VISUALIZATION

Visualization is a technique used to illustrate information. To enhance data comprehension, data visualization entails displaying information in a graphical or pictorial format. This approach provides a clear understanding of the data and its trends. Python provides various libraries for data visualization, such as Matplotlib, Seaborn, Plotly, among others.

Let's delve deeper into Matplotlib and Seaborn.

MATPLOTLIB

Matplotlib, introduced in 2002 by John Hunter. It is for 2D plots which are built on numpy. A plot of matplotlib contains:

1 Figure

2 Axis

3 Axes

4 Artists

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is the foundation for many other plotting libraries in Python, making it a versatile tool for data visualization.

Unique Features:

Extensive control over plot appearance

Support for multiple backends

Integration with various GUI toolkits

Typical Use Cases:

Creating publication-quality plots

Customizing visual elements for detailed presentations

Figure

Figure is the container in which plots are present. It can have a single plot or multiple plots.

Axes

Axes are the plots, and they are artist attached to the figure. A plot can have 2 or 3 axes. set_xlabel(), set_ylabel() are the functions used to set the labels of x and y respectively.

Axis

It is responsible for generating ticks or the limits on the axes.

Artists

The whatever content visible in a figure are the artists.

PYPLOT

Pyplot is a sub library of matplotlib where all the utilities lie under. It has different types of plots including bar graphs, scatter plots, pie charts, histograms, area charts.

We import pyplot from matplotlib as following:

```
import matplotlib.pyplot as plt
```

We also import numpy as a support for the matplotlib:

```
import numpy as np
```

Some of the plots in matplotlib:

LINE CHART

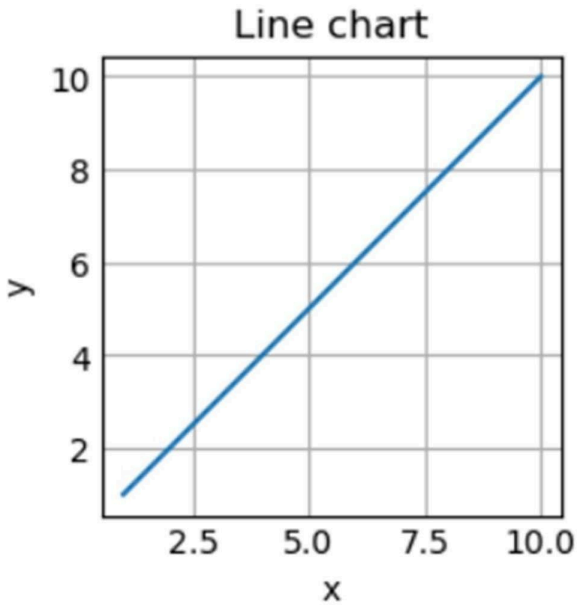
Line plots are the basic charts where dot consecutive points are connected by a continuous line.

The default plot() is used for line charts.

Code snippet:

```
import matplotlib.pyplot as plt
import numpy as np

x=np.array([1,10])
y=np.array([1,10])

plt.plot(x,y)
plt.title("Line chart")
plt.xlabel("x")
plt.ylabel("y")
plt.show()

```

Output:



Description:

np.array() will generate an array in the specified range.

plot() for the plotting the line chart.

title() for defining the title, ylabel() and xlabel () for labelling y and x axis respectively.

show() displays the graph.

BAR GRAPH

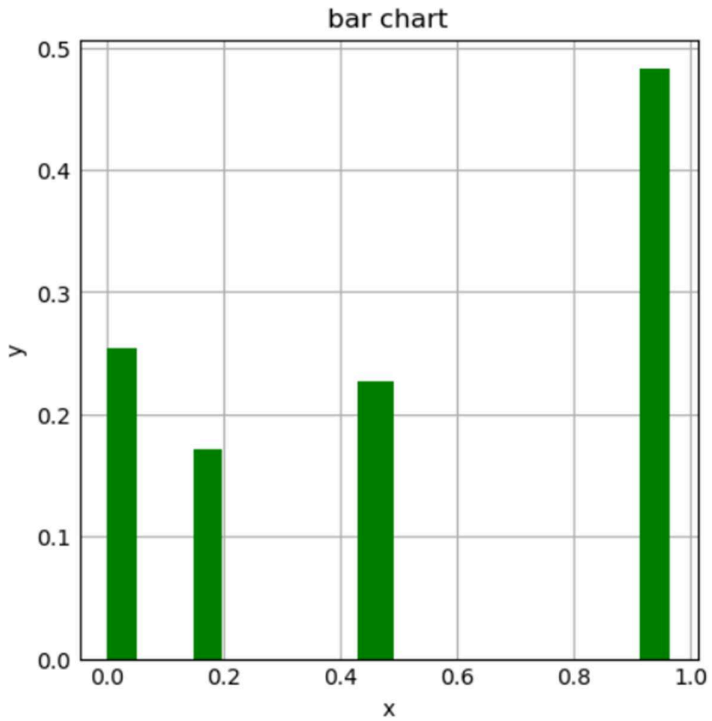
Rectangular bars are used in bar charts as the height represents the frequency of a particular element. bar() is used for plotting bar graphs, it has parameters like x, y, width, color.

bar(x, y, color, width)

