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
from sklearn.datasets import load_iris
```

Importing data

```
# url string that hosts our .csv file
absoluteRoute = "/content/drive/My Drive/semana:tec/datasets_iris/Iris.csv/"

iris = load_iris()
# Read the .csv file and store it as a pandas Data Frame
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Add the species as a column for easier grouping
df['Class'] = iris.target_names[iris.target]
```

Exploring the content of the data set

Get a general 'feel' of the data

```
#Printing the head and some statistics
print(df.head(5))

print(df.describe())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

    Class
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
```

```
50%            5.800000          3.000000           4.350000
75%            6.400000          3.300000           5.100000
max            7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```

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.value_counts(df["Class"])
```

```
Class
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

```
# Proporción de cada clase usando groupby y size
class_counts = df.groupby('Class').size()

# Calcular las proporciones
total_count = len(df)
proportions = class_counts / total_count

# Mostrar las proporciones para cada clase
print("Proportion of Iris-versicolor: ", proportions['versicolor'])
print("Proportion of Iris-virginica: ", proportions['virginica'])
print("Proportion of Iris-setosa: ", proportions['setosa'])
```

```
Proportion of Iris-versicolor:  0.3333333333333333
Proportion of Iris-virginica:  0.3333333333333333
Proportion of Iris-setosa:  0.3333333333333333
```

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 (excluding missing values)
not_null_ob = df.count()
print("Total number of non-null observations:\n", not_null_ob)

# Total number of null observations
null_ob = df.isna().sum()
print("Total number of null observations:\n", null_ob)

# Total number of counts (excluding missing values)
print("Total number of counts (excluding missing values):\n", not_null_ob)
```

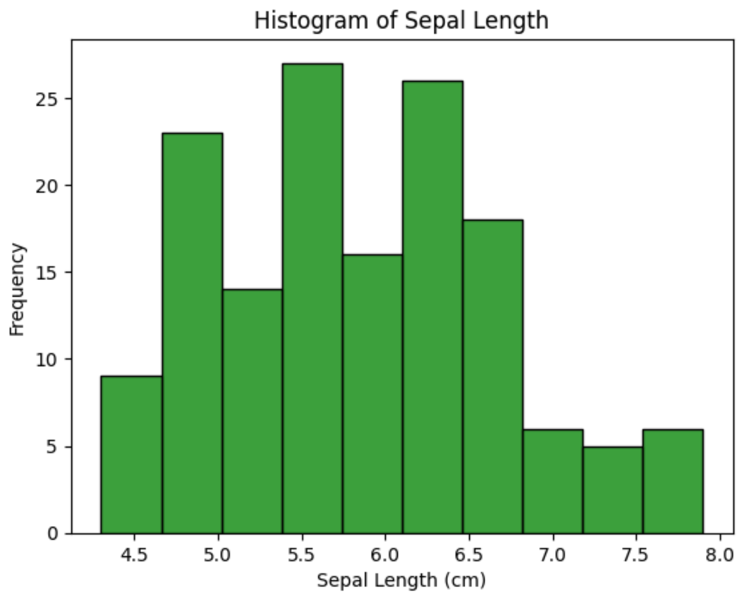
```
Total number of non-null observations:
 sepal length (cm)    150
sepal width (cm)     150
petal length (cm)    150
petal width (cm)     150
```

