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

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/"

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 + "A01641179/datasets/cartwheel/cartwheel.csv"

# Read the .csv file and store it as a pandas Data Frame

dt = pd.read_csv(url )
```

Exploring the content of the data set

Get a general 'feel' of the data

```
dt.head()
```

	ID	Age	GenderGroup	GlassesGroup	Height	Wingspan	CWDistance	CompleteGroup	Score
count	52.000000	51.000000	52.000000	52.000000	51.000000	51.000000	52.000000	51.000000	52.000000
mean	26.500000	28.411765	1.500000	0.500000	68.971569	67.313725	85.576923	0.843137	7.173077
std	15.154757	5.755611	0.504878	0.504878	5.303812	5.624021	14.353173	0.367290	2.211566
min	1.000000	22.000000	1.000000	0.000000	61.500000	57.500000	63.000000	0.000000	2.000000
25%	13.750000	25.000000	1.000000	0.000000	64.500000	63.000000	72.000000	1.000000	6.000000
50%	26.500000	27.000000	1.500000	0.500000	69.000000	66.000000	85.000000	1.000000	8.000000
75%	39.250000	30.000000	2.000000	1.000000	73.000000	72.000000	96.500000	1.000000	9.000000

```
dt.shape
```

```
(52, 12)
```

```
dt.describe()
```

	ID	Age	GenderGroup	GlassesGroup	Height	Wingspan	CWDistance	CompleteGroup	Score
count	52.000000	51.000000	52.000000	52.000000	51.000000	51.000000	52.000000	51.000000	52.000000
mean	26.500000	28.411765	1.500000	0.500000	68.971569	67.313725	85.576923	0.843137	7.173077
std	15.154757	5.755611	0.504878	0.504878	5.303812	5.624021	14.353173	0.367290	2.211566
min	1.000000	22.000000	1.000000	0.000000	61.500000	57.500000	63.000000	0.000000	2.000000
25%	13.750000	25.000000	1.000000	0.000000	64.500000	63.000000	72.000000	1.000000	6.000000
50%	26.500000	27.000000	1.500000	0.500000	69.000000	66.000000	85.000000	1.000000	8.000000
75%	39.250000	30.000000	2.000000	1.000000	73.000000	72.000000	96.500000	1.000000	9.000000
max	52.000000	56.000000	2.000000	1.000000	79.500000	76.000000	115.000000	1.000000	10.000000

```
vars = ["Age","GenderGroup","GlassesGroup","Height","Wingspan","CWDistance","CompleteGroup","Score"]
dt_dos = dt[vars].dropna()
```

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
count = dt['Age'].value_counts()
print(count)
count = dt['Gender'].value_counts()
print(count)
count = dt['GenderGroup'].value_counts()
print(count)
count = dt['GlassesGroup'].value_counts()
print(count)
count = dt['Height'].value_counts()
print(count)
count = dt['Wingspan'].value_counts()
print(count)
count = dt['CWDistance'].value_counts()
print(count)
count = dt['Complete'].value_counts()
print(count)
count = dt['CompleteGroup'].value_counts()
print(count)
count = dt['Score'].value_counts()
print(count)
```

```
73.0    2
64.5    2
62.0    2
70.0    2
68.0    1
67.0    1
58.0    1
69.0    1
57.5    1
71.5    1
Name: Wingspan, dtype: int64
72     5
92     4
85     3
66     3
79     2
96     2
82     2
101    2
90     2
115    2
74     2
98     2
107    2
81     2
87     2
65     1
78     1
71     1
99     1
100    1
86     1
91     1
67     1
75     1
111    1
106    1
63     1
64     1
70     1
103    1
Name: CWDistance, dtype: int64
Y    44
N     8
Name: Complete, dtype: int64
1.0    43
0.0     8
Name: CompleteGroup, dtype: int64
8     11
10     9
7      7
9      7
6      6
4      4
5      4
3      3
2      1
Name: Score, dtype: int64
```

```
# Proportion of each distinct value of a variable occurs in a data set

proportion_age = dt['Age'].value_counts(normalize=True)
proportion_gender = dt['Gender'].value_counts(normalize=True)
proportion_gendergroup = dt['GenderGroup'].value_counts(normalize=True)
proportion_glassesgroup = dt['GlassesGroup'].value_counts(normalize=True)
proportion_height = dt['Height'].value_counts(normalize=True)
proportion_wingspan = dt['Wingspan'].value_counts(normalize=True)
proportion_cwdistance = dt['CWDistance'].value_counts(normalize=True)
proportion_complete = dt['Complete'].value_counts(normalize=True)
proportion_completegroup = dt['CompleteGroup'].value_counts(normalize=True)
proportion_score = dt['Score'].value_counts(normalize=True)

print(proportion_age)
print(proportion_gender)
print(proportion_gendergroup)
print(proportion_glassesgroup)
print(proportion_height)
print(proportion_wingspan)
print(proportion_cwdistance)
print(proportion_complete)
```

```
print(proportion_completegroup)
print(proportion_score)
```

```
26.0    0.137255
27.0    0.137255
24.0    0.098039
28.0    0.098039
23.0    0.098039
30.0    0.078431
25.0    0.078431
33.0    0.058824
39.0    0.039216
29.0    0.039216
31.0    0.039216
38.0    0.039216
56.0    0.019608
22.0    0.019608
32.0    0.019608
Name: Age, dtype: float64
F    0.5
M    0.5
Name: Gender, dtype: float64
1    0.5
2    0.5
Name: GenderGroup, dtype: float64
1    0.5
0    0.5
Name: GlassesGroup, dtype: float64
61.50    0.078431
65.00    0.078431
62.00    0.058824
64.00    0.058824
73.00    0.058824
75.00    0.058824
66.00    0.058824
71.00    0.058824
68.00    0.039216
70.00    0.039216
69.00    0.039216
63.00    0.039216
69.50    0.039216
62.75    0.019608
73.50    0.019608
77.80    0.019608
79.50    0.019608
78.00    0.019608
72.00    0.019608
72.50    0.019608
67.80    0.019608
70.40    0.019608
77.00    0.019608
65.30    0.019608
76.00    0.019608
78.40    0.019608
74.00    0.019608
74.60    0.019608
Name: Height, dtype: float64
66.0    0.098039
63.0    0.078431
75.0    0.078431
71.0    0.078431
```

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

observations = dt.count()

print("Numero de observaciones: ", observations)

# Total number of null observations

obsNULL = dt.isnull().sum()
print("Numeros de null observaciones: ", obsNULL)

