Visualizing Data in Python

When working with a new dataset, one of the most useful things to do is to begin to visualize the data. By using **tables**, **histograms**, **boxplots**, **scatter plots** and other visual tools, we can get a better idea of what the data may be trying to tell us, and we can gain insights into the data that we may have not discovered otherwise.

In this notebook will use the Seaborn data processing library, which is a higher-level interface to **Matplotlib** that can be used to simplify many visualization tasks

The **Seaborn** provides visualisations tools that will allow to explore data from a graphical perspective.

Acknowledgments

- Data from https://www.coursera.org/ from the course "Understanding and Visualizing Data with Python" by University of Michigan

Importing libraries

```
# Import the packages that we will be using
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing data

```
# Define where you are running the code: colab or local
RunInColab          = True     # (False: no  | True: yes)

# If running in colab:
if RunInColab:
    # Mount your google drive in google colab
    from google.colab import drive
    drive.mount('/content/drive')

    # Find location
    #!pwd
    #!ls
    #!ls "/content/drive/My Drive/Colab Notebooks/MachineLearningWithPython/"

    # Define path del proyecto
    Ruta            = "/content/drive/My Drive/Colab Notebooks/NotebooksProfessor"

else:
    # Define path del proyecto
    Ruta            = ""
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# url string that hosts our .csv file
url = Ruta + "/datasets/cartwheel/cartwheel.csv"

# Read the .csv file and store it as a pandas Data Frame
df = pd.read_csv(url)
```

Exploring the content of the data set

Get a general 'feel' of the data

```
df
```

```
20  21  23.0   M   2   Y   1   69.00   67.0    66
21  22  29.0   M   2   N   0   71.00   70.0   101
22  23  25.0   M   2   N   0   70.00   68.0    82
23  24  26.0   M   2   N   0   69.00   71.0    63
24  25  23.0   F   1   Y   1   65.00   63.0    67
25  26  28.0   M   2   N   0   75.00   76.0   111
26  27  24.0   M   2   N   0   78.40   71.0    92
27  28  25.0   M   2   Y   1   76.00   73.0   107
28  29  32.0   F   1   Y   1   63.00   60.0    75
29  30  38.0   F   1   Y   1   61.50   61.0    78
30  31  27.0   F   1   Y   1   62.00   60.0    72
31  32  33.0   F   1   Y   1   65.30   64.0    91
32  33  38.0   F   1   N   0   64.00   63.0    86
33  34  27.0   M   2   N   0   77.00   75.0   100
34  35  24.0   F   1   N   0   67.80   62.0    98
35  36  27.0   M   2   N   0   68.00   66.0    74
36  37  25.0   F   1   Y   1   65.00   64.5    92
37  38  26.0   F   1   N   0   61.50   59.5    90
38  39  31.0   M   2   Y   1   73.00   74.0    72
39  40  30.0   M   2   Y   1   69.50   66.0    96
40  41  23.0   F   1   N   0   70.40   71.0    66
41  42  26.0   M   2   Y   1   73.50   72.0   115
42  43  28.0   F   1   Y   1   72.50   72.0    81
43  44  26.0   F   1   Y   1   72.00   72.0    92
44  45  30.0   F   1   Y   1   66.00   64.0    85
45  46  39.0   F   1   N   0   64.00   63.0    87
```

46	47	27.0	M	2	N	0	78.00	75.0	72
47	48	24.0	M	2	N	0	79.50	75.0	82
48	49	28.0	M	2	N	0	77.80	76.0	99
49	50	30.0	F	1	N	0	74.60	NaN	71
50	51	NaN	M	2	N	0	71.00	70.0	101
51	52	27.0	M	2	N	0	NaN	71.5	103

Pasos siguientes: Generar código con df | Ver gráficos recomendados | New interactive sheet

Frequency tables

The `value_counts()` method can be used to determine the number of times that each distinct value of a variable occurs in a data set. In statistical terms, this is the "frequency distribution" of the variable. The `value_counts()` method produces a table with two columns. The first column contains all distinct observed values for the variable. The second column contains the number of times each of these values occurs. Note that the table returned by `value_counts()` is actually a **Pandas** data frame, so can be further processed using any Pandas methods for working with data frames.