```
Class                150
dtype: int64
Total number of null observations:
 sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
Class                0
dtype: int64
Total number of counts (excluding missing values):
 sepal length (cm)    150
sepal width (cm)     150
petal length (cm)    150
petal width (cm)     150
Class                150
dtype: int64
```

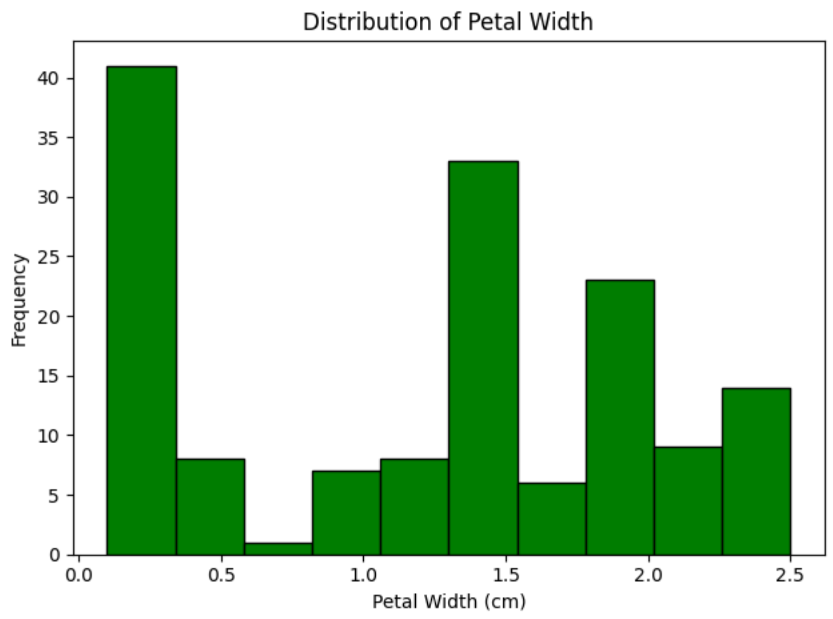
Histogram

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

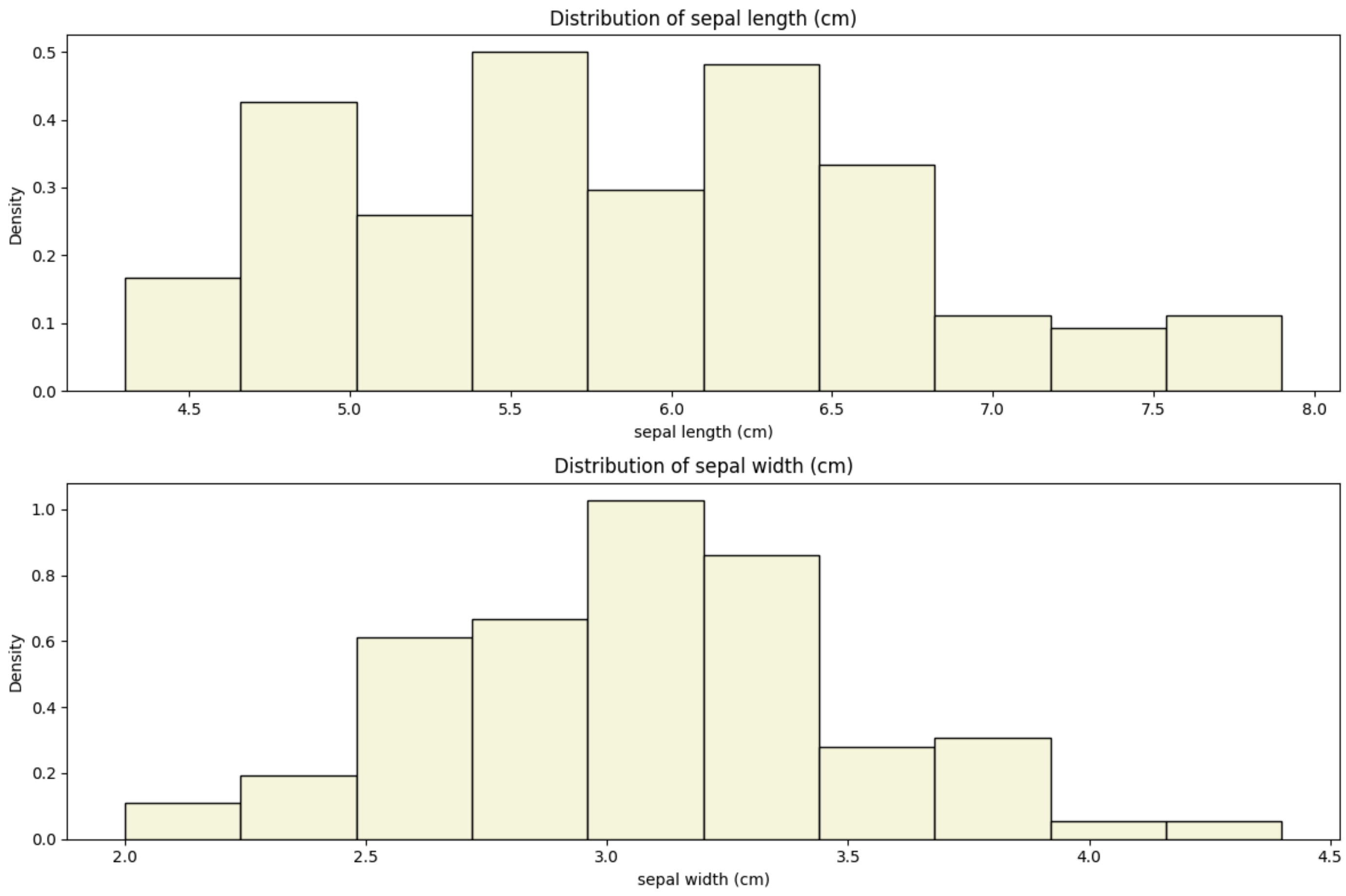
```
import seaborn as sns

# Plot histogram using seaborn
sns.histplot(df['sepal length (cm)'], bins=10, color='green', edgecolor='black')

# Adding titles and labels
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Frequency')

plt.show()
```



```
import matplotlib.pyplot as plt

# Plot distribution using matplotlib directly
fig, ax = plt.subplots()
ax.hist(df["petal width (cm)"], bins=10, color='green', edgecolor='black')

# Adding titles and labels
ax.set_title('Distribution of Petal Width')
ax.set_xlabel('Petal Width (cm)')
ax.set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



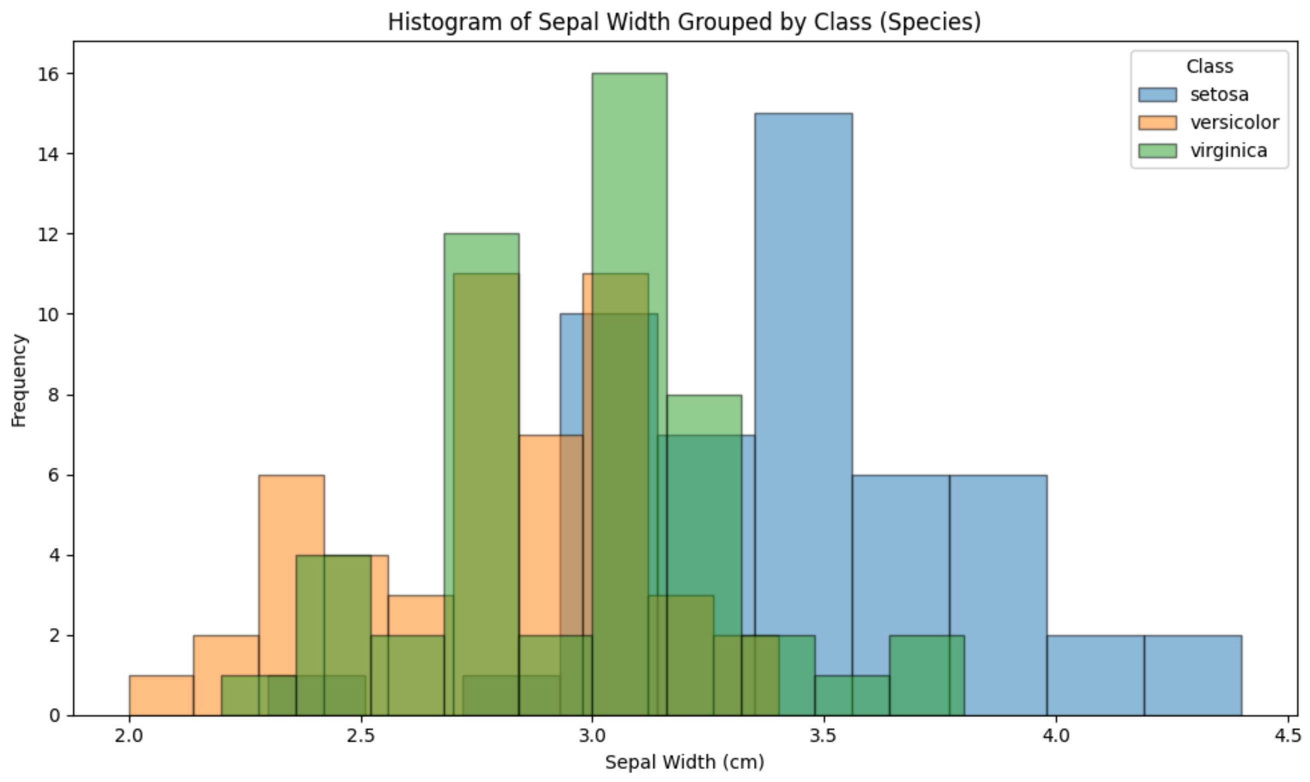
```
import matplotlib.pyplot as plt

# Crear subgráficas
fig, axes = plt.subplots(2, figsize=(12, 8))

# Características a graficar
features = ['sepal length (cm)', 'sepal width (cm)']

for feature, ax in zip(features, axes.flatten()):
    # Histograma con densidad
    ax.hist(df[feature], bins=10, color='beige', edgecolor='black', density=True)

    # Configurar el título y las etiquetas
    ax.set_title(f'Distribution of {feature}')
    ax.set_xlabel(feature)
    ax.set_ylabel('Density')

# Ajustar el diseño
plt.tight_layout()

# Mostrar los gráficos
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.