# Total number of counts (excluding missing values)

counts = observations-obsNULL
```

```
print("Total: ", counts)
```

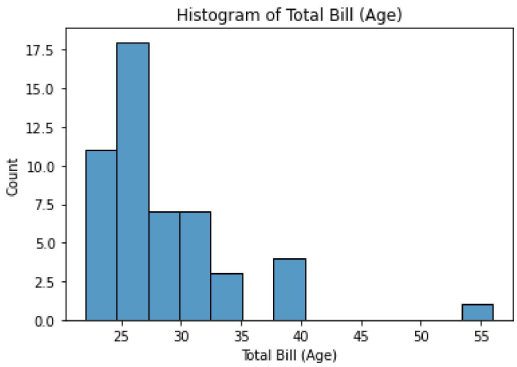
```
Numero de observaciones:  ID              52
Age             51
Gender          52
GenderGroup     52
Glasses         52
GlassesGroup    52
Height          51
Wingspan        51
CWDistance      52
Complete        52
CompleteGroup   51
Score           52
dtype: int64
Numeros de null observaciones:  ID              0
Age             1
Gender          0
GenderGroup     0
Glasses         0
GlassesGroup    0
Height          1
Wingspan        1
CWDistance      0
Complete        0
CompleteGroup   1
Score           0
dtype: int64
Total:  ID              52
Age             50
Gender          52
GenderGroup     52
Glasses         52
GlassesGroup    52
Height          50
Wingspan        50
CWDistance      52
Complete        52
CompleteGroup   50
Score           52
dtype: int64
```

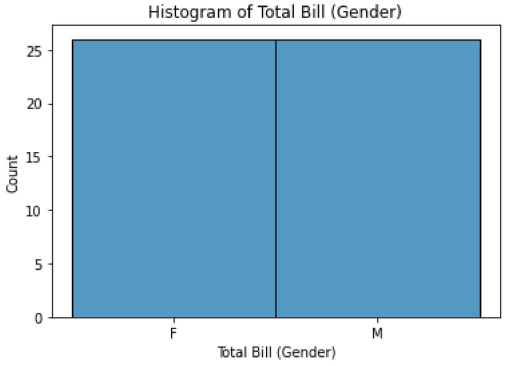
Histogram

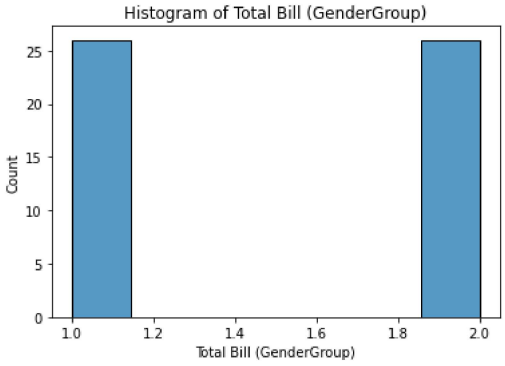
It is often good to get a feel for the shape of the distribution of the data.

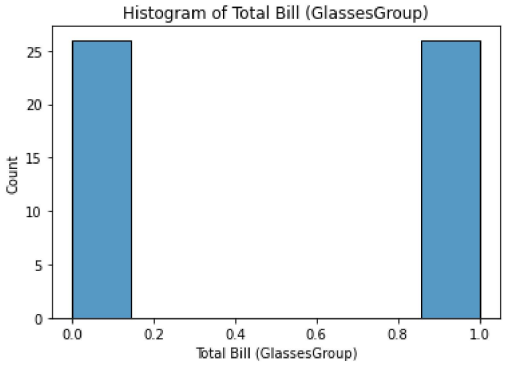
```
# Plot histogram of the total bill only
sns.histplot(data=dt, x="Age")
plt.title("Histogram of Total Bill (Age)")
plt.xlabel("Total Bill (Age)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="Gender")
plt.title("Histogram of Total Bill (Gender)")
plt.xlabel("Total Bill (Gender)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="GenderGroup")
plt.title("Histogram of Total Bill (GenderGroup)")
plt.xlabel("Total Bill (GenderGroup)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="GlassesGroup")
plt.title("Histogram of Total Bill (GlassesGroup)")
plt.xlabel("Total Bill (GlassesGroup)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="Height")
plt.title("Histogram of Total Bill (Height)")
plt.xlabel("Total Bill (Height)")
plt.ylabel("Count")
```

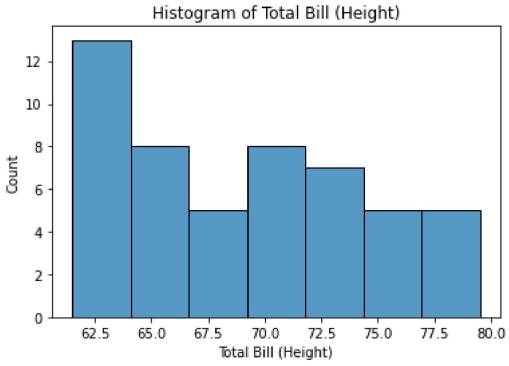
```
plt.show()