```
# Number of times that each distinct value of a variable occurs in a data set
df.CompleteGroup.value_counts()
```

CompleteGroup	count
1.0	43
0.0	8

dtype: int64

```
# Proportion of each distinct value of a variable occurs in a data set
x = df.CompleteGroup.value_counts(normalize=True)
x = x * 100
x
```

CompleteGroup	proportion
1.0	84.313725
0.0	15.686275

dtype: float64

Note that the `value_counts()` method excludes missing values. We confirm this below by adding up observations to your data frame with some missing values and then computing `value_counts()` and comparing this to the total number of rows in the data set, which is 28. This tells us that there are 28 - (21+6) = 1 missing values for this variable (other variables may have different numbers of missing values).

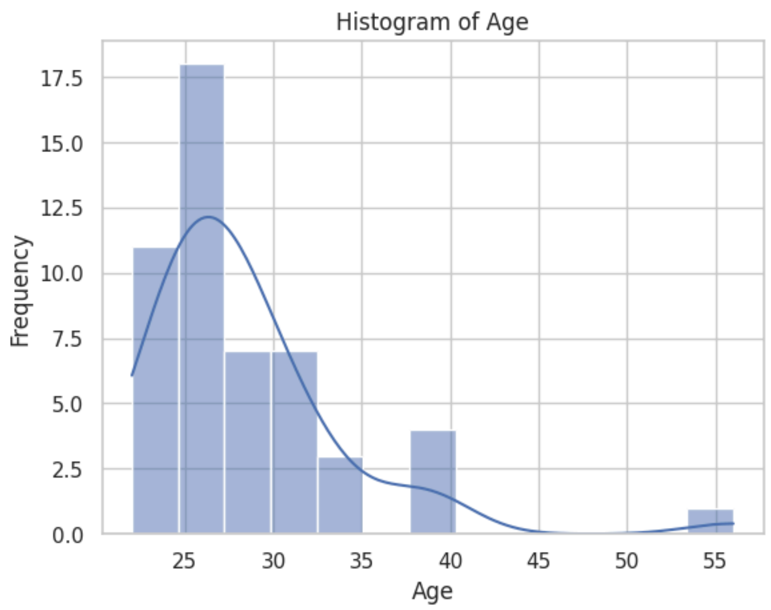
```
# Total number of observations
print("Total number of observations: " + str(df.shape[0]))


# Total number of null observations
print("Total number of null observations: " + str(df.Age.isnull().sum()))


# Total number of counts (excluding missing values)
print("Total number of counts : " + str(df.Age.notnull().sum()))
```

```
Total number of observations: 52
Total number of null observations: 1
Total number of counts : 51
```

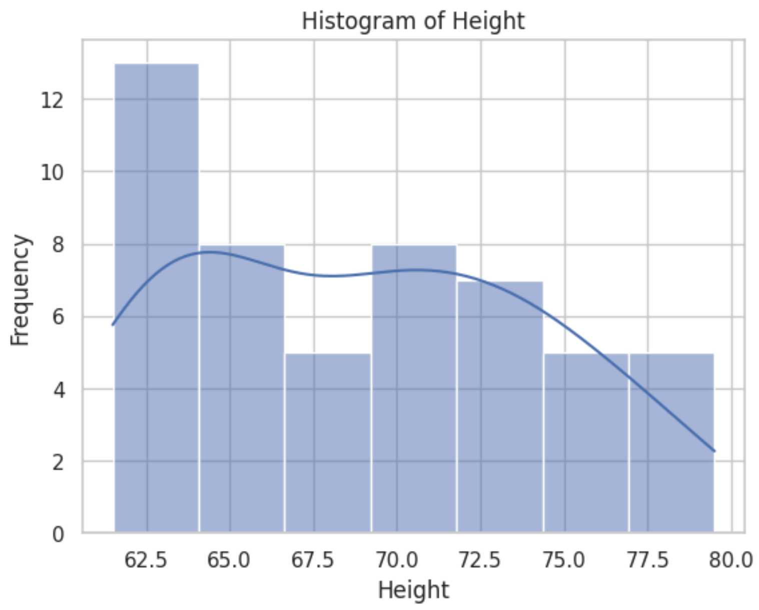
Histogram

It is often good to get a feel for the shape of the distribution of the data.

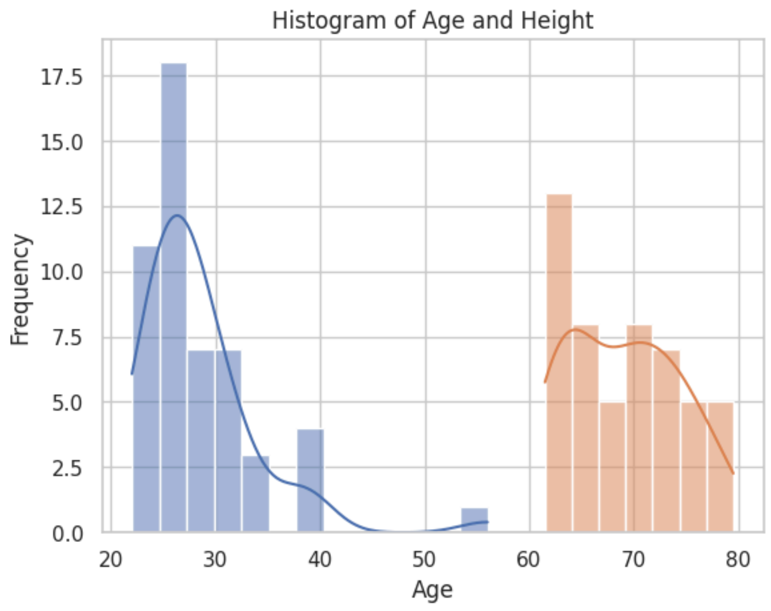
```
# Plot histogram of the total bill only
var2plot = df["Age"]

sns.histplot(var2plot, kde=True)
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Histogram of Age")
plt.show()
```