```
import matplotlib.pyplot as plt

# Crear subgráficas
fig, ax = plt.subplots(figsize=(10, 6))

# Agrupar datos por 'Class' y trazar histogramas apilados
classes = df['Class'].unique()
for cls in classes:
    subset = df[df['Class'] == cls]
    ax.hist(subset['sepal width (cm)'], bins=10, label=cls, alpha=0.5, edgecolor='black')

# Configurar título y etiquetas
ax.set_title('Histogram of Sepal Width Grouped by Class (Species)')
ax.set_xlabel('Sepal Width (cm)')
ax.set_ylabel('Frequency')
ax.legend(title='Class')

# Mostrar el gráfico
plt.tight_layout()
plt.show()
```

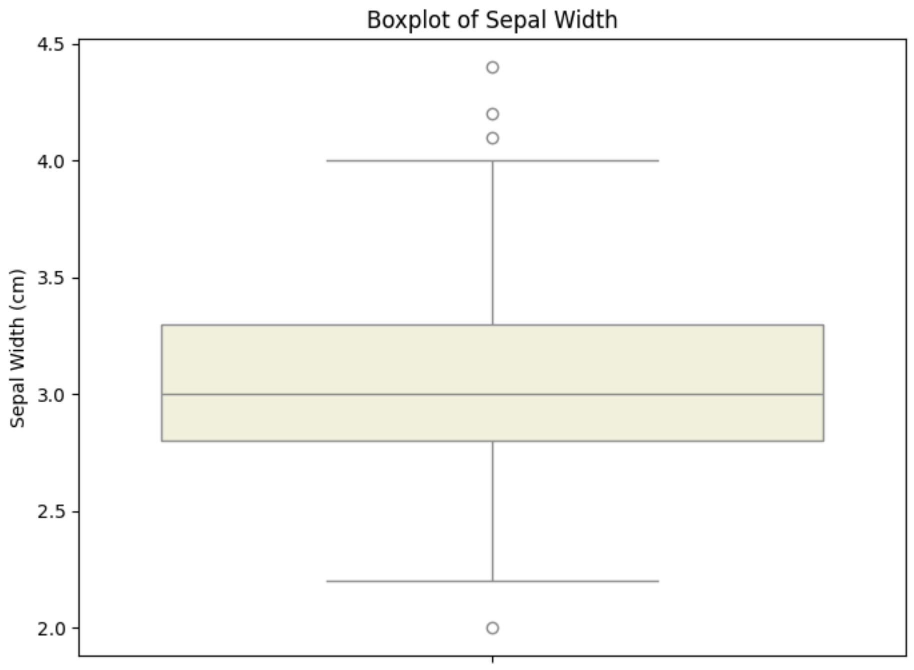


Boxplots

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

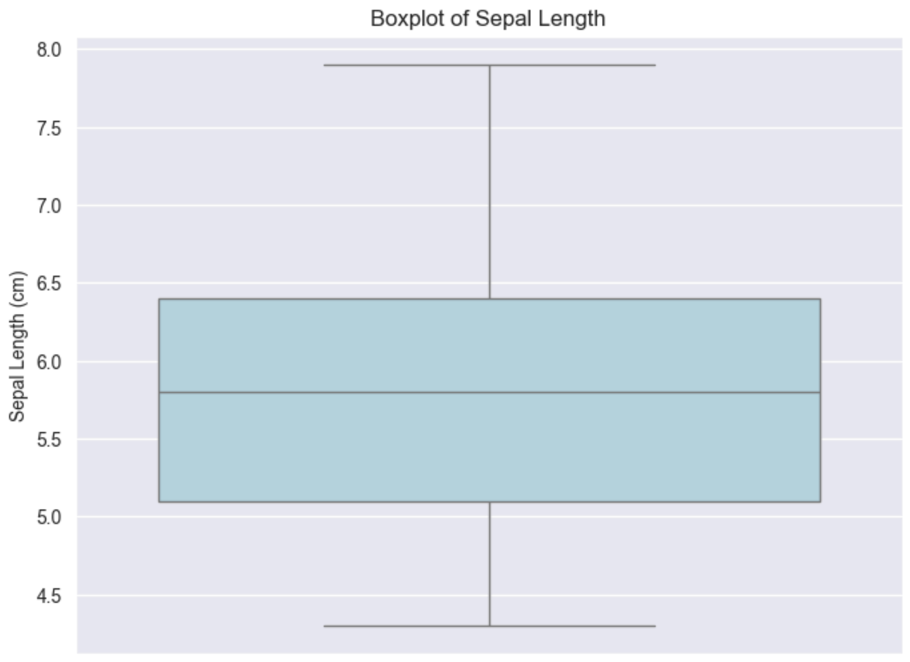
```
# Create the boxplot of the "sepal width" amounts
plt.figure(figsize=(8, 6))
sns.boxplot(y='sepal width (cm)', data=df, color='beige')