sns.histplot(data=dt, x="Wingspan")
plt.title("Histogram of Total Bill (Wingspan)")
plt.xlabel("Total Bill (Wingspan)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="CWDistance")
plt.title("Histogram of Total Bill (CWDistance)")
plt.xlabel("Total Bill (CWDistance)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="Complete")
plt.title("Histogram of Total Bill (Complete)")
plt.xlabel("Total Bill (Complete)")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="CompleteGroup")
plt.title("Histogram of Total Bill (CompleteGroup )")
plt.xlabel("Total Bill (CompleteGroup )")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="Score")
plt.title("Histogram of Total Bill (Score)")
plt.xlabel("Total Bill (Score)")
plt.ylabel("Count")
plt.show()
```

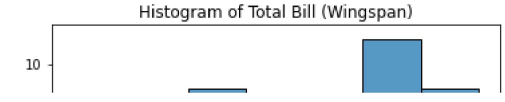

Histogram of Total Bill (Age)
Count
Total Bill (Age)


Histogram of Total Bill (Gender)
Count
F
M
Total Bill (Gender)

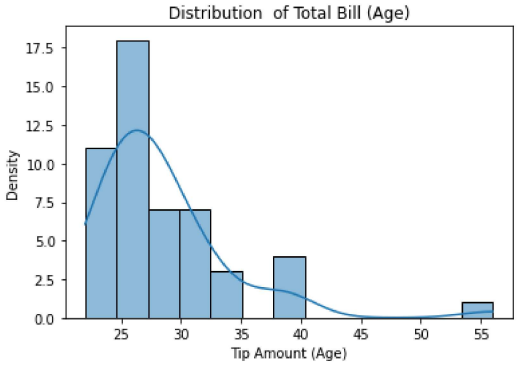

Histogram of Total Bill (GenderGroup)
Count
Total Bill (GenderGroup)

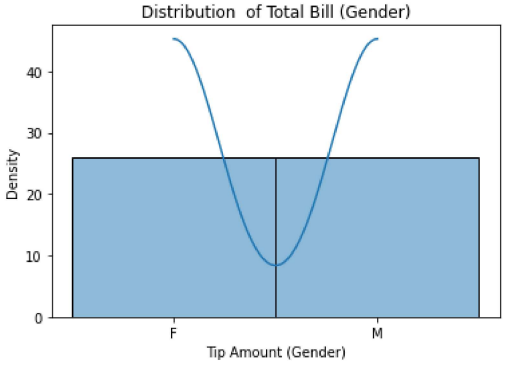

Histogram of Total Bill (GlassesGroup)
Count
Total Bill (GlassesGroup)

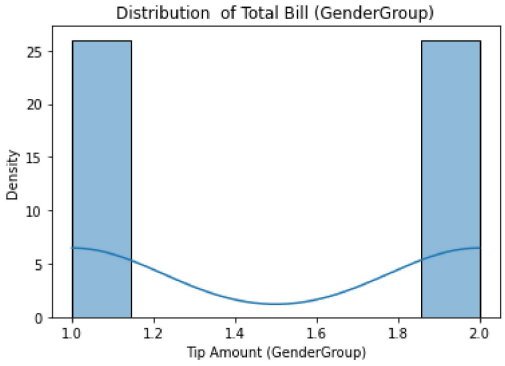

Histogram of Total Bill (Height)
Count
Total Bill (Height)

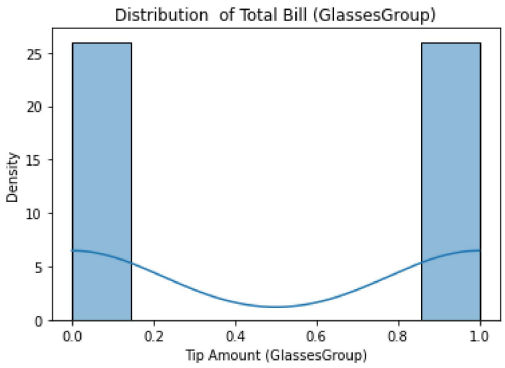

Histogram of Total Bill (Wingspan)

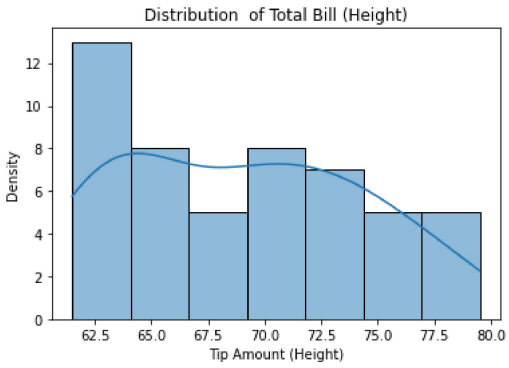
```
# Plot distribution of the tips only
sns.histplot(data=dt, x="Age", kde=True)
plt.title("Distribution  of Total Bill (Age)")
plt.xlabel("Tip Amount (Age)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="Gender", kde=True)
plt.title("Distribution  of Total Bill (Gender)")
plt.xlabel("Tip Amount (Gender)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="GenderGroup", kde=True)
plt.title("Distribution  of Total Bill (GenderGroup)")
plt.xlabel("Tip Amount (GenderGroup)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="GlassesGroup", kde=True)
plt.title("Distribution  of Total Bill (GlassesGroup)")
plt.xlabel("Tip Amount (GlassesGroup)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="Height", kde=True)
plt.title("Distribution  of Total Bill (Height)")
plt.xlabel("Tip Amount (Height)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="Wingspan", kde=True)
plt.title("Distribution  of Total Bill (Wingspan)")
plt.xlabel("Tip Amount (Wingspan)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="CWDistance", kde=True)
plt.title("Distribution  of Total Bill (CWDistance)")
plt.xlabel("Tip Amount (CWDistance)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="Complete", kde=True)
plt.title("Distribution  of Total Bill (Complete)")
plt.xlabel("Tip Amount (Complete)")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="CompleteGroup", kde=True)
plt.title("Distribution  of Total Bill (CompleteGroup )")
plt.xlabel("Tip Amount (CompleteGroup )")
plt.ylabel("Density")
plt.show()

sns.histplot(data=dt, x="Score", kde=True)
plt.title("Distribution  of Total Bill (Score)")
plt.xlabel("Tip Amount (Score)")
plt.ylabel("Density")
plt.show()
```

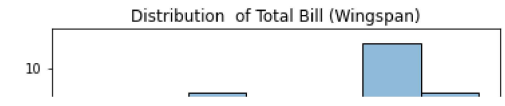

Distribution of Total Bill (Age)
Density
Tip Amount (Age)


Distribution of Total Bill (Gender)
Density
F
M
Tip Amount (Gender)

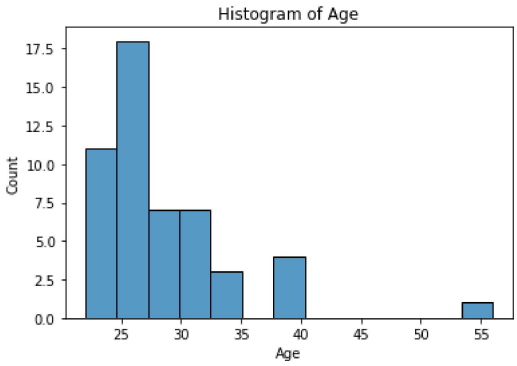

Distribution of Total Bill (GenderGroup)
Density
Tip Amount (GenderGroup)

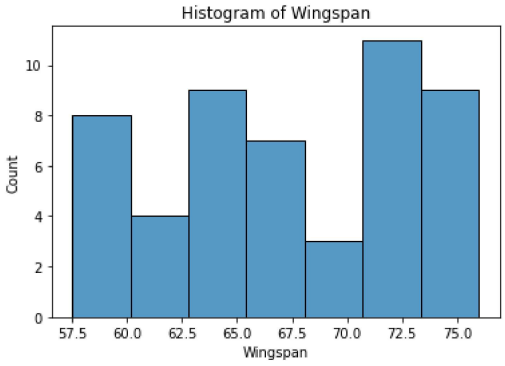

Distribution of Total Bill (GlassesGroup)
Density
Tip Amount (GlassesGroup)


Distribution of Total Bill (Height)
Density
Tip Amount (Height)


Distribution of Total Bill (Wingspan)

```
# Plot histogram of both the Age and the Wingspan

sns.histplot(data=dt, x="Age")
plt.title("Histogram of Age")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()

sns.histplot(data=dt, x="Wingspan")
plt.title("Histogram of Wingspan")
plt.xlabel("Wingspan")
plt.ylabel("Count")
plt.show()
```

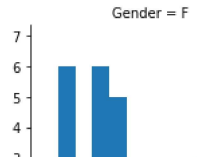




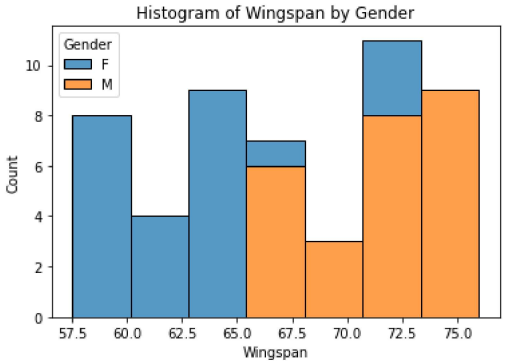
Histograms plotted by groups

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"

h = sns.FacetGrid(dt, row = "Gender")
h = h.map(plt.hist, "Wingspan")
plt.show()
```




```
# Create histograms of the "Wingspan" grouped by "Gender"

sns.histplot(data=dt, x="Wingspan", hue="Gender", multiple="stack")
plt.title("Histogram of Wingspan by Gender")
plt.xlabel("Wingspan")
plt.ylabel("Count")
plt.show()
```