```
# Plot distribution of the tips only
var2plot = df["Height"]

sns.histplot(var2plot, kde=True)
plt.xlabel("Height")
plt.ylabel("Frequency")
plt.title("Histogram of Height")
plt.show()
```



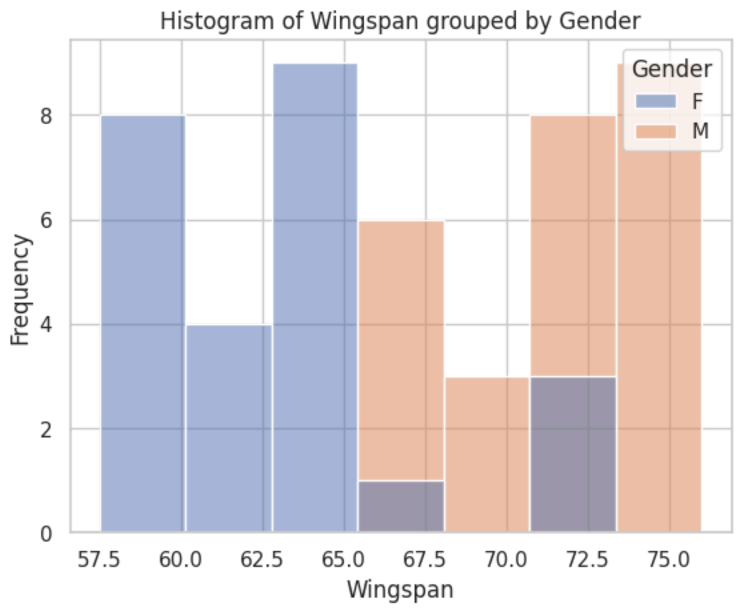
```
# Plot histogram of both the Age and the Wingspan
var2plot = df["Age"]
var2plot2 = df["Height"]