plt.title('Boxplot of Sepal Width')
plt.ylabel('Sepal Width (cm)')
```

```
Text(0, 0.5, 'Sepal Width (cm)')
```

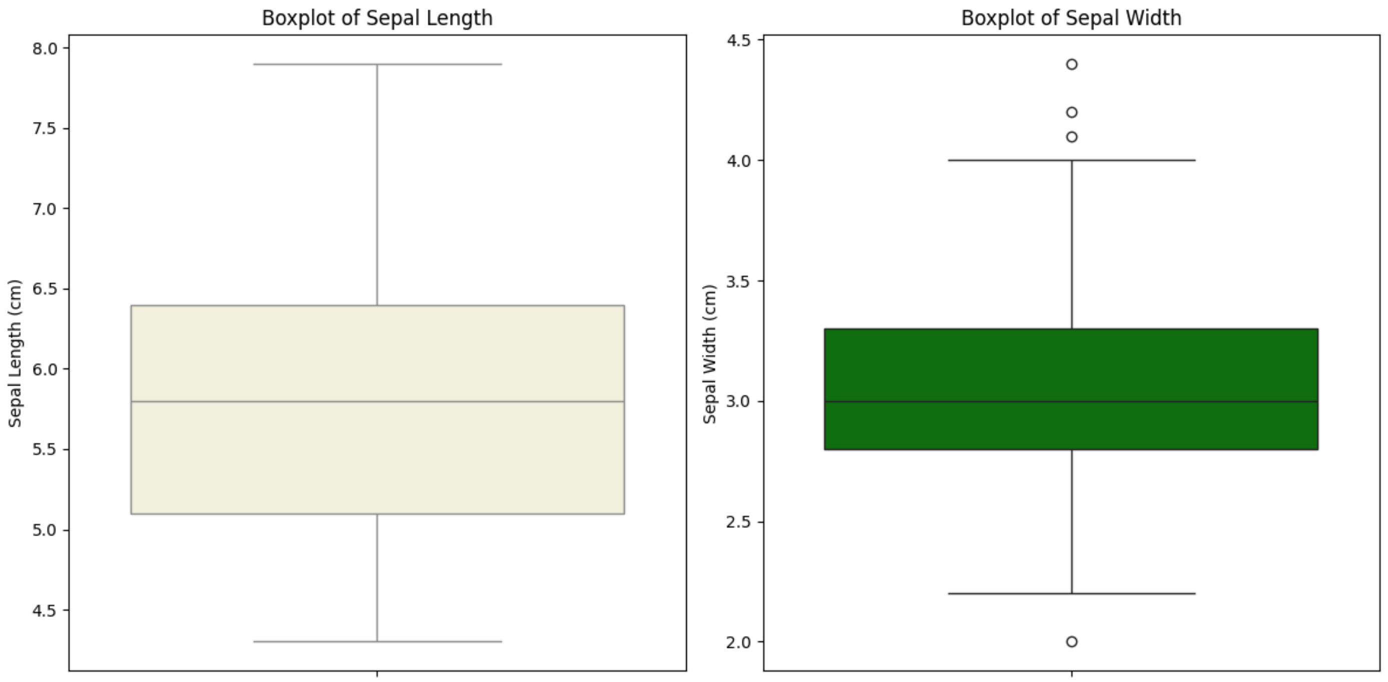


```
# Create the boxplots of the "sepal length" amounts
plt.figure(figsize=(8, 6))
sns.boxplot(y='sepal length (cm)', data=df, color='lightblue')

plt.title('Boxplot of Sepal Length')
plt.ylabel('Sepal Length (cm)')
```

```
Text(0, 0.5, 'Sepal Length (cm)')
```



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

# Crear la figura y los ejes
plt.figure(figsize=(12, 6))

# Gráfico de caja para 'sepal length (cm)'
plt.subplot(1, 2, 1)
sns.boxplot(y='sepal length (cm)', data=df, color='beige')
plt.title('Boxplot of Sepal Length')
plt.ylabel('Sepal Length (cm)')

# Gráfico de caja para 'sepal width (cm)'
plt.subplot(1, 2, 2)
sns.boxplot(y='sepal width (cm)', data=df, color='green')
plt.title('Boxplot of Sepal Width')
plt.ylabel('Sepal Width (cm)')

# Ajustar el diseño
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```

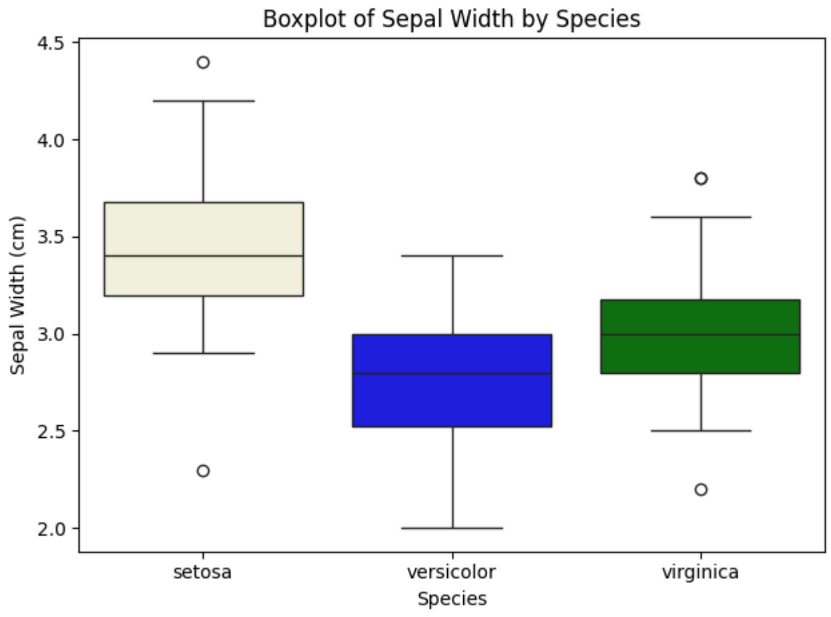


Boxplots plotted by groups

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.

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

# Definir una paleta de colores personalizada
custom_palette = ['beige', 'blue', 'green']

# Crear el gráfico de caja
sns.boxplot(x='Class', y='sepal width (cm)', data=df, palette=custom_palette)

# Configurar el título y etiquetas
plt.title('Boxplot of Sepal Width by Species')
plt.xlabel('Species')
plt.ylabel('Sepal Width (cm)')

# Ajustar el diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