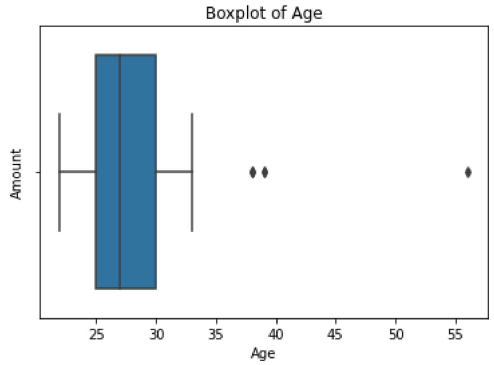


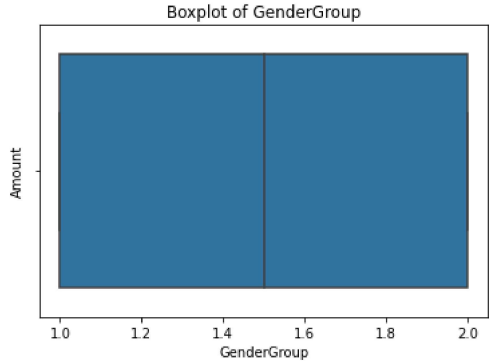
Boxplots

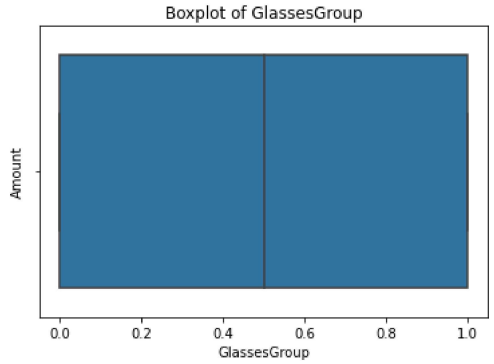
Boxplots do not show the shape of the distribution, but they can give us a better idea about the center and spread of the distribution as well as any potential outliers that may exist. Boxplots and Histograms often complement each other and help an analyst get more information about the data

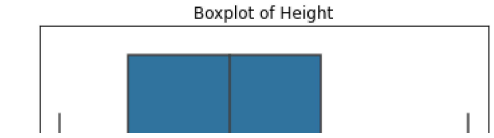
```
# Create the boxplot of the "total bill" amounts
sns.boxplot(x=dt["Age"])
plt.title("Boxplot of Age")
plt.xlabel("Age")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["GenderGroup"])
plt.title("Boxplot of GenderGroup")
plt.xlabel("GenderGroup")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["GlassesGroup"])
plt.title("Boxplot of GlassesGroup")
plt.xlabel("GlassesGroup")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["Height"])
plt.title("Boxplot of Height")
plt.xlabel("Height")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["Wingspan"])
plt.title("Boxplot of Wingspan")
plt.xlabel("Wingspan")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["CWDistance"])
plt.title("Boxplot of CWDistance")
plt.xlabel("CWDistance")
plt.ylabel("Amount")
plt.show()
```

```
sns.boxplot(x=dt["Complete"])
plt.title("Boxplot of Complete")
plt.xlabel("Complete")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["CompleteGroup "])
plt.title("Boxplot of CompleteGroup ")
plt.xlabel("CompleteGroup ")
plt.ylabel("Amount")
plt.show()

sns.boxplot(x=dt["Score"])
plt.title("Boxplot of Score")
plt.xlabel("Score")
plt.ylabel("Amount")
plt.show()
```



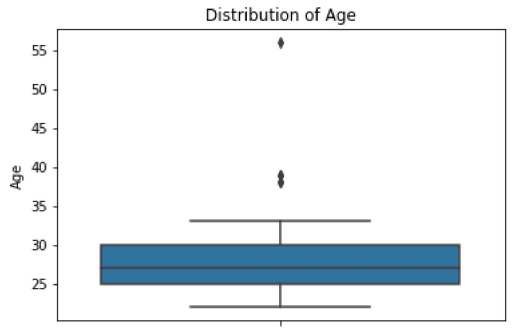


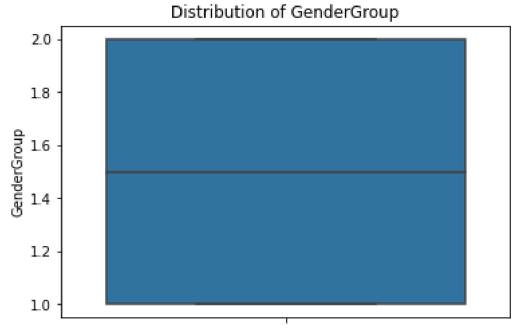


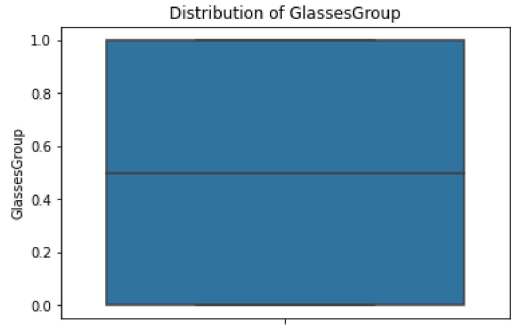


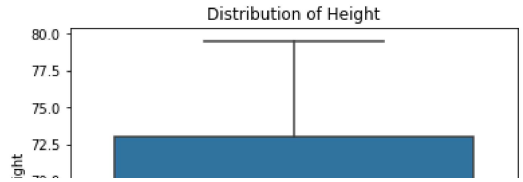

```
# Create the boxplot of the "tips" amounts

sns.boxplot(data=dt, y='Age')
plt.ylabel('Age')
plt.title('Distribution of Age')
plt.show()

sns.boxplot(data=dt, y='GenderGroup')
plt.ylabel('GenderGroup')
plt.title('Distribution of GenderGroup')
plt.show()

sns.boxplot(data=dt, y='GlassesGroup')
plt.ylabel('GlassesGroup')
plt.title('Distribution of GlassesGroup')
plt.show()

sns.boxplot(data=dt, y='Height')
plt.ylabel('Height')
plt.title('Distribution of Height')
plt.show()

sns.boxplot(data=dt, y='Wingspan')
plt.ylabel('Wingspan')
plt.title('Distribution of Wingspan')
plt.show()
```

```
sns.boxplot(data=dt, y='CWDistance')
plt.ylabel('CWDistance')
plt.title('Distribution of CWDistance')
plt.show()

sns.boxplot(data=dt, y='Complete')
plt.ylabel('Complete')
plt.title('Distribution of Complete')
plt.show()

sns.boxplot(data=dt, y='Age')
plt.ylabel('Age')
plt.title('Distribution of Age')
plt.show()

sns.boxplot(data=dt, y='CompleteGroup ')
plt.ylabel('CompleteGroup ')
plt.title('Distribution of CompleteGroup ')
plt.show()

sns.boxplot(data=dt, y='Score')
plt.ylabel('Score')
plt.title('Distribution of Score')
plt.show()
```