sns.histplot(var2plot, kde=True)
sns.histplot(var2plot2, kde=True)
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Histogram of Age and Height")
plt.show()
```



Histograms plotted by groups

While looking at a single variable is interesting, it is often useful to see how a variable changes in response to another. Thus, we can create a histograms of one quantitative variable grouped by another categorical variables.

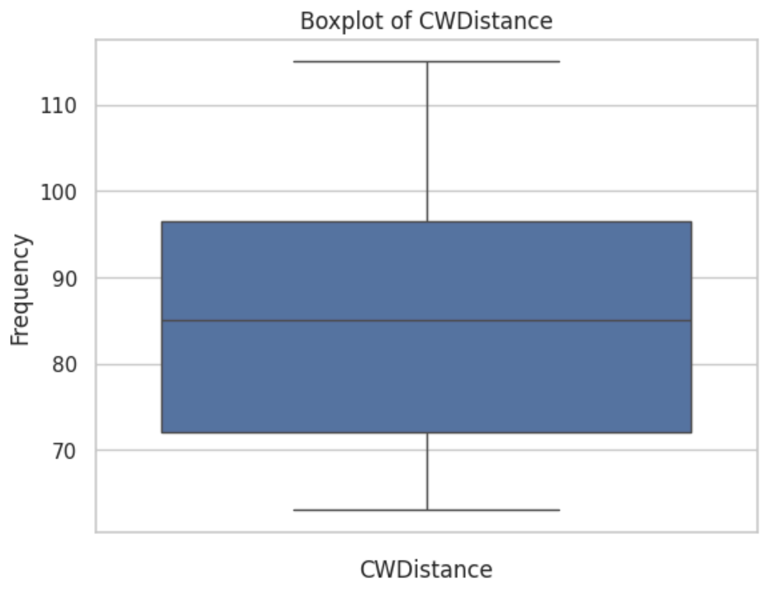
```
# Create histograms of the "Wingspan" grouped by "Gender"
sns.histplot(data=df, x="Wingspan", hue="Gender")
plt.xlabel("Wingspan")
plt.ylabel("Frequency")
plt.title("Histogram of Wingspan grouped by Gender")
plt.show()
```



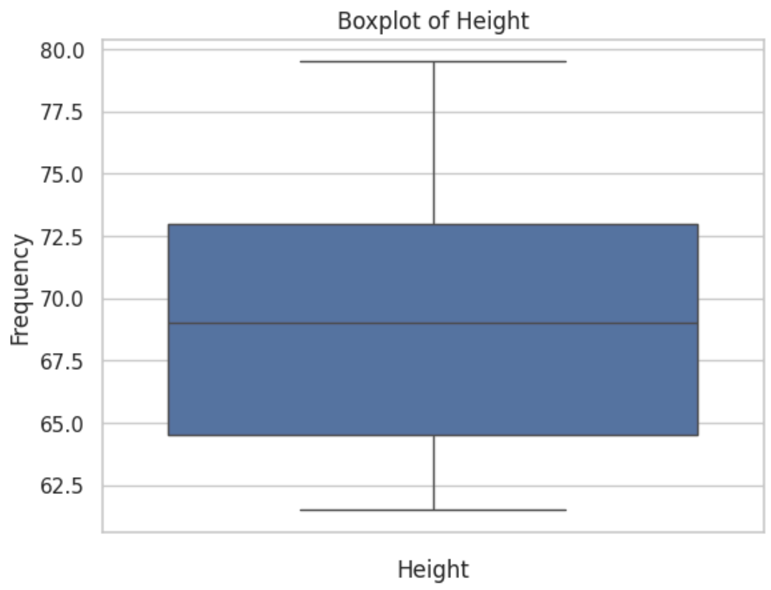
Boxplots

Boxplots do not show the shape of the distribution, but they can give us a better idea about the center and spread of the distribution as well as any potential outliers that may exist. Boxplots and Histograms often complement each other and help an analyst get more information about the data

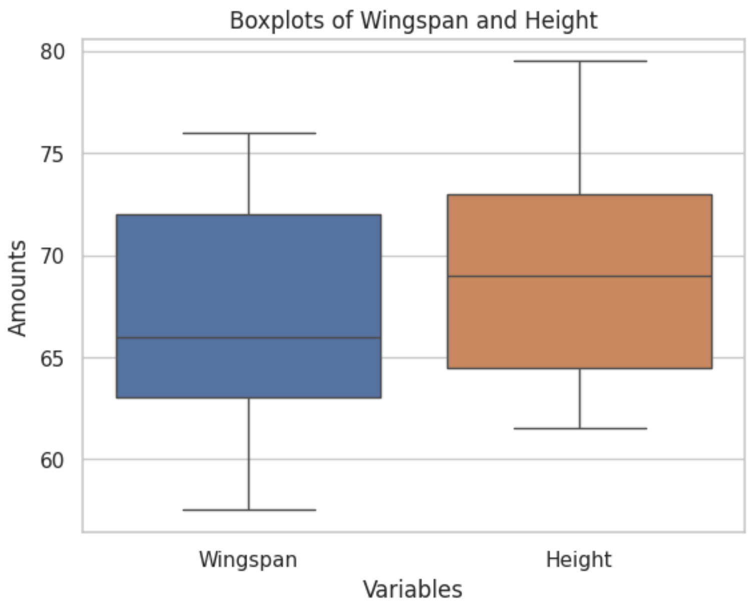
```
# Create the boxplot of the "CWDistance" amounts
sns.boxplot(data = df, y = "CWDistance")
plt.xlabel("CWDistance")
plt.ylabel("Frequency")
plt.title("Boxplot of CWDistance")
plt.show()
```



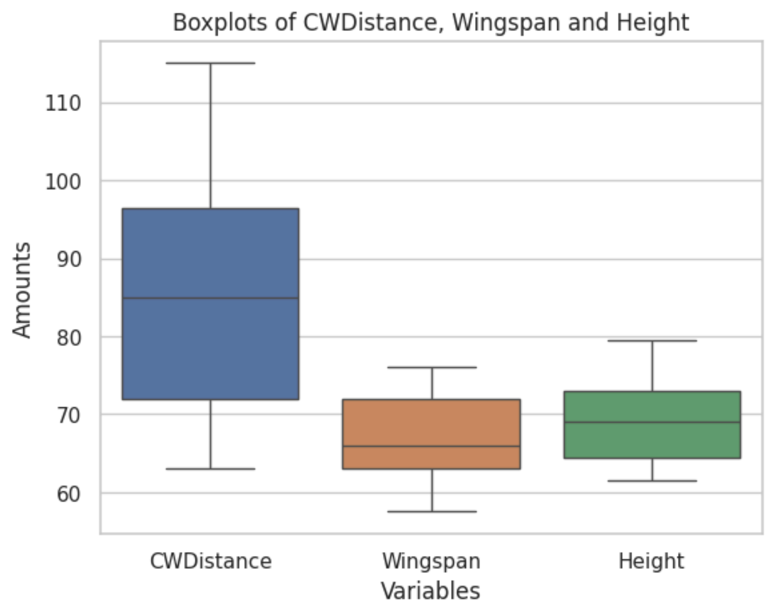
```
# Create the boxplot of the "Height" amounts
sns.boxplot(data = df, y = "Height")
plt.xlabel("Height")
plt.ylabel("Frequency")
plt.title("Boxplot of Height")
plt.show()
```