Code snippet:

```
import matplotlib.pyplot as plt
import numpy as np

x=np.random.rand(5)
y=np.random.rand(5)
print(x)

fig = plt.figure(figsize = (4, 4))
plt.bar(x,y, width=0.05, color="green")
plt.title("bar chart")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Output:



Description:
random.rand(5) generates a random array.

Here bar() is used along with the parameters, width which denoted the width of the rectangular bars and color for the bars.

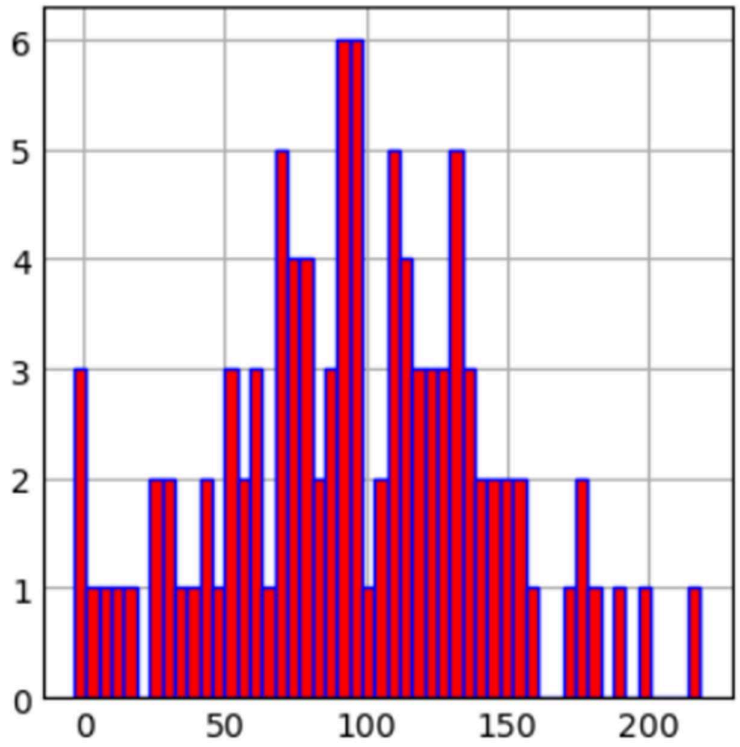
HISTOGRAM

Histogram is a type of bar graph where the graph is represented in groups. hist() is used for plotting histogram with parameters x, bins, color, edgecolor.

Code snippet:

```
import matplotlib.pyplot as plt
import numpy as np

x=np.random.normal(100,50,100)
fig = plt.figure(figsize = (3, 3))
plt.hist(x,bins=50,color="red", edgecolor="blue")
plt.show()
```

Output:



Description:

hist() have parameters x which is a random generated array around one hundred with standard deviation 50 of 100 values.

bins indicate the number of sections on x axis.

color indicates the color of rectangular bars.

edge color that divides the rectangular bars.

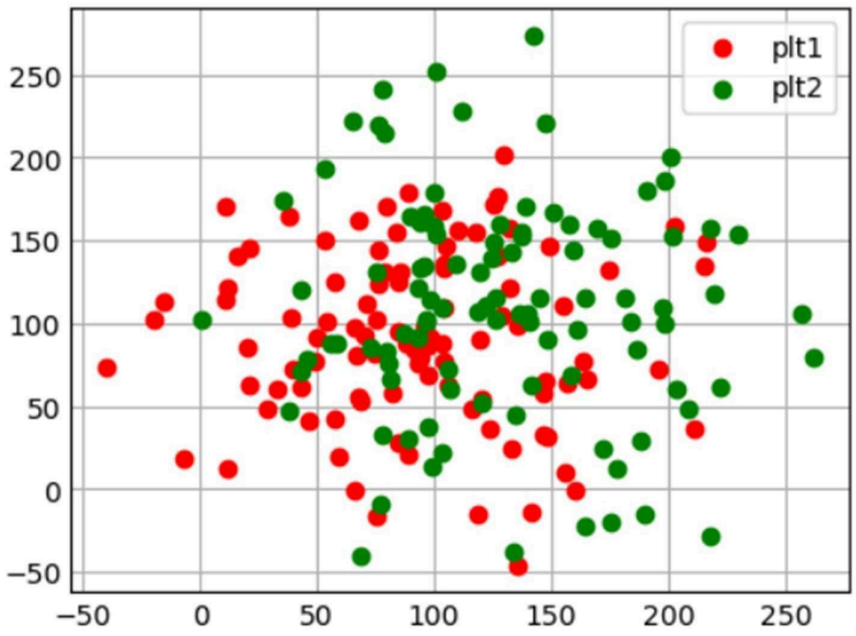
SCATTER PLOT

Scatter plot uses dots to depict the relationship between the data. scatter() is used for plotting scatter plot and scatter plot can be done for multiple datasets at the same time by differentiating it with colors. Legends help to achieve this difference.

Code snippet:

```
import matplotlib.pyplot as plt
import numpy as np

x1=np.random.normal(100,50,100)
y1=np.random.normal(100,50,100)
x2=np.random.normal(120,60,100)
y2=np.random.normal(120,60,100)
fig = plt.figure(figsize = (4, 3))
plt.scatter(x1,y1,c="r")
plt.scatter(x2,y2,c="g")
plt.legend(["plt1","plt2"])

plt.show()
```

Output:



Description:

x1, y1 belong to one dataset and x2,y2 belong to another dataset.

Here the scatter() is used for the scatter plot and the parameters x, y and c which represents color.

Legend adds the label which helps to differentiate the plots.

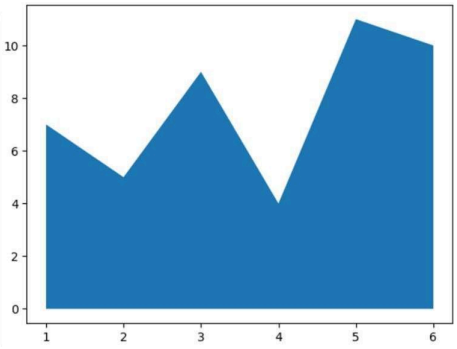
AREA CHART

In area chart, the area under the line to x axis is filled or shaded. It can be done by using fill_between() function or stackplot() function.

Code snippet:

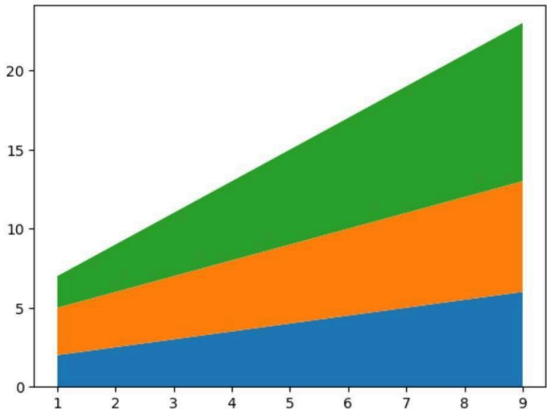
Using fill_between() function