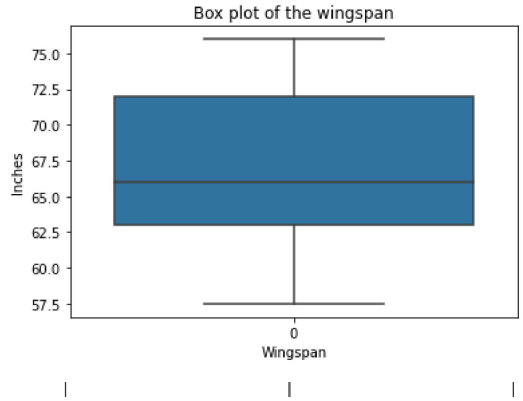




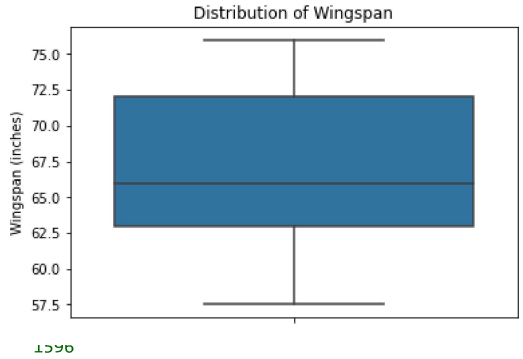




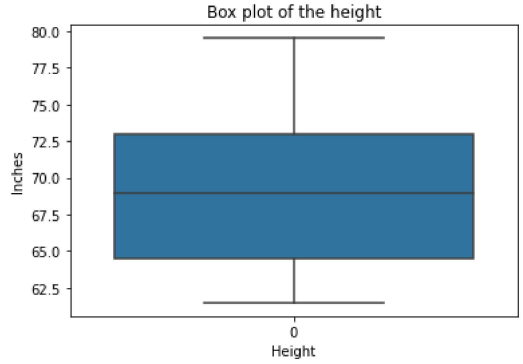

```
# Create the boxplots of the "Wingspan" and of the "Height" amounts
sns.boxplot(dt["Wingspan"]).set_title("Box plot of the wingspan")
plt.xlabel("Wingspan")
plt.ylabel("Inches")
plt.show()
```



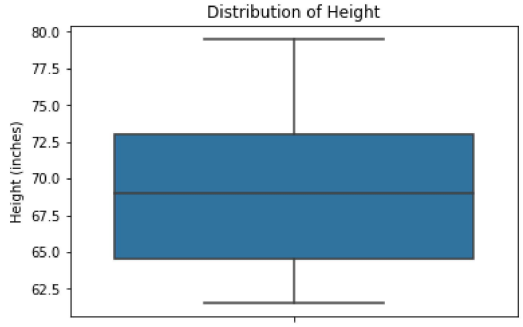

```
# Create the boxplots of the "Wingspan" and of the "tips" amounts
sns.boxplot(data=dt, y='Wingspan')
plt.ylabel('Wingspan (inches)')
plt.title('Distribution of Wingspan')
plt.show()
```



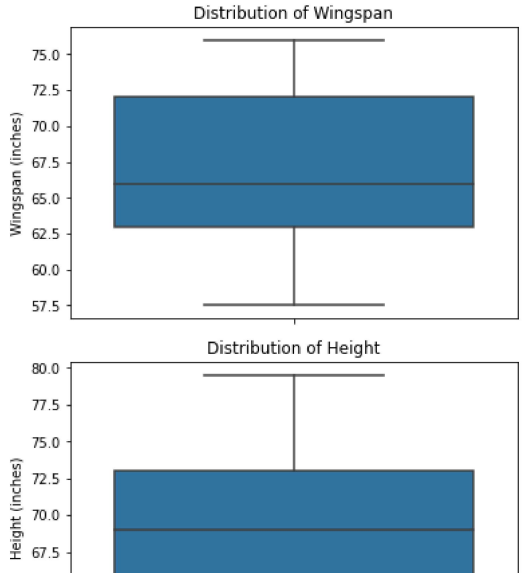

```
# Create the boxplot of the "Height" amounts

sns.boxplot(dt["Height"]).set_title("Box plot of the height")
plt.xlabel("Height")
plt.ylabel("Inches")
plt.show()
```



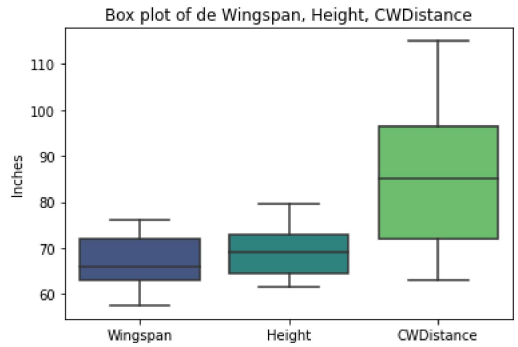

```
sns.boxplot(data=dt, y='Height')
plt.ylabel('Height (inches)')
plt.title('Distribution of Height')
plt.show()
```




```
# Create the boxplots of the "Wingspan" and of the "Height" amounts

sns.boxplot(data=dt, y='Wingspan')
plt.ylabel('Wingspan (inches)')
plt.title('Distribution of Wingspan')
plt.show()

sns.boxplot(data=dt, y='Height')
plt.ylabel('Height (inches)')
plt.title('Distribution of Height')
plt.show()
```

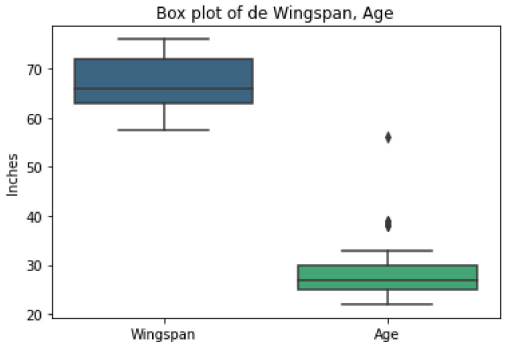


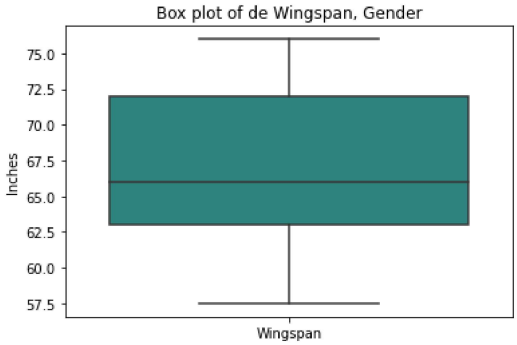

```
# Create the boxplots of the "Wingspan" and of the "tips" amounts

x = dt.loc[:, ["Wingspan", "Height", "CWDistance"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, Height, CWDistance")
plt.ylabel("Inches")
plt.show()
```

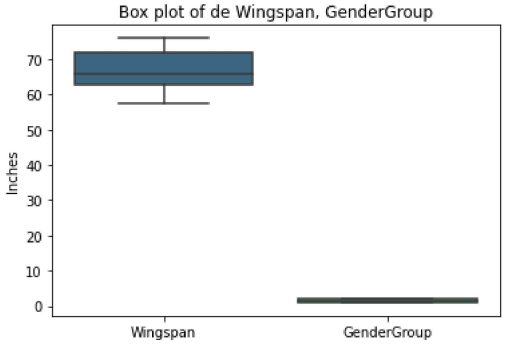


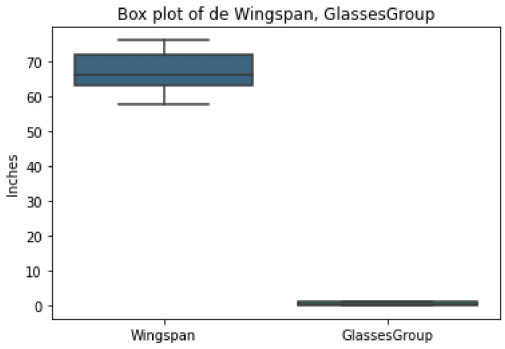

```
# Create the boxplots of the "Wingspan" and of the "tips" amounts

x = dt.loc[:, ["Wingspan", "Age"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, Age")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "Gender"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, Gender")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "GenderGroup"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, GenderGroup")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "GlassesGroup"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, GlassesGroup")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "Height"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, Height")
plt.ylabel("Inches")
```