```
# Create the boxplots of the "Wingspan" and of the "Height" amounts
df_variables = df[["Wingspan", "Height"]]

sns.boxplot(data = df_variables)
plt.xlabel("Variables")
plt.ylabel("Amounts")
plt.title("Boxplots of Wingspan and Height")
plt.show()
```



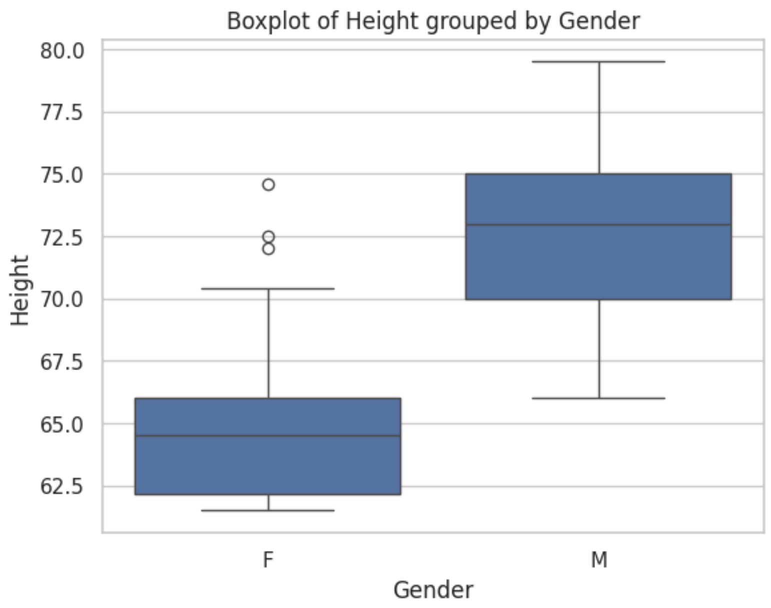
```
# Create the boxplots of the "CWDistance", "Wingspan" and of the "Height" amounts
df_variables = df[["CWDistance", "Wingspan", "Height"]]

sns.boxplot(data = df_variables)
plt.xlabel("Variables")
plt.ylabel("Amounts")
plt.title("Boxplots of CWDistance, Wingspan and Height")
plt.show()
```



Boxplots plotted by groups

While looking at a single variable is interesting, it is often useful to see how a variable changes in response to another. Thus, we can create a side-by-side boxplots of one quantitative variable grouped by another categorical variables.

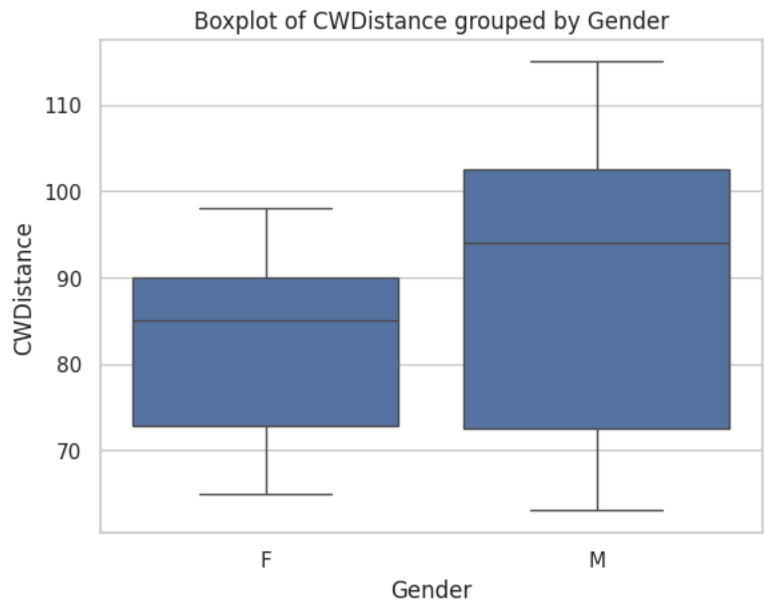
```
# Create side-by-side boxplots of the "Height" grouped by "Gender"
df_variables = df[["Height", "Gender"]]

sns.boxplot(data=df_variables, x="Gender", y="Height")
plt.xlabel("Gender")
plt.ylabel("Height")
plt.title("Boxplot of Height grouped by Gender")
plt.show()
```

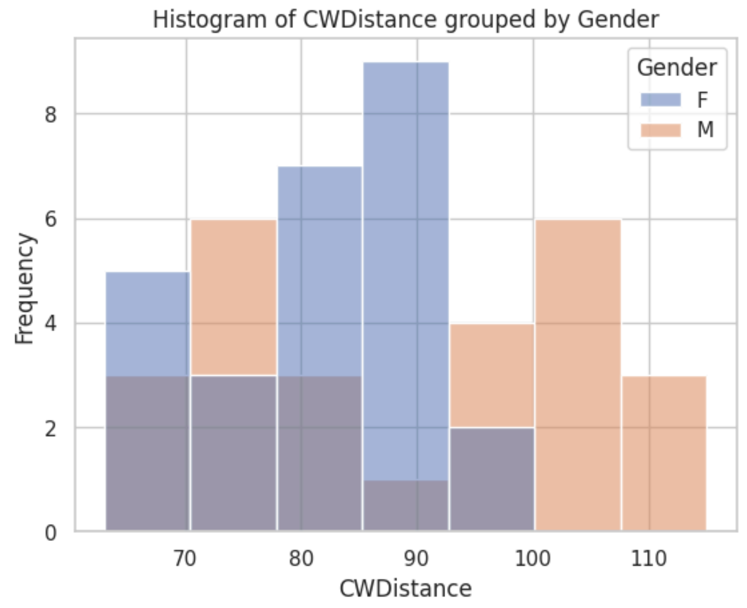


Histograms and boxplots plotted by groups

We cal also create both boxplots and histograms of one quantitative variable grouped by another categorical variables

