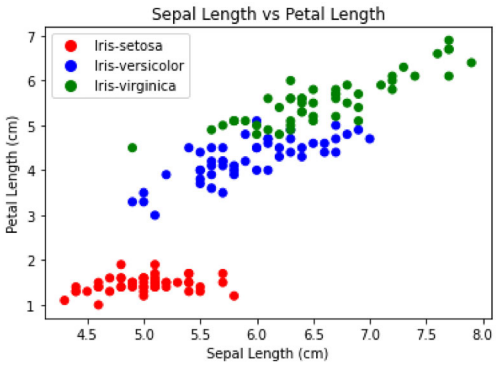


```
# Grafico de dispersão

plt.scatter(df_iris['sepal_length'], df_iris['petal_length'])

plt.title('Sepal Length vs Petal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Length (cm)')
plt.show()
```

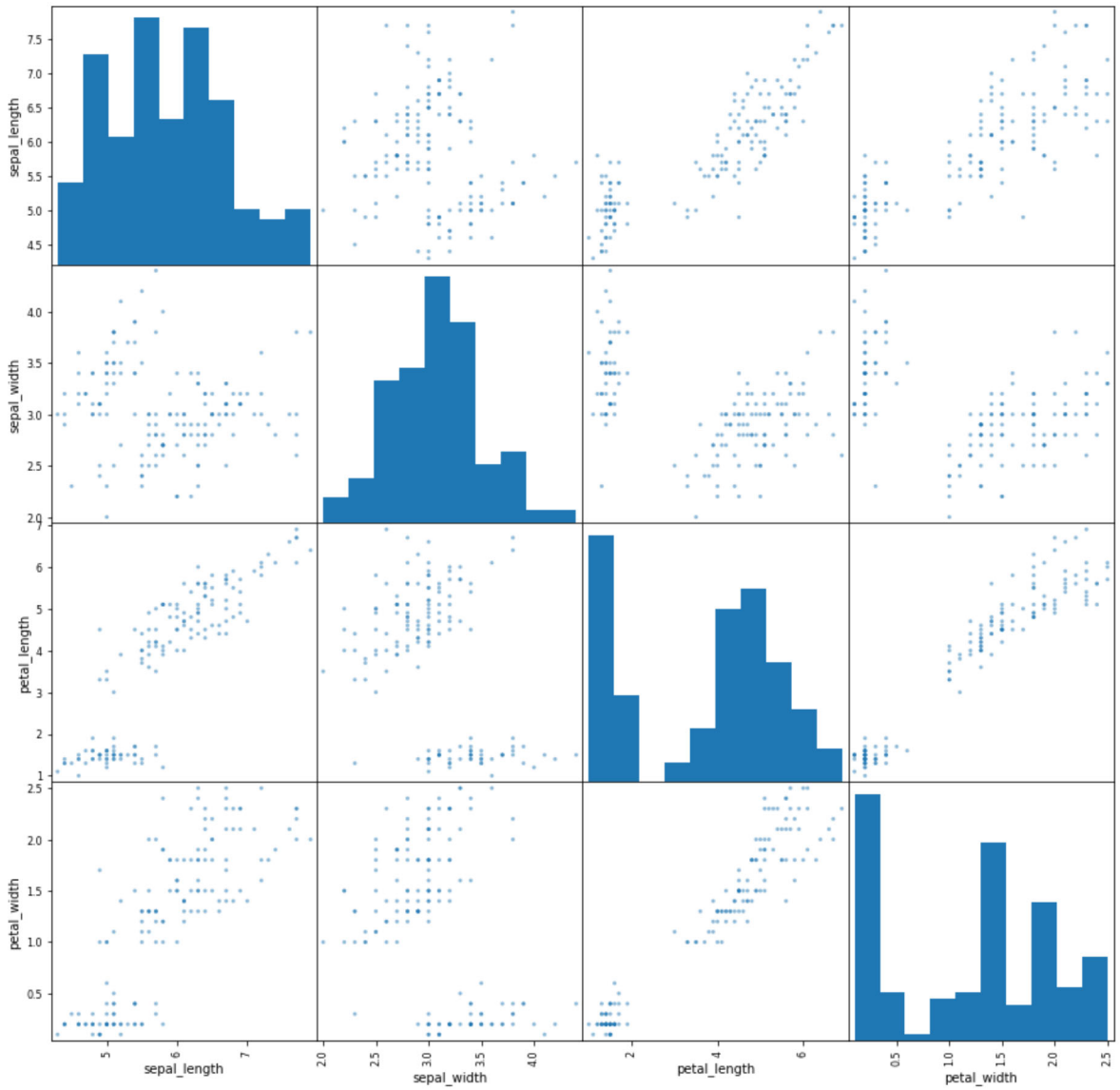



```
# Os mesmos dados mas agora cada classe de uma cor diferente

colors = {'Iris-setosa':'red', 'Iris-versicolor':'blue', 'Iris-virginica':'green'}

plt.scatter(df['sepal_length'], df['petal_length'], c=df['species'].map(colors), label=colors)

plt.title('Sepal Length vs Petal Length')
```

```
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Petal Length (cm)')
plt.legend(handles=[plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=color, markersize=10) for color in colors.values()], labels=c
plt.show()
```




```
# scatter plot matrix
# tudo junto e misturado

from pandas.plotting import scatter_matrix

scatter_matrix(df,figsize=(15, 15))

plt.show()
```