```
import numpy as np
import matplotlib.pyplot as plt


x=[1,2,3,4,5,6]
y=[7,5,9,4,11,10]

plt.fill_between(x, y)
plt.show()
```



Using stackplot() function

```
import numpy as np
import matplotlib.pyplot as plt

x=[1,3,5,7,9]
y1=[2,3,4,5,6]
y2=[3,4,5,6,7]
y3=[2,4,6,8,10]

plt.stackplot(x,y1, y2, y3)
plt.show()
```



Description:

fill_between() function have parameters of x and y.

stackplot() function parameters are x, y1,y1,y3 where y1, y2, y3 are the different groups.

Legends are added by mentioning the labels attribute in stackplot() function and giving the position to the legend using plt.legend() function with attribute loc.

SEABORN

Seaborn is a visualization library in python which is built on top of matplotlib. It is more advanced than matplotlib and have different features which are default like style, color palette. Seaborn has different categories of plots like relational plot, categorical plot, distribution plot, regression plot, matrix plot.

Let us see the different plots in each category.

1 Relational Plots:

- scatterplot()
- lineplot()
- relplot()

2 Categorical Plots:

- barplot()
- countplot()
- boxplot()
- violinplot()
- swarmplot()
- pointplot()
- catplot()

3 Distribution Plots:

- histplot()
- kdeplot()
- rugplot()
- distplot() 4

Relational Plots:

- regplot()
- lmplot()

5 Matrix Plots:

- heatmap()
- clustermap()

Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Unique Features:

Built-in themes for styling

Simplified syntax for complex visualizations

Integrated with Pandas data structures

Typical Use Cases:

Exploring and understanding data

Visualizing statistical relationships

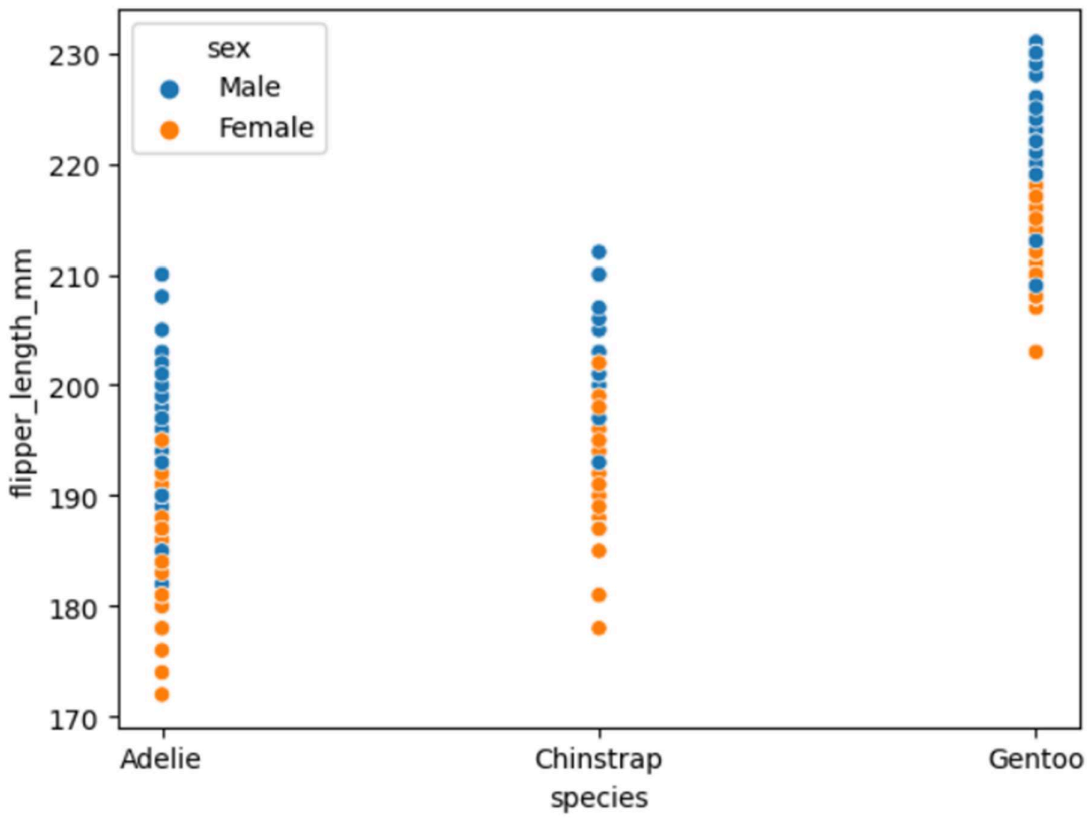
Here are some sample codes for some of the graphs.

Scatter plot:

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

df = sns.load_dataset("penguins")
```

```
sns.scatterplot(y='flipper_length_mm', x='species',hue='sex', data=df)
plt.show()
```

Output:



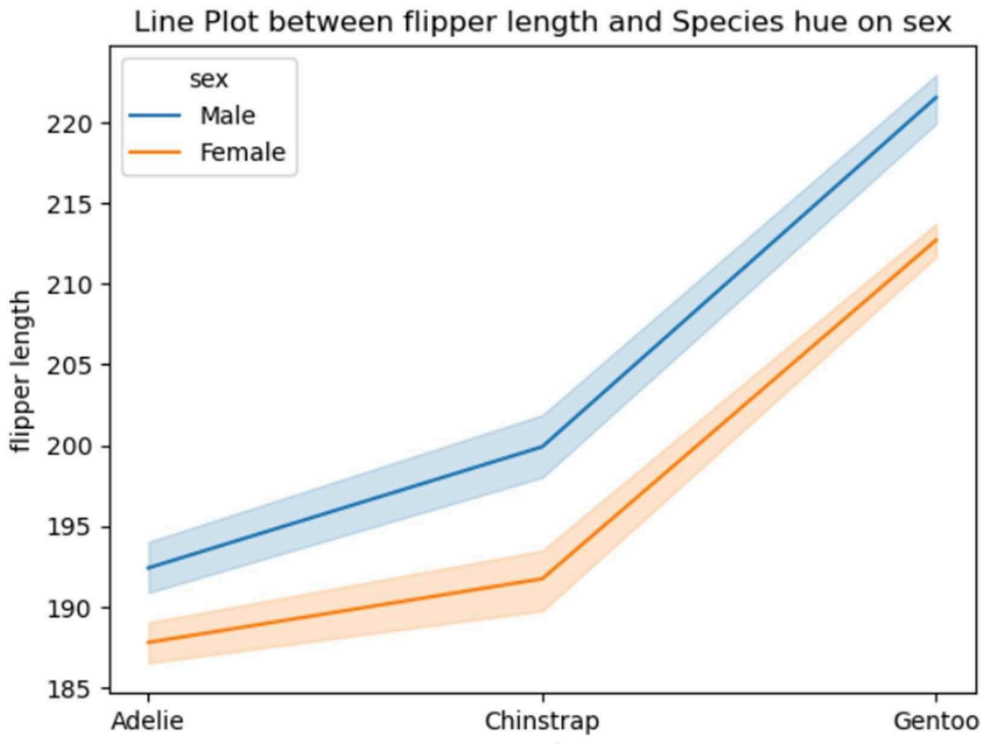
Description:

Here the different colors on sex are due to the parameter hue. Since hue is on “sex” column there is a difference for male and female.

LINE PLOT

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

sns.lineplot(y='flipper_length_mm', x='species',hue='sex', data=df)
plt.title('Line Plot between flipper length and Species hue on sex')
plt.ylabel('flipper length')
plt.xlabel('Species')
plt.show()
```

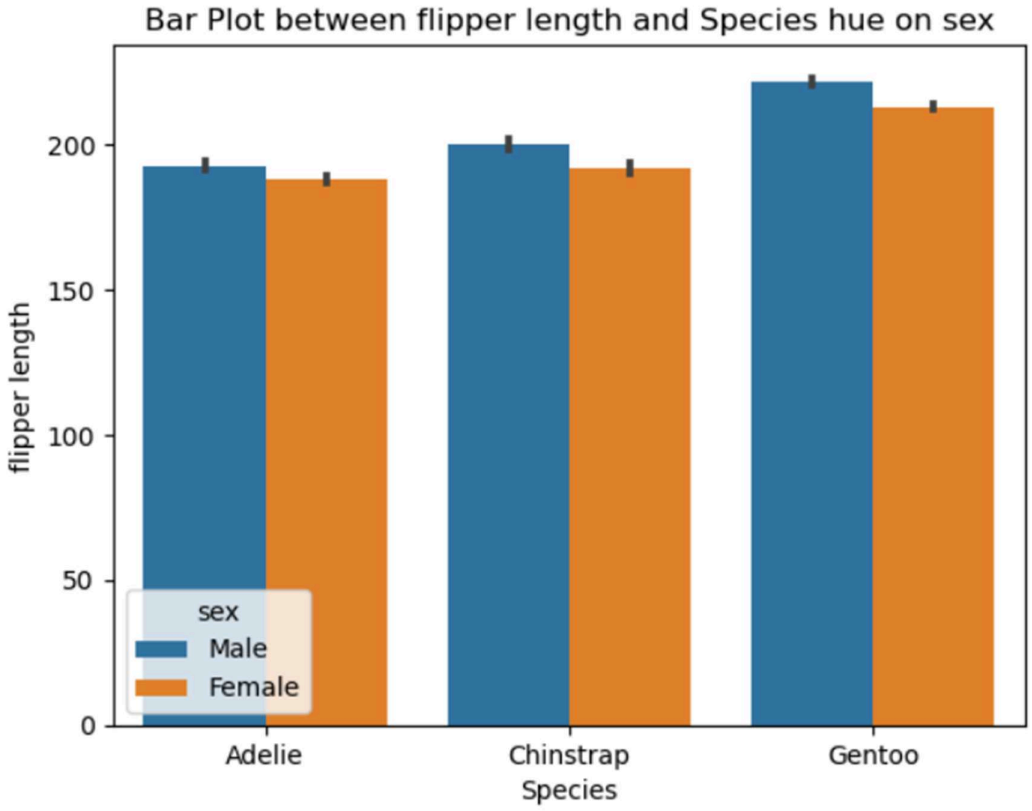
Output:



BAR PLOT

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

sns.barplot(y='flipper_length_mm', x='species',hue='sex', data=df)
plt.title('Bar Plot between flipper length and Species hue on sex')
plt.ylabel('flipper length')
plt.xlabel('Species')
plt.show()
```