```
# Create a boxplot and histogram of the "CWDistance" grouped by "Gender"
df_variables = df[["CWDistance", "Gender"]]

sns.boxplot(data=df_variables, x="Gender", y="CWDistance")
plt.xlabel("Gender")
plt.ylabel("CWDistance")
plt.title("Boxplot of CWDistance grouped by Gender")
plt.show()

sns.histplot(data=df_variables, x="CWDistance", hue="Gender")
plt.xlabel("CWDistance")
plt.ylabel("Frequency")
plt.title("Histogram of CWDistance grouped by Gender")
plt.show()
```

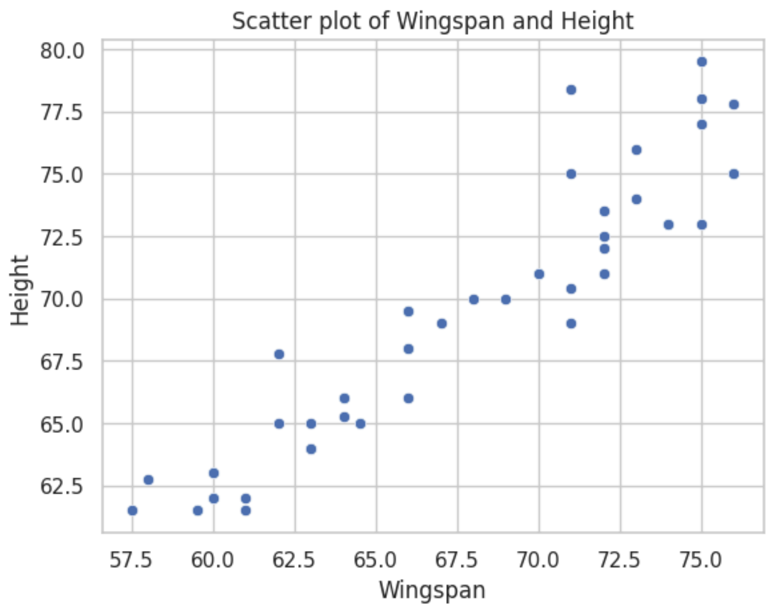




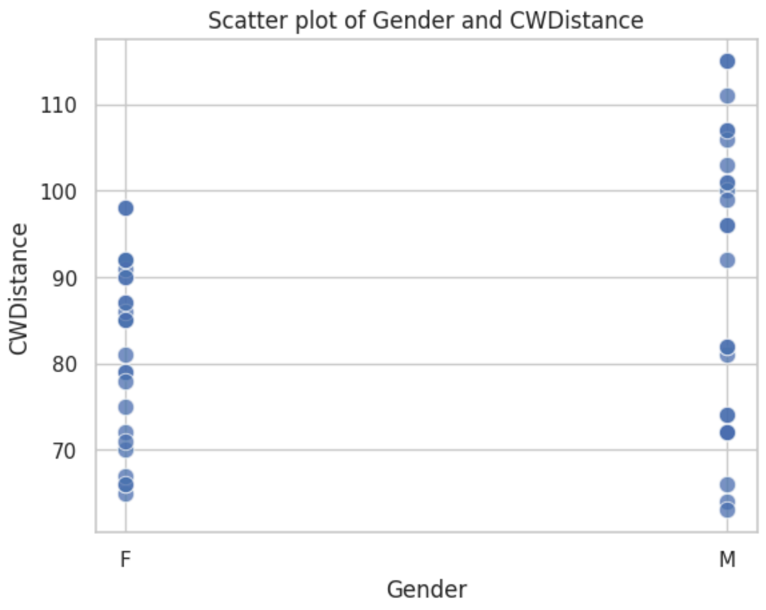
Scatter plot

Plot values of one variable versus another variable to see how they are correlated