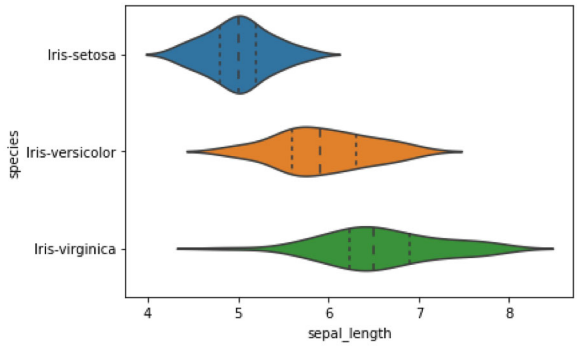
Vamos utilizar o `seaborn` para visualizar gráficos mais bonitos

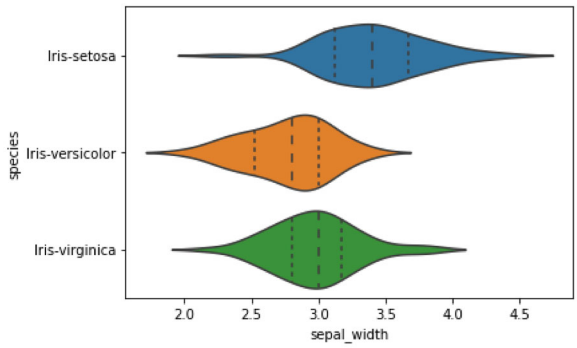
```
import seaborn as sns

# A cor vem do campo `species` do dataframe

sns.pairplot(df, hue='species', height=5)

plt.show()
```

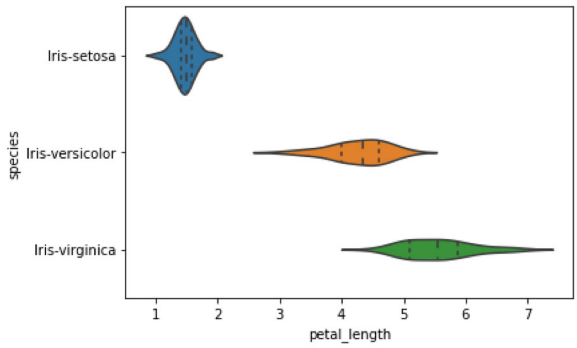
O Violin plot é similar ao box plot, exibe a distribuição de variaveis numéricas em niveis, pode ser configurada de muitas formas e é uma forma de visualização interessante de dados.

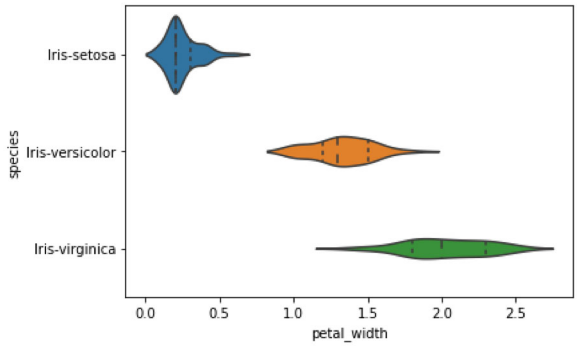
Saiba mais em: https://seaborn.pydata.org/generated/seaborn.violinplot.html

```
# Violin plot
g = sns.violinplot(y='species', x='sepal_length', data=df, inner='quartile')
plt.show()
g = sns.violinplot(y='species', x='sepal_width', data=df, inner='quartile')
plt.show()
g = sns.violinplot(y='species', x='petal_length', data=df, inner='quartile')
plt.show()
g = sns.violinplot(y='species', x='petal_width', data=df, inner='quartile')
plt.show()
```