```
<ipython-input-17-8d4cb3595db8>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.boxplot(x='Class', y='sepal width (cm)', data=df, palette=custom_palette)
```

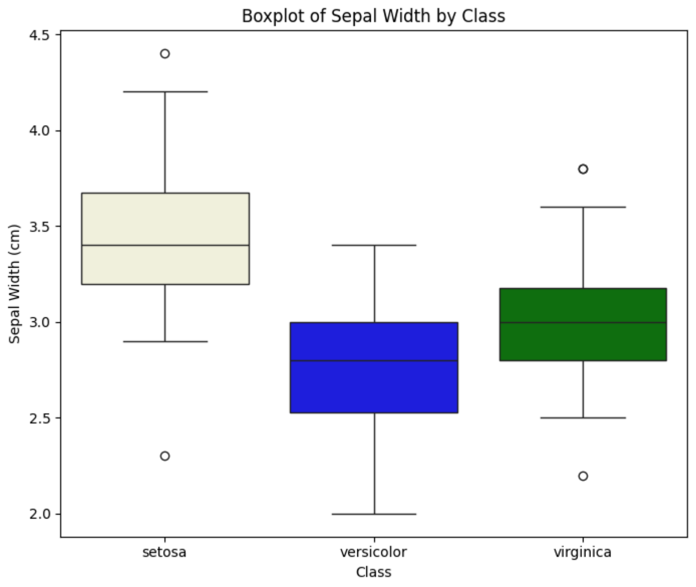


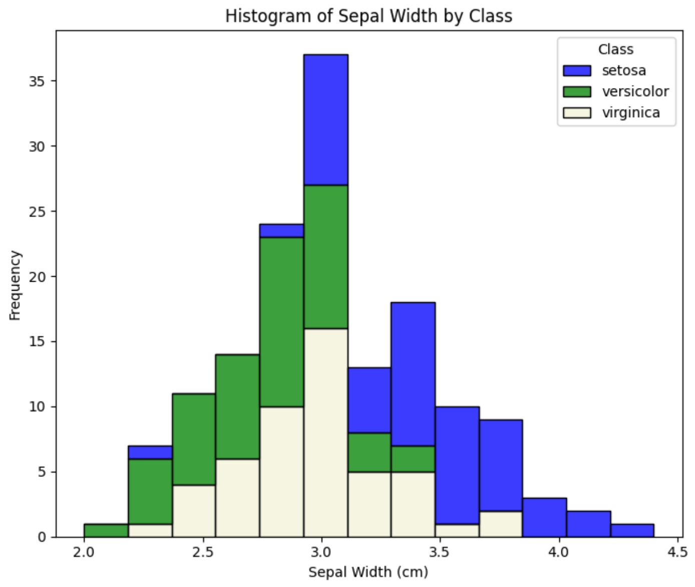
Histograms and boxplots plotted by groups

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

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

# Definir paletas de colores personalizadas
boxplot_palette = ['beige', 'blue', 'green']
histogram_palette = ['blue', 'green', 'beige']

# Crear la figura
plt.figure(figsize=(14, 6))

# Boxplot para 'sepal width (cm)' agrupado por 'Class'
plt.subplot(1, 2, 1)
sns.boxplot(x='Class', y='sepal width (cm)', data=df, palette=boxplot_palette)
plt.title('Boxplot of Sepal Width by Class')
plt.xlabel('Class')
plt.ylabel('Sepal Width (cm)')

# Histograma para 'sepal width (cm)' agrupado por 'Class'
plt.subplot(1, 2, 2)
sns.histplot(data=df, x='sepal width (cm)', hue='Class', multiple='stack', palette=histogram_palette)
plt.title('Histogram of Sepal Width by Class')
plt.xlabel('Sepal Width (cm)')
plt.ylabel('Frequency')

# Ajustar el diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