```
# scatter plot between two variables
sns.scatterplot(data=df, x="Wingspan", y="Height")
plt.xlabel("Wingspan")
plt.ylabel("Height")
plt.title("Scatter plot of Wingspan and Height")
plt.show()
```

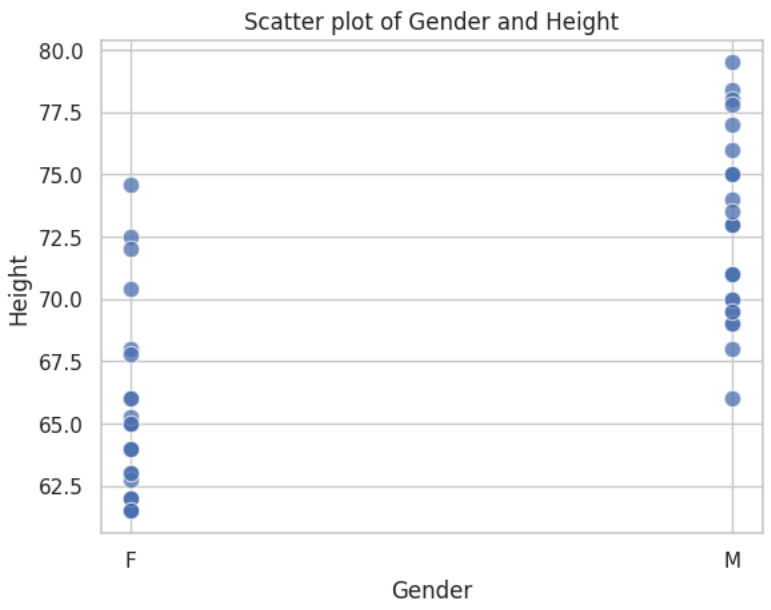


```
# scatter plot between two variables (one categorical)
sns.scatterplot(data=df, x="Gender", y="CWDistance", s = 80, alpha = 0.75)
plt.xlabel("Gender")
plt.ylabel("CWDistance")
plt.title("Scatter plot of Gender and CWDistance")
plt.show()
```



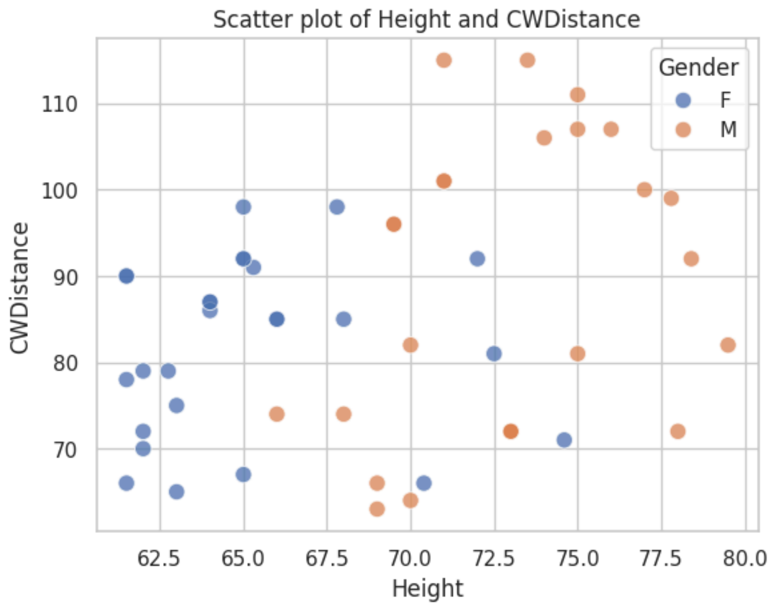
```
# scatter plot between two variables (one categorical)
sns.scatterplot(data=df, x="Gender", y="Height", s = 80, alpha = 0.75)
plt.xlabel("Gender")
plt.ylabel("Height")
plt.title("Scatter plot of Gender and Height")
```

```
Text(0.5, 1.0, 'Scatter plot of Gender and Height')
```



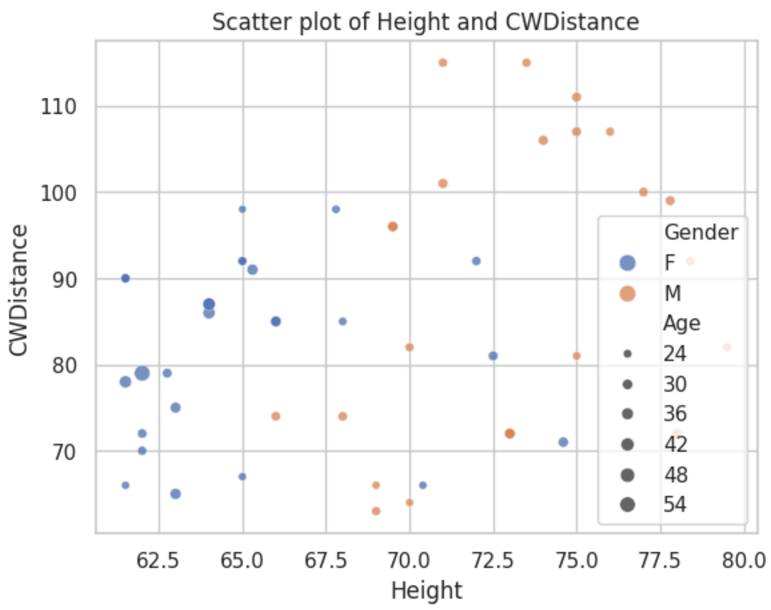
```
# scatter plot between two variables grouped according to a categorical variable
sns.scatterplot(data=df, x="Height", y="CWDistance", hue="Gender", s = 80, alpha = 0.75)
plt.xlabel("Height")
plt.ylabel("CWDistance")
plt.title("Scatter plot of Height and CWDistance")
```