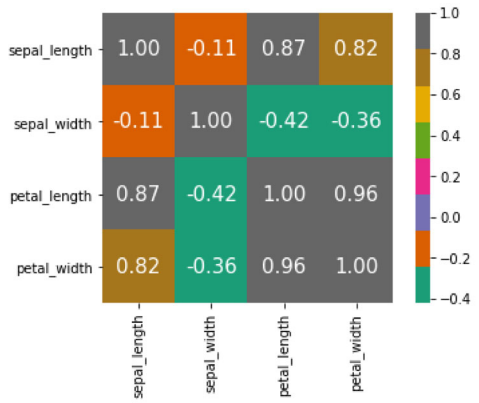
Correlação entre atributos

A matriz de correlação avalia a relação entre duas ou mais variaveis (correlação).

valores:

- 0.9 a 1 positivo ou negativo indica uma correlação muito forte.
- 0.7 a 0.9 positivo ou negativo indica uma correlação forte.
- 0.5 a 0.7 positivo ou negativo indica uma correlação moderada.
- 0.3 a 0.5 positivo ou negativo indica uma correlação fraca.
- 0 a 0.3 positivo ou negativo indica uma correlação desprezível.
- **lembre-se*** que: alta correlação não implica em causa. (causa e consequência). Para entender melhor vale a pena dar uma olhada nesse site que mostra correlações absurdas..['spurious correlations']

```
cols = ['sepal_length', 'sepal_width', 'petal_length','petal_width']
corr_matx = df[cols].corr()

heatmap = sns.heatmap(corr_matx,cbar=True,annot=True,square=True,fmt='.2f',annot_kws={'size': 15},yticklabels=cols,xticklabels=cols,cmap='Darl
```



Desafio 2

Analise os gráficos gerados até o momento para responder as questões abaixo:

1. A especie que possui na média a menor sepala é a mesma que possui a menor petala?
2. Existe sobreposição entre as medições, ou seja, uma petala de tamanho x pode ser tanto da especie versicolor ou da virginica?
3. É possivel classificar as especies de iris com base apenas em suas dimensões?

```
### Implemente sua solução e apresente sua análise....
R1: Não, na média, a espécie que possui a menor sépala não é a que possui a menor pétala, analisando os gráficos trazidos. A íris setosa apre
ao passo que a iris versicolor apresenta a menor sépala, na média.

R2: Sim, existe sobreposição entre as medidções das epécies versicolor e virginica em determinados intervalos.

R3. Sim, é possível classificar as espécies de iris com base em suas dimensões, na medida em que elas são bem distintas por espécie, considera
```

Acessando dados de um Dataframe

Há várias maneiras de acessar o conteúdo de um `DataFrame`. Os mais simples são aqueles que usam a notação de colchetes. Primeiramente, podemos acessar uma coluna através do seu índice, retornando uma `Series`, ou seja, uma coluna do dataframe.

```
df['petal_length'].head()
```

```
0    1.4
1    1.4
```

```
2    1.3
3    1.5
4    1.4
Name: petal_length, dtype: float64
```

Por outro lado, se dentro dos colchetes passamos uma lista de nomes de coluna, o reultado é outro `DataFrame` contendo aquelas colunas. Isso vale inclusive para uma coluna simples:

```
# Mostra apenas a coluna petal_len
df[['petal_length']].head()
```

	petal_length
0	1.4
1	1.4
2	1.3
3	1.5
4	1.4

```
# Mostra as colunas petal_length e petal_width
df[['petal_length', 'petal_width']].head()
```