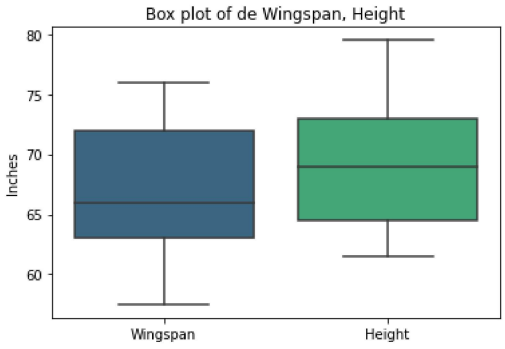
```
plt.show()

x = dt.loc[:, ["Wingspan", "CWDistance"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, CWDistance")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "Complete"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, Complete")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "CompleteGroup"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, CompleteGroup ")
plt.ylabel("Inches")
plt.show()

x = dt.loc[:, ["Wingspan", "Score"]]
x2bp = sns.boxplot(data=x, orient="v", palette="viridis")
x2bp.set_title("Box plot of de Wingspan, Score")
plt.ylabel("Inches")
plt.show()
```


Box plot of de Wingspan, Age
Inches
70
60
50
40
30
20
Wingspan
Age


Box plot of de Wingspan, Gender
Inches
75.0
72.5
70.0
67.5
65.0
62.5
60.0
57.5
Wingspan


Box plot of de Wingspan, GenderGroup
Inches
70
60
50
40
30
20
10
0
Wingspan
GenderGroup


Box plot of de Wingspan, GlassesGroup
Inches
70
60
50
40
30
20
10
0
Wingspan
GlassesGroup


Box plot of de Wingspan, Height
Inches
80
75
70
65
60
Wingspan
Height

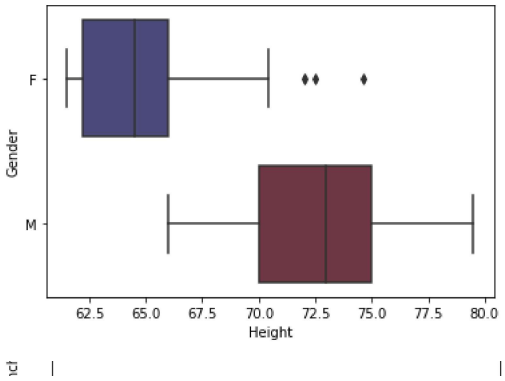

Box plot of de Wingspan, CWDistance
110
100

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.

Box plot of de Wingspan, Complete

```
# Create side-by-side boxplots of the "Height" grouped by "Gender"

sns.boxplot(x = dt["Height"], y = dt["Gender"], palette = "icefire")
plt.show()
```

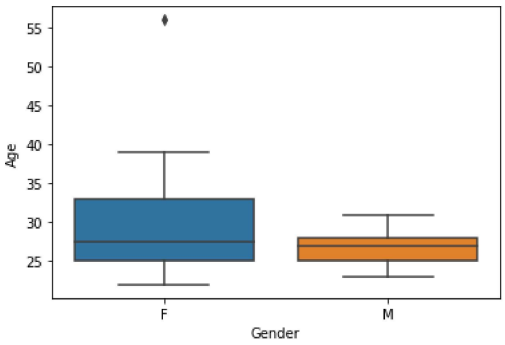


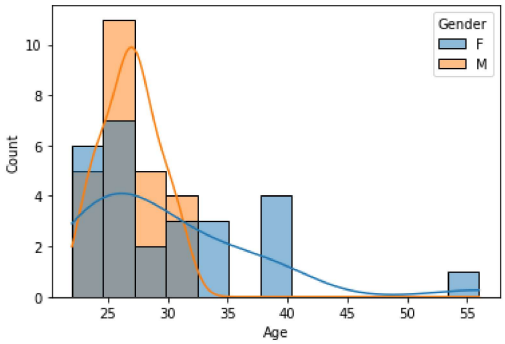
Histograms and boxplots plotted by groups

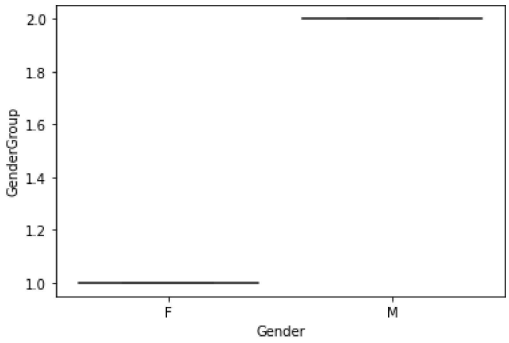
We cal also create both boxplots and histograms of one quantitative variable grouped by another categorical variables

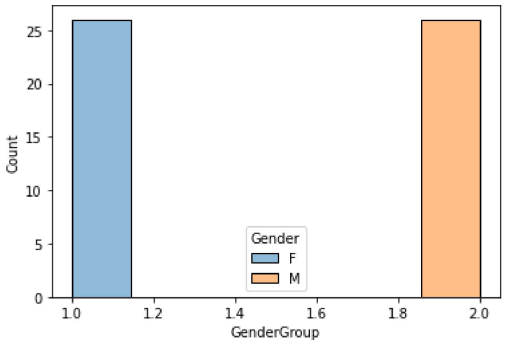
Wingspan CompleteGroup

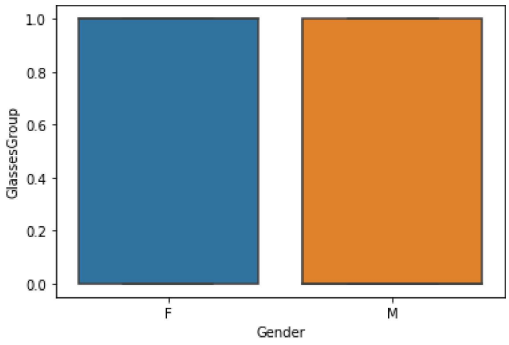
```
# Create a boxplot and histogram of the "tips" grouped by "Gender"

sns.boxplot(x="Gender", y="Age", data=dt)
plt.show()
sns.histplot(x="Age", hue="Gender", data=dt, kde=True)
plt.show()

sns.boxplot(x="Gender", y="GenderGroup", data=dt)
plt.show()
sns.histplot(x="GenderGroup", hue="Gender", data=dt, kde=True)
plt.show()

sns.boxplot(x="Gender", y="GlassesGroup", data=dt)
plt.show()
sns.histplot(x="GlassesGroup", hue="Gender", data=dt, kde=True)
plt.show()

sns.boxplot(x="Gender", y="Height", data=dt)
plt.show()
sns.histplot(x="Height", hue="Gender", data=dt, kde=True)
plt.show()

sns.boxplot(x="Gender", y="Wingspan", data=dt)
plt.show()
sns.histplot(x="Wingspan", hue="Gender", data=dt, kde=True)
plt.show()

sns.boxplot(x="Gender", y="CWDistance", data=dt)
plt.show()
sns.histplot(x="CWDistance", hue="Gender", data=dt, kde=True)
plt.show()

sns.boxplot(x="Gender", y="Score", data=dt)
plt.show()
sns.histplot(x="Score", hue="Gender", data=dt, kde=True)
plt.show()
```

