Submitted by: Angel Ruban

VISUALIZATION

Data visualization refers to presenting information in the form of charts, graphs, and plots so that patterns and insights become easier to understand. Instead of reading long tables of numbers, visual representations help us quickly observe trends, comparisons, and relationships.

Python provides several libraries for data visualization, such as Matplotlib, Seaborn, Plotly, and Bokeh.

Let us dive deeper into on Matplotlib and Seaborn, two of the most widely used visualization tools.

MATPLOTLIB

Matplotlib is one of the oldest and most popular plotting libraries in Python. Created by John Hunter in 2002, it is mainly used for producing 2D plots. It is built on top of NumPy and gives users very fine control over how the final graph looks.

A plot of matplotlib contains:

1. Figure
2. Axes
3. Axis
4. Artists

Figure

This is the overall container that holds all the plots. 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

These generate ticks and limits for both the x-axis and y-axis.

Artists

Everything you see on the figure (lines, texts, labels, shapes) is called an Artist.

PYPLOT

pyplot is a commonly used sub library of Matplotlib that provides simple functions to create different types of graphs. It includes methods for line charts, bar charts, scatter plots, histograms, pie charts, and more.

We import pyplot from matplotlib as shown below:

```
[1]: 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

A line plot is one of the basic types of graphs. It connects data points using a continuous line to show trends over time or another continuous variable.

The default plot() is used for line charts.

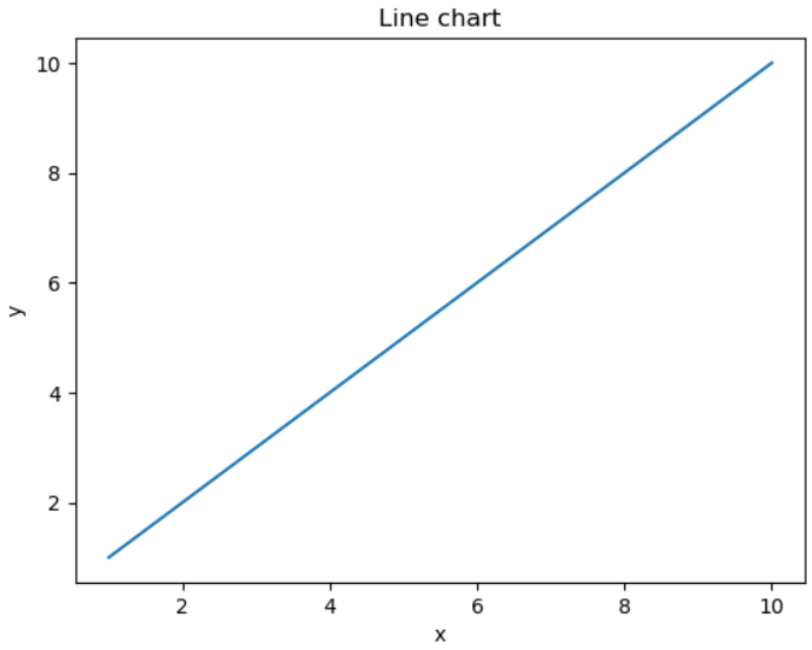
Code snippet:

```
[3]: 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() is used to generate an array in the specified range.

plot() is used for plotting the line chart.

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

show() is used to display the graph.

BAR GRAPH

Bar graph represents data using rectangular bars, where the height of each bar shows the frequency of a specific element. The bar() function is used to draw bar graphs and includes parameters like x, y, width, and color.

bar(x, y, color, width)

Code snippet:

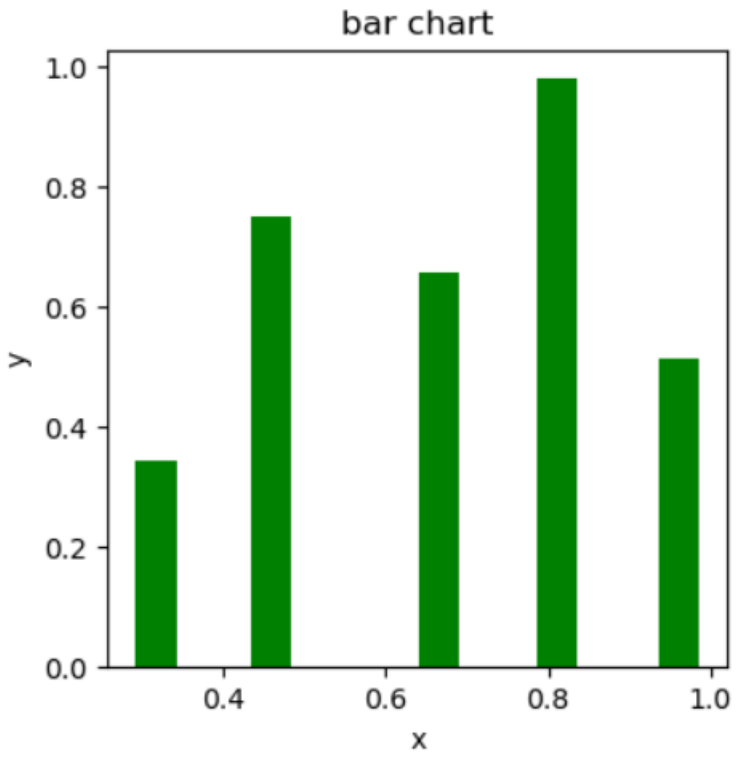
[5]:
```
import matplotlib.pyplot as plt
import numpy as np

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:

```
[0.31685312 0.66508179 0.96189492 0.81063504 0.45769563]
```



Description:
random.rand(5) is used to create a random array.

In this graph, the bar() function is applied with the width parameter to control the width of the rectangular bars and the color parameter to specify the bar colors.

HISTOGRAM

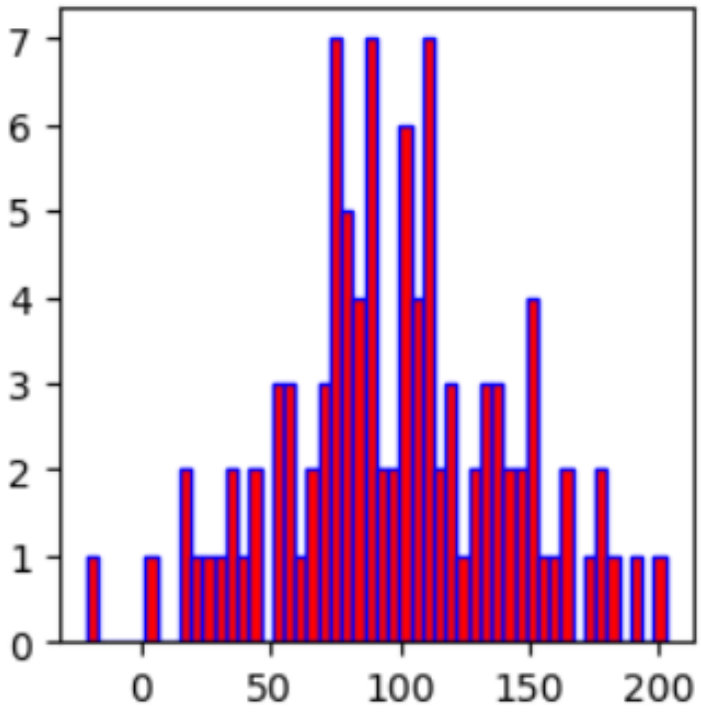
Histogram is a type of bar in which data is grouped into ranges. The hist() function is used to create a histogram and includes parameters such as x, bins, color, and edgecolor.

Code snippet:

```
[20]: 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 determines the number of intervals on x axis.

color sets the fill color of the rectangular bars.

edge color defines the color of the borders separating the rectangular bars.

SCATTER PLOT

Scatter plot represents data points using dots to show the relationship between variables. The scatter() function is used to create scatter plots, and multiple datasets can be plotted together by using different colors. Legends help distinguish between these datasets.