	petal_length	petal_width
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2

```
# Criando uma nova característica: área da sépala

df['sepal_area'] = df['sepal_length'] * df['sepal_width']

df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species	sepal_area
0	5.1	3.5	1.4	0.2	Iris-setosa	17.85
1	4.9	3.0	1.4	0.2	Iris-setosa	14.70
2	4.7	3.2	1.3	0.2	Iris-setosa	15.04
3	4.6	3.1	1.5	0.2	Iris-setosa	14.26
4	5.0	3.6	1.4	0.2	Iris-setosa	18.00

Desafios 3

1. **Limpeza de Dados**: Introduza valores faltantes no dataset Iris. Tente usar diferentes métodos para tratar essas imperfeições e compare os resultados.
2. **Manipulação de Dados**: Use as funções do pandas para responder às seguintes perguntas sobre o dataset Iris:
    - Qual é a média da largura da sépala para cada espécie?
    - Quantas flores têm uma área da sépala maior que 20 cm^2?
3. **feature engineering**: Pense em outras características que podem ser criadas a partir do dataset Iris. Por exemplo, uma característica que represente a proporção entre a largura e o comprimento da sépala.

```
## R1

import pandas as pd
import numpy as np
```

```
# Caminho do arquivo
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Lê e carrega o arquivo para a memória
df = pd.read_csv(url, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])

# Introduzindo valores faltantes de forma aleatória em 10% das células
np.random.seed(0)  # Define a semente para reprodutibilidade
percentual_faltante = 0.1  # 10% de valores faltantes
tamanho_amostra = int(percentual_faltante * df.size)
indices_para_faltantes = np.random.choice(df.size, tamanho_amostra, replace=False)
df.values.flat[indices_para_faltantes] = np.nan

# Verifica se há dados faltantes
dados_faltantes = df.isnull()

# Conta os valores nulos em cada coluna
contagem_nulos_por_coluna = dados_faltantes.sum()

print("Dados faltantes por coluna:")
print(contagem_nulos_por_coluna)

## R2.1
import pandas as pd

# Caminho do arquivo
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Lê e carrega o arquivo para a memória
df_iris = pd.read_csv(url, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])

# Calcula a média das larguras das sépalas por espécie
grouped = df_iris.groupby('species')['sepal_width'].mean()

# Exibe as médias das larguras das sépalas por espécie
print("Médias das larguras das sépalas por espécie:")
print(grouped)

## R2.2 - 41 flores

import pandas as pd

# Caminho do arquivo
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Lê e carrega o arquivo para a memória
df_iris = pd.read_csv(url, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])

# Calcula a área da sépala (sepal_area) multiplicando o comprimento pela largura
df_iris['sepal_area'] = df_iris['sepal_length'] * df_iris['sepal_width']

# Conta quantas flores têm uma área de sépala maior do que 20 cm²
flores_com_area_maior_que_20 = df_iris[df_iris['sepal_area'] > 20].shape[0]

# Exibe o número de flores com área de sépala maior que 20 cm²
print(f"Número de flores com área de sépala maior que 20 cm²: {flores_com_area_maior_que_20}")

## R3

import pandas as pd

# Caminho do arquivo
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Lê e carrega o arquivo para a memória
df_iris = pd.read_csv(url, header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'])

# Calcula a proporção entre a largura e o comprimento da sépala e adiciona como uma nova coluna
df_iris['sepal_width_length_ratio'] = df_iris['sepal_width'] / df_iris['sepal_length']

# Exibe as primeiras linhas do DataFrame com a nova coluna
print(df_iris[['sepal_width', 'sepal_length', 'sepal_width_length_ratio']])
```

```
Dados faltantes por coluna:
sepal_length    0
```

```
sepal_width     0
petal_length    0
petal_width     0
class           0
dtype: int64
Médias das larguras das sépalas por espécie:
species
Iris-setosa        3.418
Iris-versicolor    2.770
Iris-virginica     2.974
Name: sepal_width, dtype: float64
Número de flores com área de sépala maior que 20 cm²: 41
     sepal_width  sepal_length  sepal_width_length_ratio
0            3.5           5.1                  0.686275
1            3.0           4.9                  0.612245
2            3.2           4.7                  0.680851
3            3.1           4.6                  0.673913
4            3.6           5.0                  0.720000
..           ...           ...                       ...
145          3.0           6.7                  0.447761
146          2.5           6.3                  0.396825
147          3.0           6.5                  0.461538
148          3.4           6.2                  0.548387
149          3.0           5.9                  0.508475