```
<ipython-input-18-eaf09bf77391>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.boxplot(x='Class', y='sepal width (cm)', data=df, palette=boxplot_palette)
```



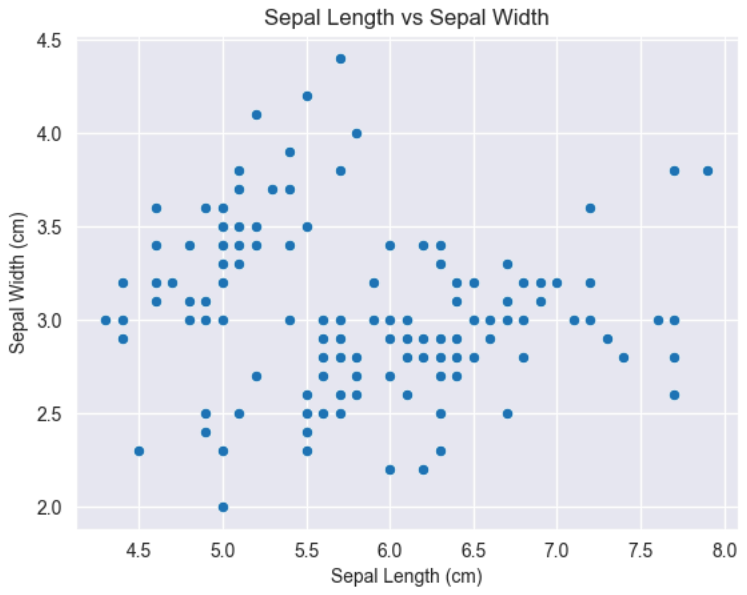


Scatter plot

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

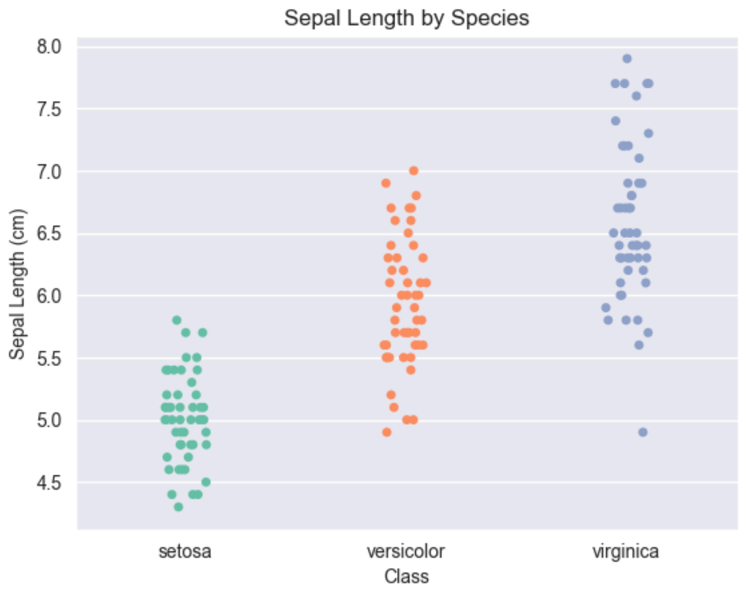
```
# scatter plot between two variables
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', data=df)
plt.title('Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
```

```
Text(0, 0.5, 'Sepal Width (cm)')
```



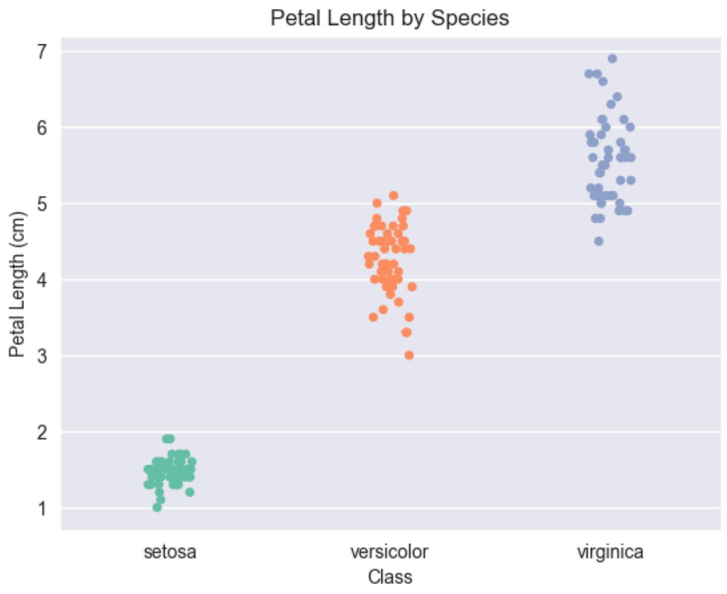
```
# scatter plot between two variables (one categorical)
sns.stripplot(x='Class', y='sepal length (cm)', data=df, hue="Class" ,jitter=True, palette='Set2')
plt.title('Sepal Length by Species')
plt.xlabel('Class')
plt.ylabel('Sepal Length (cm)')
```

```
Text(0, 0.5, 'Sepal Length (cm)')
```



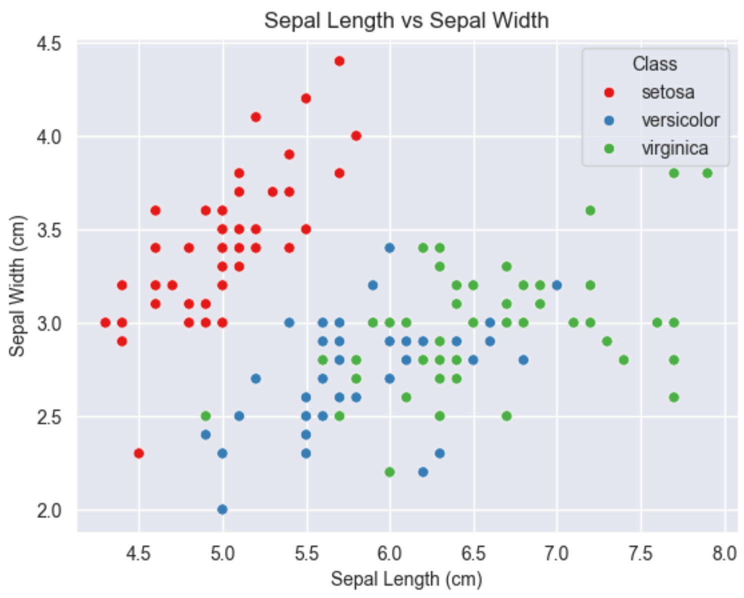
```
# scatter plot between two variables (one categorical)
sns.stripplot(x='Class', y='petal length (cm)', data=df, hue="Class" ,jitter=True, palette='Set2')
plt.title('Petal Length by Species')
plt.xlabel('Class')
plt.ylabel('Petal Length (cm)')
```