```
Text(0.5, 1.0, 'Scatter plot of Height and CWDistance')
```



```
# scatter plot between two variables grouped according to a categorical variable and with size of markers
sns.scatterplot(data=df, x="Height", y="CWDistance", hue="Gender", size="Age", s = 80, alpha = 0.75)
plt.xlabel("Height")
plt.ylabel("CWDistance")
plt.title("Scatter plot of Height and CWDistance")
```

```
Text(0.5, 1.0, 'Scatter plot of Height and CWDistance')
```



Final remarks

- Visualizing your data using **tables**, **histograms**, **boxplots**, **scatter plots** and other tools is essential to carry put analysis and extract conclusions
- There are several ways to do the same thing
- The **Seaborn** package provides visualisations tools that allow to explore data from a graphical perspective

Activity: work with the iris dataset

Repeat this tutorial with the iris data set and respond to the following inquiries

1. Plot the histograms for each of the four quantitative variables
2. Plot the histograms for each of the quantitative variables
3. Plot the boxplots for each of the quantitative variables
4. Plot the boxplots of the petal width grouped by type of flower
5. Plot the boxplots of the setal length grouped by type of flower
6. Provide a description (explaination from your observations) of each of the quantitative variables

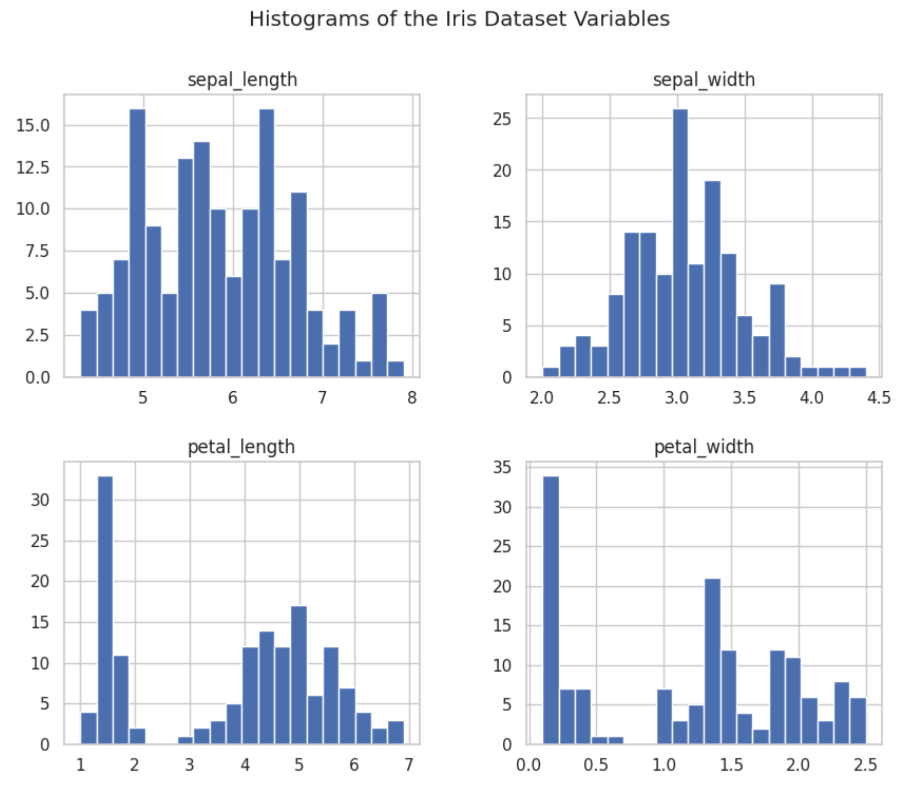
```
# Import necessary libraries
import seaborn as sns

# Load the Iris dataset from seaborn
df = sns.load_dataset('iris')

# Set a style for the plots
sns.set(style="whitegrid")

df
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
149	5.9	3.0	5.1	1.8	virginica

Pasos siguientes: Generar código con df | Ver gráficos recomendados | New interactive sheet

Plot Histograms for Each Quantitative Variable

```
# Create histograms for each quantitative variable
df.hist(bins=20, figsize=(10, 8))
plt.suptitle("Histograms of the Iris Dataset Variables")
plt.show()
```



Plot Boxplots for Each Quantitative Variable

No se ha podido establecer conexión con el servicio reCAPTCHA. Comprueba tu conexión a Internet y vuelve a cargar la página para ver otro reCAPTCHA.