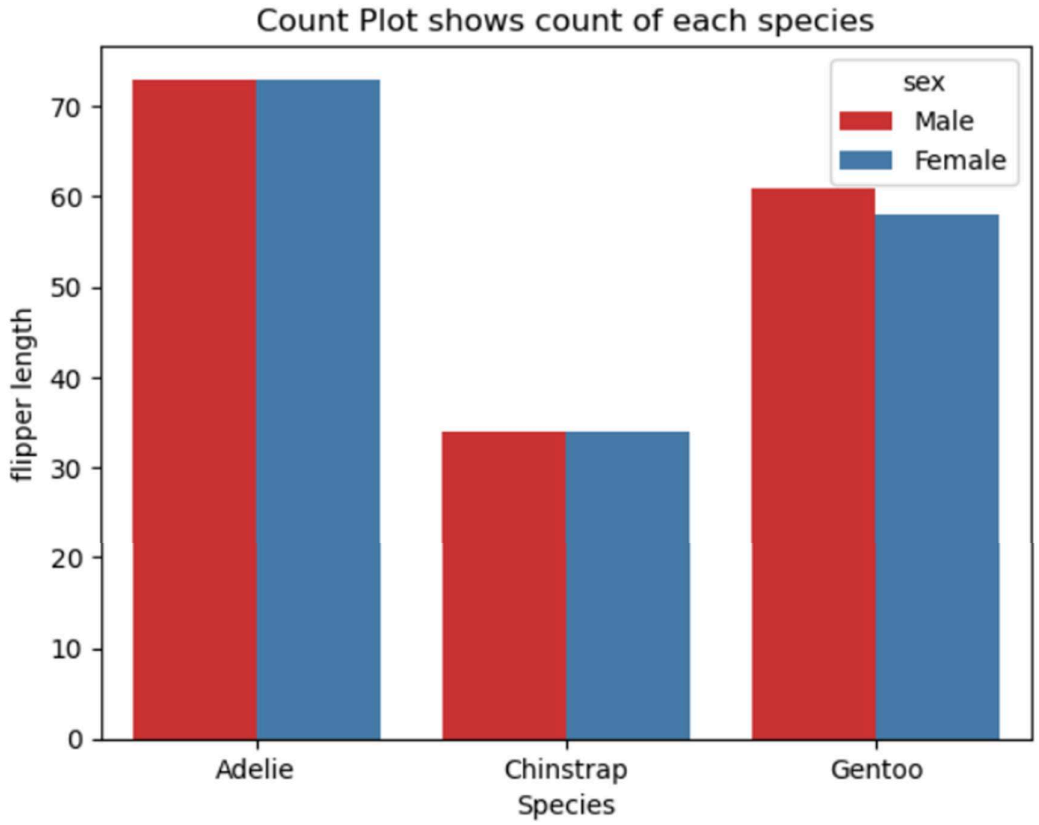
Output:



COUNT PLOT

```
1 import matplotlib.pyplot as plt
2 import pandas as pd
3 import seaborn as sns
4
5 sns.countplot(x='species',hue='species', data=df, palette="Set1")
6 plt.title('Count Plot shows count of each species')
7 plt.ylabel('flipper length')
8 plt.xlabel('Species')
9 plt.show()
```

Output:



Description:

The count plots give the count of the data passed.

Here x is the species, hue is sex and palette as Set1.

Palette sets the color and it is Set1 is the default palette in seaborn.

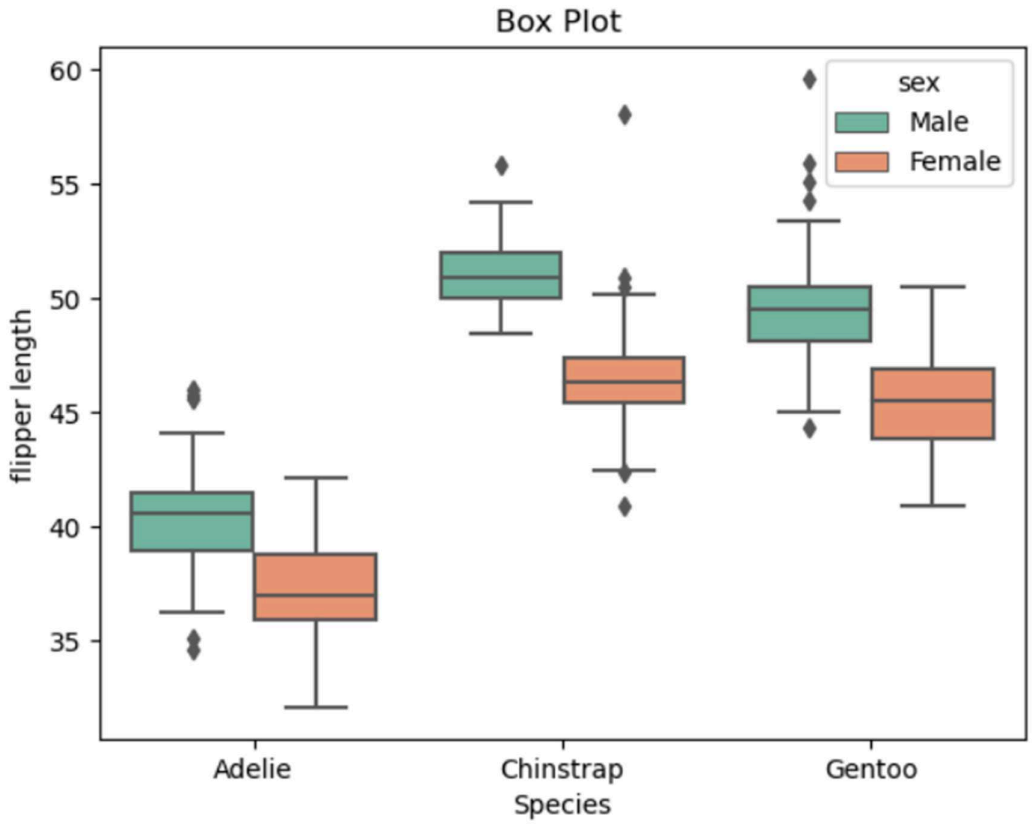
BOX PLOT

Box plot shows the quartiles where the data lies in interquartile range will be inside the box. The points which are away from the whiskers are outliers.