```
Text(0, 0.5, 'Petal Length (cm)')
```



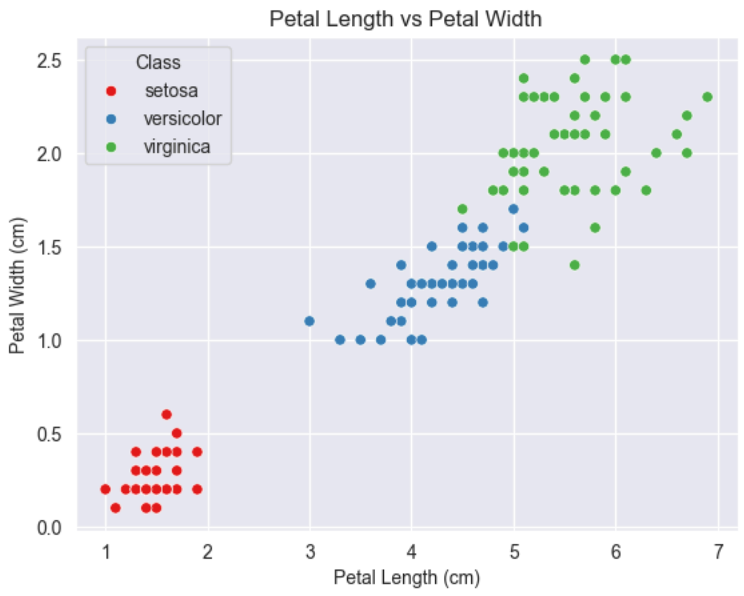
```
# scatter plot between two variables grouped according to a categorical variable
sns.scatterplot(x='sepal length (cm)', y='sepal width (cm)', hue='Class', data=df, palette='Set1')
plt.title('Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
```

```
Text(0, 0.5, 'Sepal Width (cm)')
```



```
# scatter plot between two variables grouped according to a categorical variable and with size of markers
sns.scatterplot(x='petal length (cm)', y='petal width (cm)', hue='Class', data=df, palette='Set1')
plt.title('Petal Length vs Petal Width')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
```

```
Text(0, 0.5, 'Petal Width (cm)')
```



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

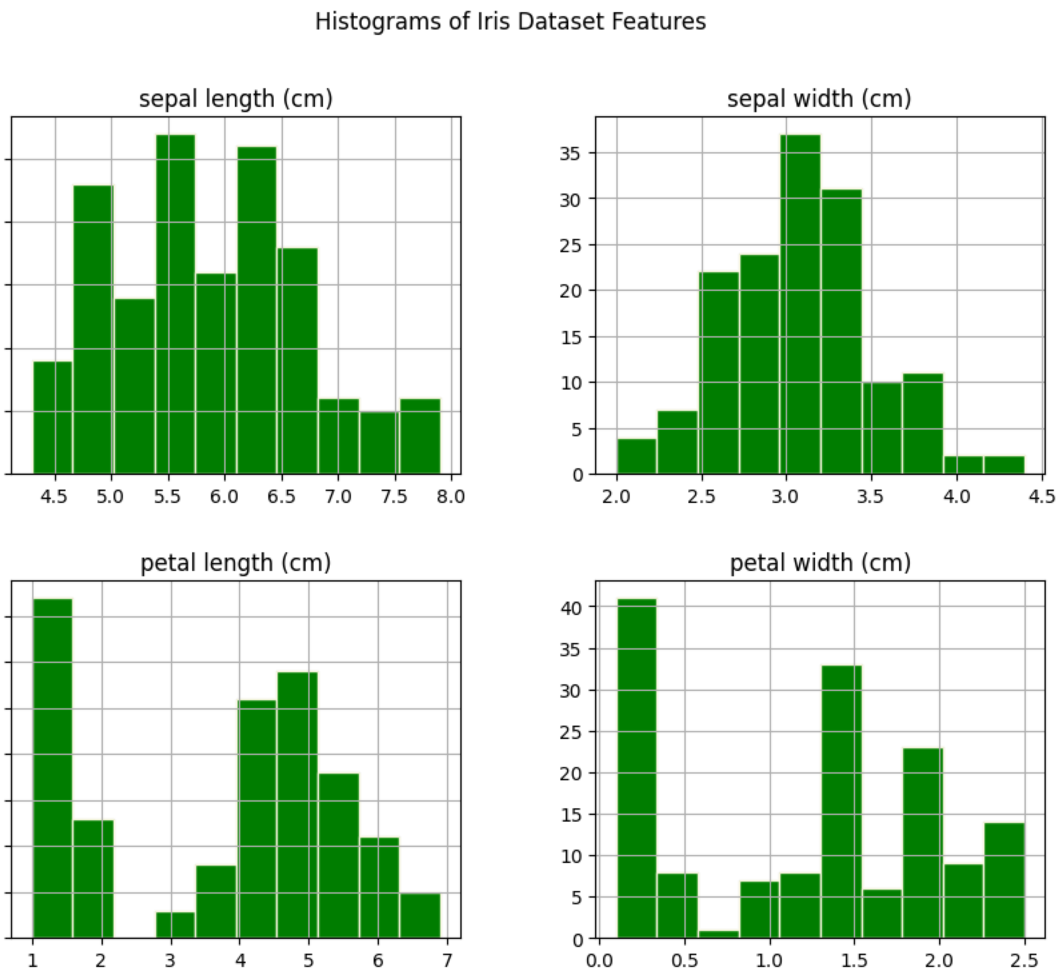
Activity: work with the iris dataset

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

1.

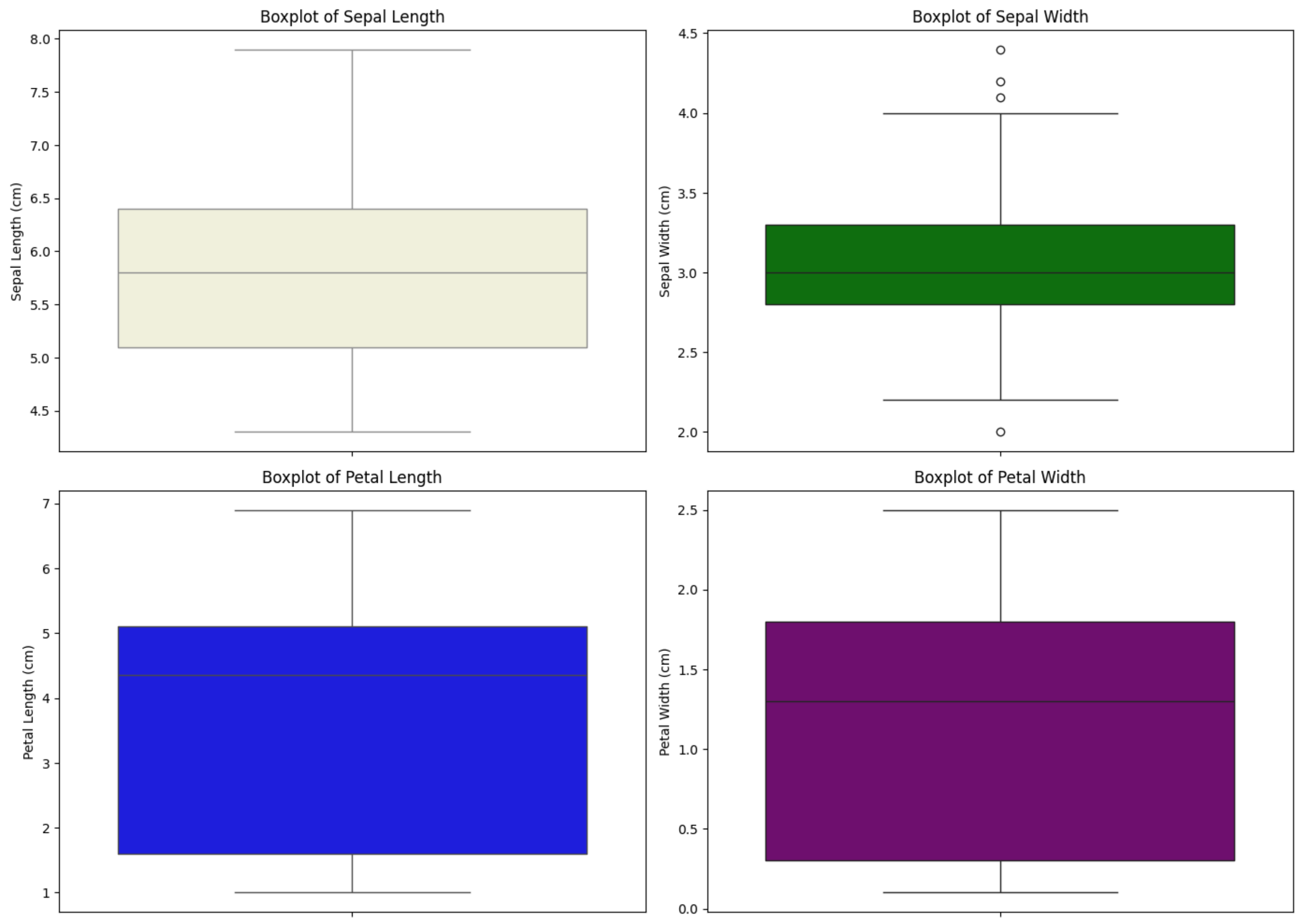
```
df.hist(bins=10, figsize=(10, 8), color='green', edgecolor='beige')