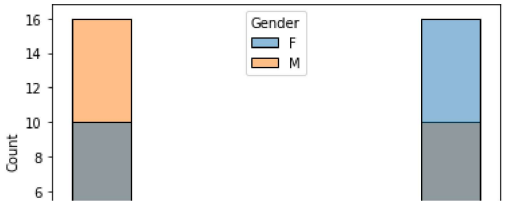

Age
Gender
F
M


Count
Age
Gender
F
M


GenderGroup
Gender
F
M


Count
GenderGroup
Gender
F
M

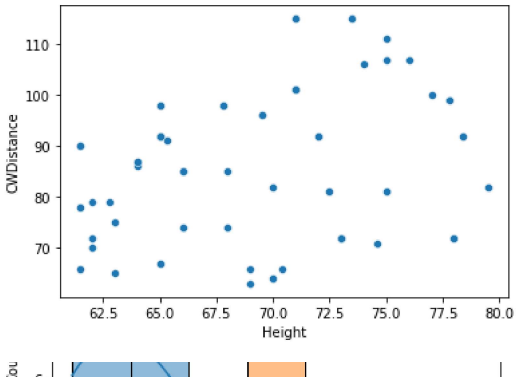

GlassesGroup
Gender
F
M


Count
Gender
F
M

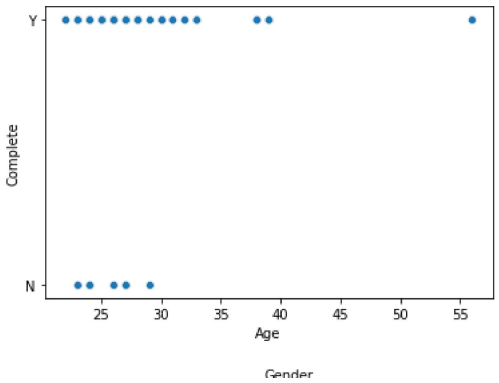
Scatter plot

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

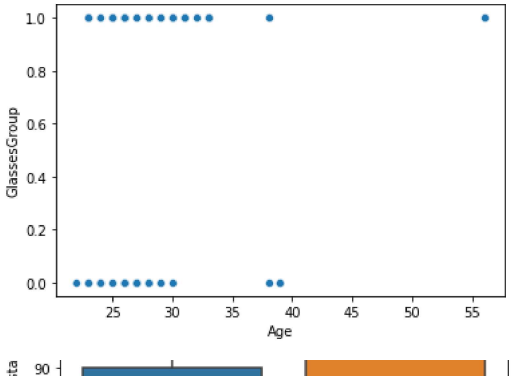
```
# scatter plot between two variables
sns.scatterplot(data = dt, y = "CWDistance", x = "Height")
plt.show()
```



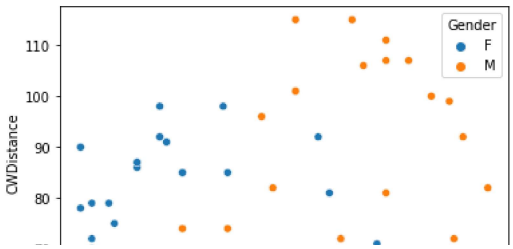

```
# scatter plot between two variables (one categorical)
sns.scatterplot(data = dt, y = "Complete", x = "Age")
plt.show()
```



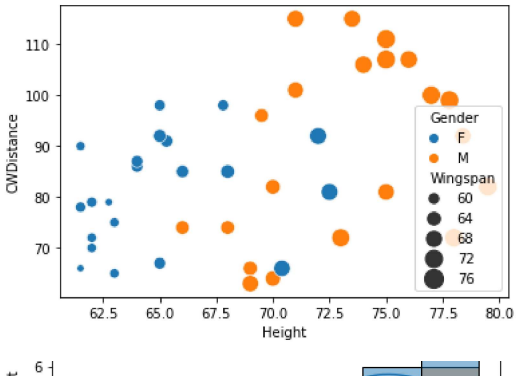

```
# scatter plot between two variables (one categorical)
sns.scatterplot(data = dt, y = "GlassesGroup", x = "Age")
plt.show()
```




```
# scatter plot between two variables grouped according to a categorical variable
sns.scatterplot(x='Height', y='CWDistance', hue='Gender', data=dt)
plt.show()
```




```
# scatter plot between two variables grouped according to a categorical variable and with size of markers
sns.scatterplot(x='Height', y='CWDistance', hue='Gender', size='Wingspan', sizes=(30, 200), data=dt)
plt.show()
```