Code snippet:

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

sns.boxplot(x='species',y="bill_length_mm",hue='sex', data=df, palette="Set2")
plt.title('Box Plot ')
plt.ylabel('flipper length')
plt.xlabel('Species')
plt.show()
```

Output:



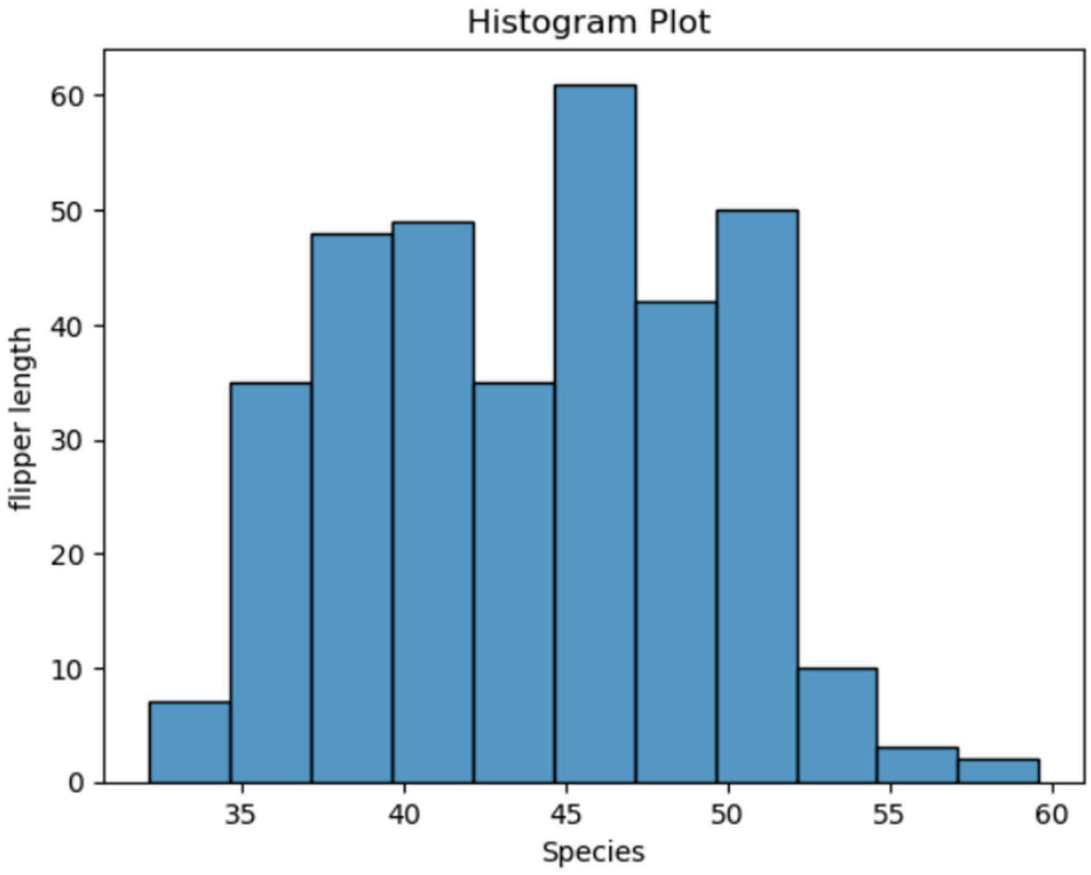
Description:

The parameter x should be numeric as the plot shows the difference between categories.

HISTOPLOT

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

sns.histplot(x='bill_length_mm', data=df, palette="Set2")
plt.title('Histogram Plot ')
plt.ylabel('flipper length')
plt.xlabel('Species')
plt.show()
```

Output:



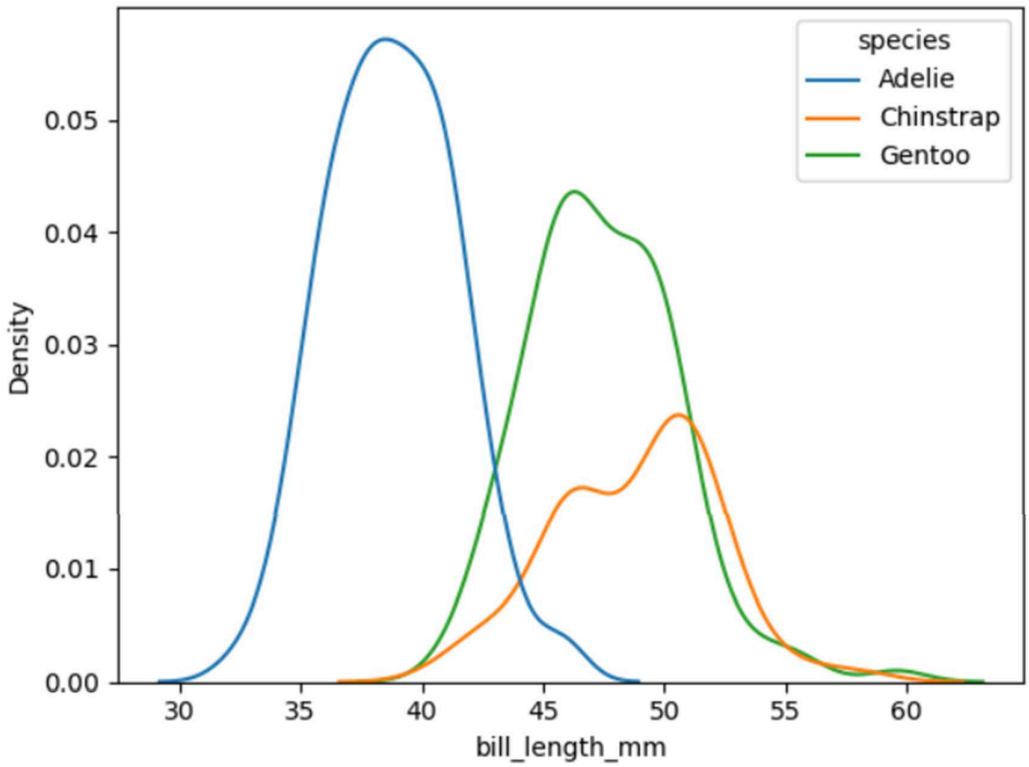
KDEPLOT

KDE stands for kernel density estimate. It creates a curve based on probability. It creates a single graph for multiple data samples.

Code snippet:

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

sns.kdeplot(x='bill_length_mm',hue='species',data=df )
plt.show()
```

Output:



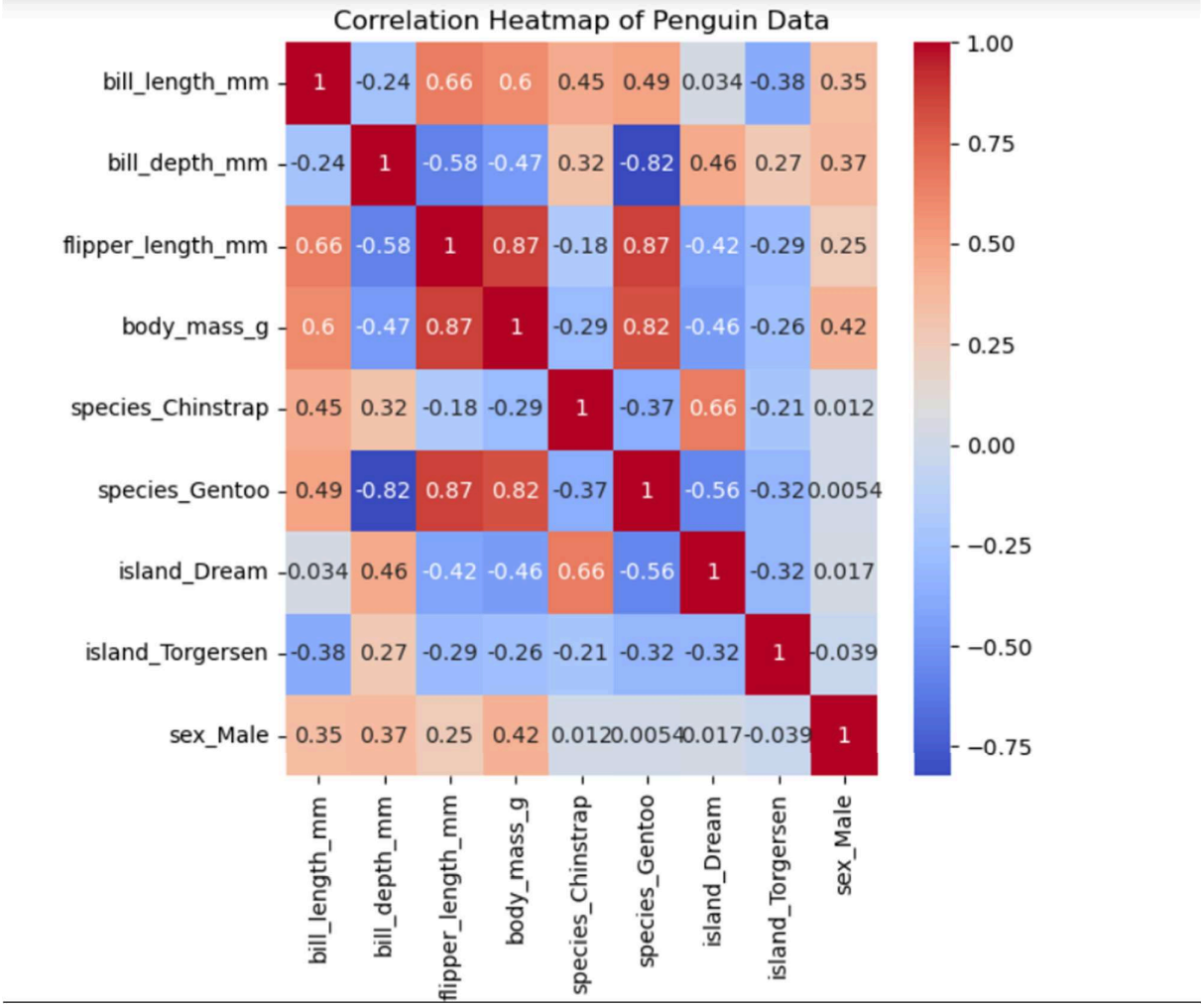
HEATMAP

Heatmaps uses correlation matrix and visualizes data. The data points where the higher values get brighter colors and lower values get darker colors.

Code snippet:

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

penguins_encoded = pd.get_dummies(df, columns=['species','island','sex'], drop_first=True)

correlation_matrix = penguins_encoded.corr()

plt.figure(figsize=(6, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap of Penguin Data")
plt.show()