# Add title
plt.suptitle('Histograms of Iris Dataset Features')

# Show plot
plt.show()
```



2.

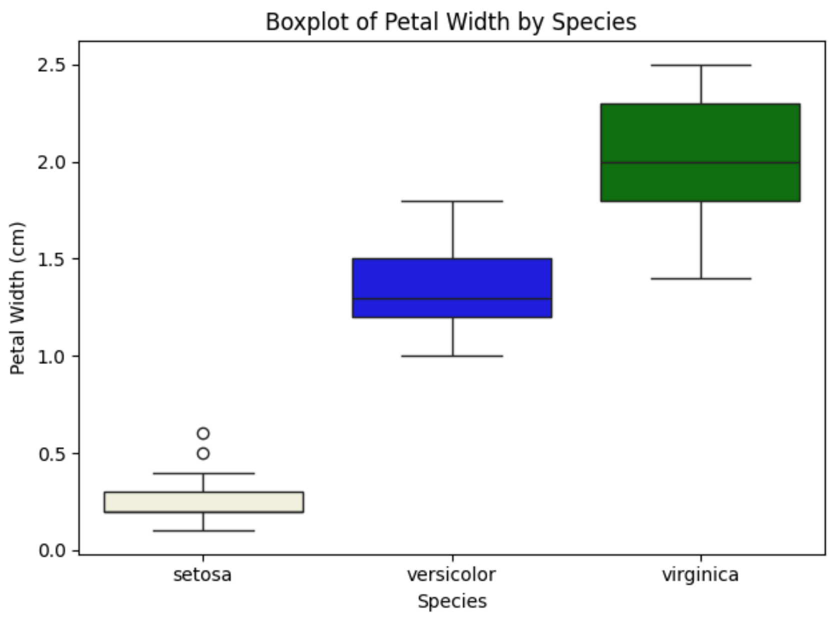
```
import seaborn as sns
import matplotlib.pyplot as plt

# Crear la figura
plt.figure(figsize=(14, 10))

# Gráfico de caja para la longitud del sépalo
plt.subplot(2, 2, 1)
sns.boxplot(y=df['sepal length (cm)'], color='beige')
plt.title('Boxplot of Sepal Length')
plt.ylabel('Sepal Length (cm)')
```

```
# Gráfico de caja para el ancho del sépalo
plt.subplot(2, 2, 2)
sns.boxplot(y=df['sepal width (cm)'], color='green')
plt.title('Boxplot of Sepal Width')
plt.ylabel('Sepal Width (cm)')

# Gráfico de caja para la longitud del pétalo
plt.subplot(2, 2, 3)
sns.boxplot(y=df['petal length (cm)'], color='blue')
plt.title('Boxplot of Petal Length')
plt.ylabel('Petal Length (cm)')

# Gráfico de caja para el ancho del pétalo
plt.subplot(2, 2, 4)
sns.boxplot(y=df['petal width (cm)'], color='purple')
plt.title('Boxplot of Petal Width')
plt.ylabel('Petal Width (cm)')

# Ajustar el diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```



3.

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

# Definir colores personalizados
custom_palette = ['beige', 'blue', 'green']  # Ajusta los colores según tus preferencias

# Crear el gráfico de caja
sns.boxplot(x='Class', y='petal width (cm)', data=df, hue='Class', palette=custom_palette)

# Configurar el título y etiquetas
plt.title('Boxplot of Petal Width by Species')
plt.xlabel('Species')
plt.ylabel('Petal Width (cm)')

# Ajustar el diseño
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```



4.