[150 rows x 3 columns]
```

Desafio 4

Faça agora uma exploração de dados em outra base, conheça a base, e crie hipoteses de teste.

1. Importar os dados do dataset `Breast Cancer Data Set` acesse o site: `https://archive.ics.uci.edu/ml/datasets/breast+cancer`
2. Nomear as colunas de acordo com o arquivo `breast-cancer.names`
3. Ralizar a Análise Exploratória de Dados (EDA):
4. Visualize a distribuição de cada característica do conjunto de dados.
5. Determine quais características têm maior correlação com a classificação de malignidade.
6. Crie novas características a partir das existentes.
7. Crie representaçãoes gráficas do dataset para contribuir para sua análise

```
### R1

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Caminho do arquivo
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer/breast-cancer.data"

# Lê e carrega o arquivo para a memória
df = pd.read_csv(url, header=None)

# Exibe as primeiras linhas do DataFrame
print(df.head())

## R2

# Nomes das colunas
nomes_colunas = ['Class', 'Age', 'Menopause', 'Tumor Size', 'Inv Nodes', 'Node Caps',
                 'Deg Malig', 'Breast', 'Breast Quad', 'Irradiat']

# Lê e carrega o arquivo para a memória, utilizando os nomes das colunas fornecidos
df = pd.read_csv(url, header=None, names=nomes_colunas)

# Exibe as primeiras linhas do DataFrame com os nomes das colunas
print(df.head())


## R3


# Lê e carrega o arquivo para a memória, utilizando os nomes das colunas fornecidos
df = pd.read_csv(url, header=None, names=nomes_colunas)

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

```
# R4
# Resumo estatístico do DataFrame
print("\nResumo estatístico do DataFrame:")
print(df.describe())

# Contagem de valores únicos em cada coluna
print("\nContagem de valores únicos em cada coluna:")
print(df.nunique())

# Contagem de valores em 'Class'
print("\nContagem de valores em 'Class':")
print(df['Class'].value_counts())

# Histograma de idades
plt.figure(figsize=(8, 6))
sns.histplot(df['Age'], bins=20, kde=True)
plt.title('Distribuição de Idades')
plt.xlabel('Idade')
plt.ylabel('Frequência')
plt.show()

# Gráfico de contagem para 'Class'
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Class')
plt.title('Distribuição das Classes')
plt.xlabel('Classe')
plt.ylabel('Contagem')
plt.show()

#R5

# Nomes das colunas
nomes_colunas = ['Class', 'Age', 'Menopause', 'Tumor Size', 'Inv Nodes', 'Node Caps',
                 'Deg Malig', 'Breast', 'Breast Quad', 'Irradiat']

# Lê e carrega o arquivo para a memória, utilizando os nomes das colunas fornecidos
df = pd.read_csv(url, header=None, names=nomes_colunas)

# Converte a coluna 'Class' para valores numéricos (0 para benigno, 1 para maligno)
df['Class'] = df['Class'].apply(lambda x: 1 if x == 'malignant' else 0)

# Calcula as correlações absolutas com a coluna 'Deg Malig'
correlacoes_deg_malig = df.corr(method='pearson')['Deg Malig'].abs().sort_values(ascending=False)

# Exibe as características com maiores correlações absolutas com 'Deg Malig'
print("Características com maiores correlações com 'Deg Malig':")
print(correlacoes_deg_malig)

# R.6
# Encontrar a segunda maior correlação em relação a 'Deg Malig'
segunda_maior_correlacao = correlacoes_deg_malig.index[1]

# Exibir a segunda maior correlação
print(f"Segunda maior correlação com 'Deg Malig': {segunda_maior_correlacao}")

# Exibir as primeiras linhas do DataFrame para observar os dados relacionados a essa correlação
print(df[[segunda_maior_correlacao, 'Deg Malig']].head())

# R.7

# Gráfico de dispersão

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Tumor Size', y='Inv Nodes', hue='Class', palette='coolwarm')
plt.title('Gráfico de Dispersão: Tumor Size vs Inv Nodes')
plt.xlabel('Tumor Size')
plt.ylabel('Inv Nodes')
plt.show()

# BoxPlot
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='Class', y='Tumor Size', palette='Set2')
plt.title('Boxplot: Tumor Size por Classe')
plt.xlabel('Class')
plt.ylabel('Tumor Size')
plt.show()
```

```
# Barras empilhadas
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Menopause', hue='Class', palette='coolwarm')
plt.title('Gráfico de Barras Empilhadas: Menopause por Classe')
plt.xlabel('Menopause')
plt.ylabel('Contagem')
plt.show()
```