```

Output:

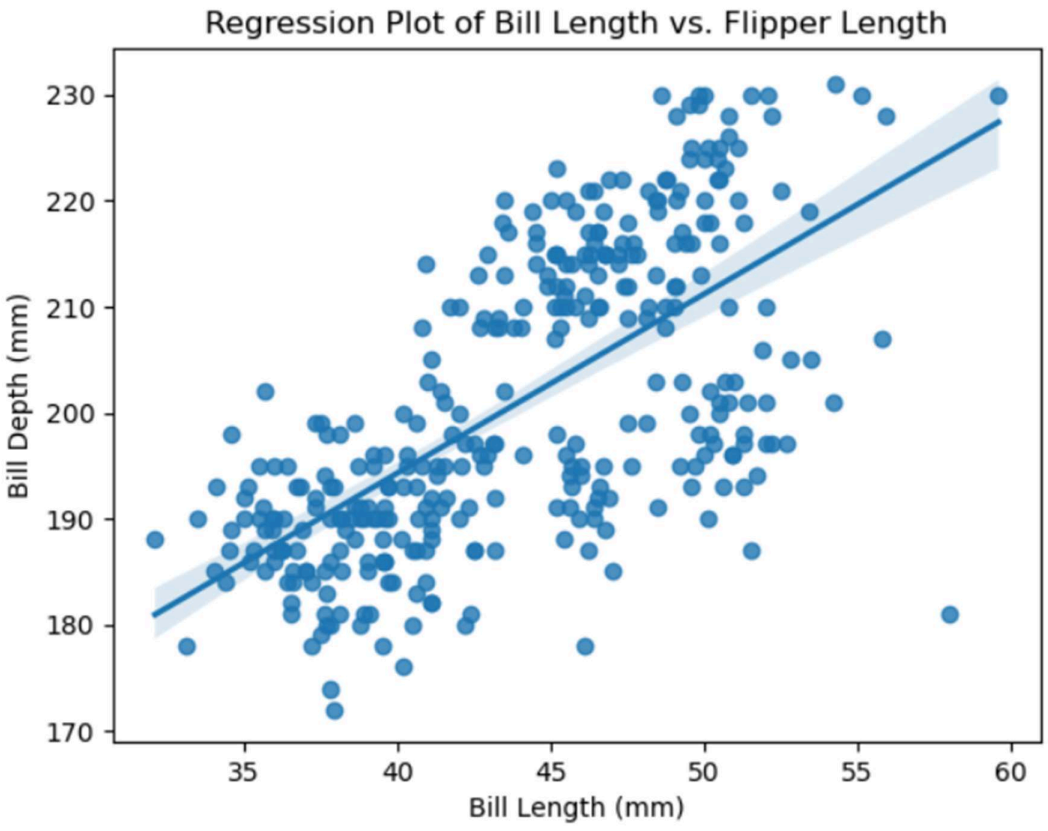


REGPLOT

Regression plots show the relationship between variables along with the regression line. Code snippet:

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

sns.regplot(x="bill_length_mm", y="flipper_length_mm", data=penguins)
plt.title("Regression Plot of Bill Length vs. Flipper Length")
plt.xlabel("Bill Length (mm)")
plt.ylabel("Bill Depth (mm)")
plt.show()

```

Output:



COMPARISON OF MATPLOTLIB AND SEABORN

Matplotlib:

- Core Library: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides a low-level interface for creating plots with fine-grained control over every aspect of the figure.
- Flexibility: Matplotlib allows users to create any type of plot imaginable, from basic line plots and scatter plots to complex 3D plots and geographical maps.
- Mature: Matplotlib has been around for a long time and is a well-established library in the Python ecosystem. Many other libraries, including Seaborn, are built on top of Matplotlib.
- Customization: Matplotlib offers extensive customization options, allowing users to customize every aspect of a plot, including colors, line styles, markers, fonts, annotations, and more.
- Integration: Matplotlib can be easily integrated with other libraries and frameworks, such as NumPy, Pandas, SciPy, which makes it a versatile choice for data visualization in various contexts.

Advantages of Matplotlib:

- High degree of customization and control over plots.
- Wide range of plot types and styles.
- Strong community support and extensive documentation.
- Well-suited for creating publication-quality figures and graphics.

Seaborn:

- High-Level Interface: Seaborn is built on top of Matplotlib and provides a high-level interface for creating attractive and informative statistical graphics. It simplifies the process of creating complex visualizations by providing default aesthetics and convenient functions for common tasks.
- Statistical Plotting: Seaborn specializes in statistical plotting and offers a variety of plot types specifically designed for visualizing relationships in data, such as scatter plots, bar plots, box plots, violin plots, pair plots, and more.
- Aesthetics: Seaborn comes with built-in themes and color palettes that make it easy to create visually appealing plots with minimal effort. It also provides tools for fine-tuning plot aesthetics, such as controlling the size of plot elements and adjusting the color palette.
- Integration with Pandas: Seaborn seamlessly integrates with Pandas dataframes, allowing users to easily plot data stored in Pandas data structures without the need for extensive data manipulation.

Advantages of Seaborn:

- Simplified syntax and high-level functions for creating complex plots.
- Built-in support for statistical plotting and data exploration.
- Attractive default aesthetics and color palettes.
- Seamless integration with Pandas dataframes for data visualization.
- Ideal for exploratory data analysis and quick visualization of relationships in data.

Overall, the choice between Matplotlib and Seaborn depends on the specific requirements of the data visualization task. Matplotlib is preferred for its flexibility and fine-grained control over plots, while Seaborn is preferred for its ease of use, statistical plotting capabilities, and attractive default aesthetics. Many users combine both libraries, leveraging the strengths of each as needed.