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

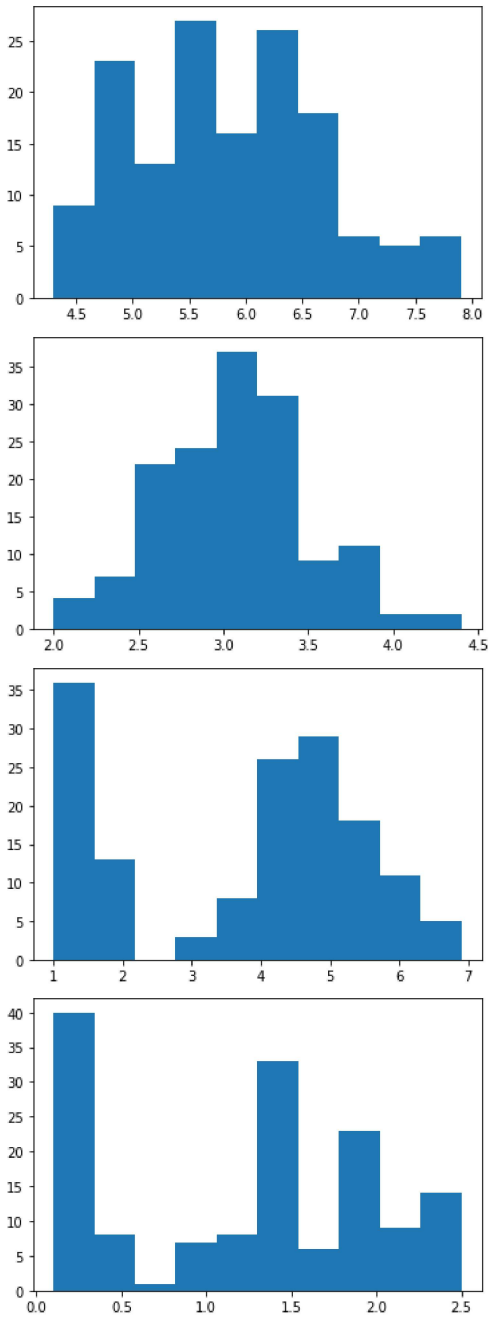
Activity: work with the iris dataset

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

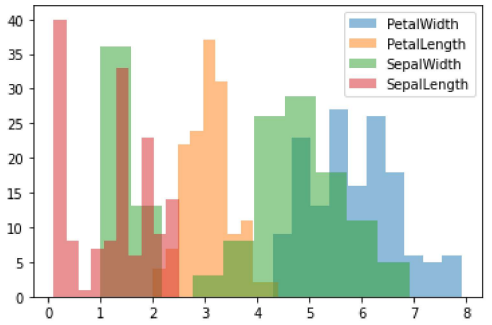
1. Plot the histograms for each of the four quantitative variables
2. Plot the histograms for each of the quantitative variables
3. Plot the boxplots for each of the quantitative variables
4. Plot the boxplots of the petal width grouped by type of flower
5. Plot the boxplots of the setal length grouped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

```
irs_url = Ruta + "A01641179/datasets/iris/iris.csv"
dataset = pd.read_csv(irs_url )
dataset.columns = ["PetalWidth", "PetalLength", "SepalWidth", "SepalLength", "Type"]


#Plot the histograms for each of the four quantitative variables

plt.hist(dataset["PetalWidth"])
plt.show()

plt.hist(dataset["PetalLength"])
plt.show()

plt.hist(dataset["SepalWidth"])
plt.show()
```

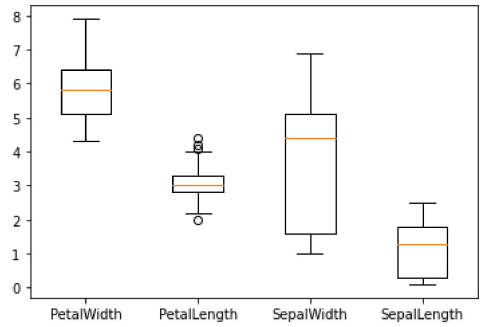
```
plt.hist(dataset["SepalLength"])
plt.show()
```



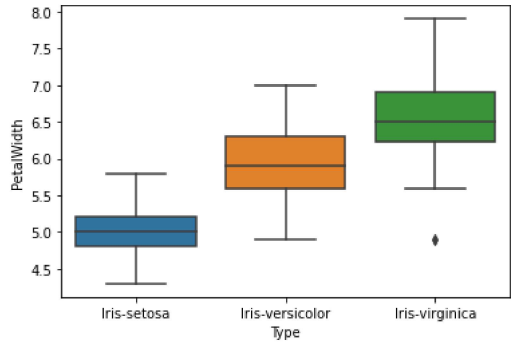
```
plt.hist(dataset["PetalWidth"], alpha=0.5, label="PetalWidth")
plt.hist(dataset["PetalLength"], alpha=0.5, label="PetalLength")
plt.hist(dataset["SepalWidth"], alpha=0.5, label="SepalWidth")
plt.hist(dataset["SepalLength"], alpha=0.5, label="SepalLength")
plt.legend()
plt.show()
```



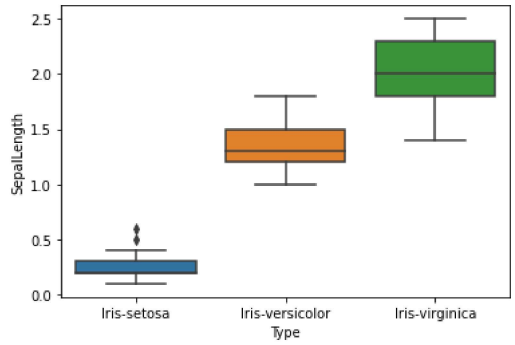
```
plt.boxplot([dataset["PetalWidth"], dataset["PetalLength"], dataset["SepalWidth"], dataset["SepalLength"]])
plt.xticks([1, 2, 3, 4], ["PetalWidth", "PetalLength", "SepalWidth", "SepalLength"])
plt.show()
```



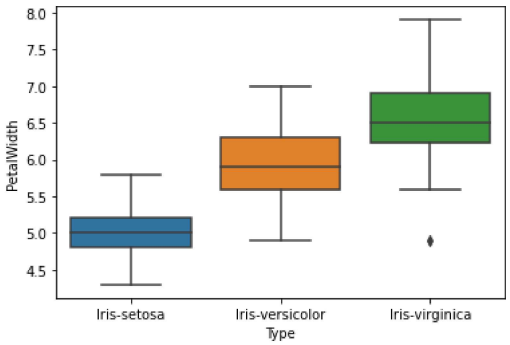

```
sns.boxplot(data=dataset, x="Type", y="PetalWidth")
plt.show()
```

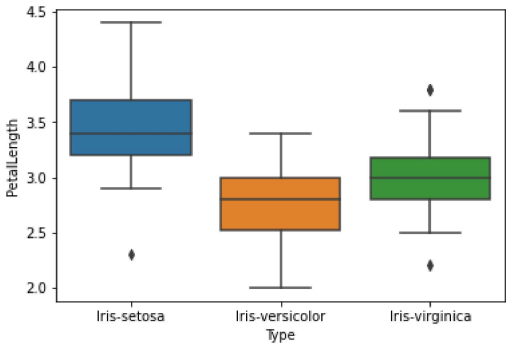


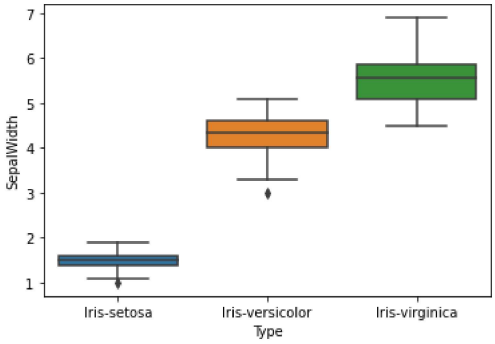

```
sns.boxplot(data=dataset, x="Type", y="SepalLength")
plt.show()
```

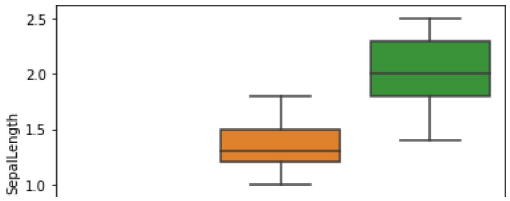



```
#All the boxplots grouped by type of flower

sns.boxplot(data=dataset, x="Type", y="PetalWidth")
plt.show()

sns.boxplot(data=dataset, x="Type", y="PetalLength")
plt.show()

sns.boxplot(data=dataset, x="Type", y="SepalWidth")
plt.show()

sns.boxplot(data=dataset, x="Type", y="SepalLength")
plt.show()
```










```
#Provide a description (explaination from your observations) of each of the quantitative variables

"""
En el analisis de las graficas podemos determinar que entre mas crece
se tiende a tener une iris virginica y en el caso contrario se tiende a tener
una setosa.

"""

"""
En el analisis de las graficas podemos determinar que entre mas crece
se tiende a tener une iris virginica y en el caso contrario se tiende a tener
una iris setosa.

"""
```

```
'\nEn el analisis de las graficas podemos determinar que entre mas crece\nse tiende a tener une iris virginica y en el caso contrario s
e tiende a tener\nuna iris setosa.\n\n'
```

✓ 0 s se ejecutó 23:02