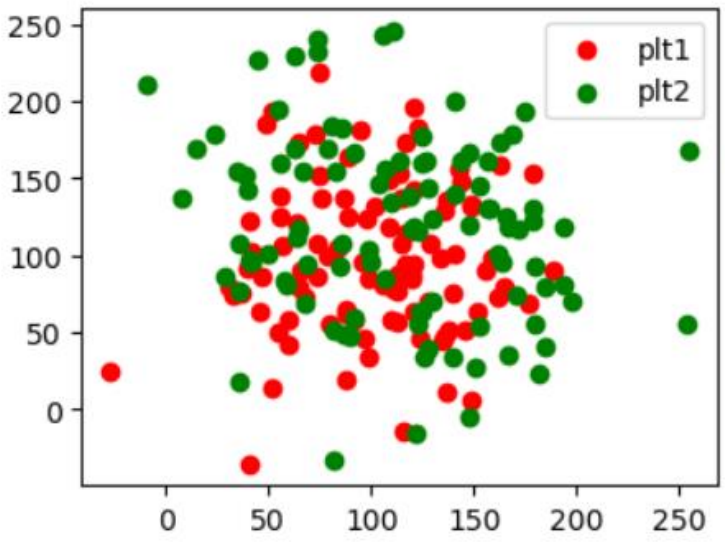
Code snippet:

```
[40]: 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, while x2, y2 belong to another.

The scatter() function is used with parameters x, y, and c (for color).

legend() function adds labels to differentiate between the datasets in the plot.

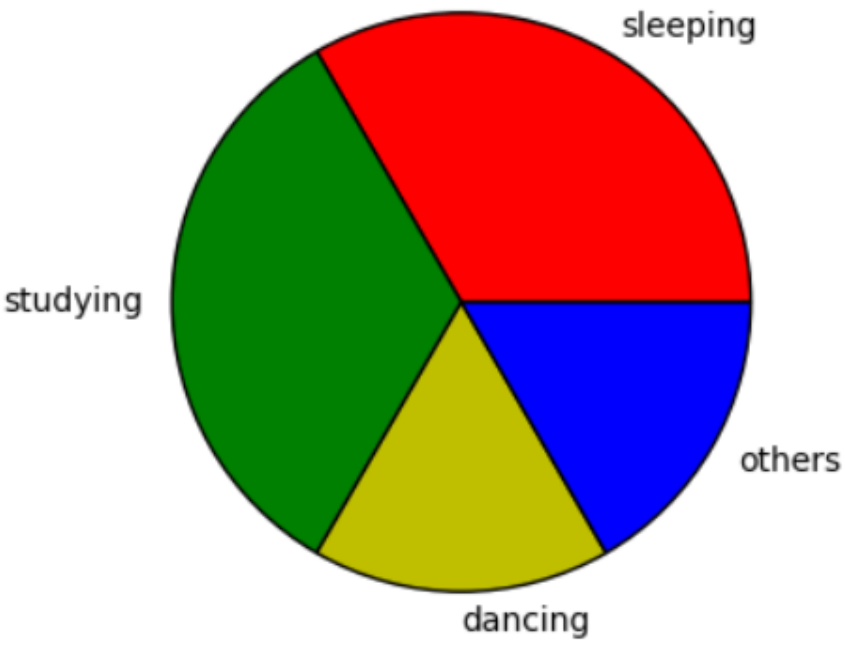
PIE CHART

Pie charts are used to plot data of same type, showing how different elements contribute to the whole in terms of percentages.

Code snippet:

```
[61]: from matplotlib import pyplot as plt
      import numpy as np
      time = [8,8,4,4]
      color = ["r","g","y","b"]
      work = ["sleeping","studying","dancing","others"]
      fig = plt.figure(figsize=(4, 4))
      plt.pie(time, labels=work, wedgeprops={"edgecolor":"black","linewidth":1}, colors=color)
      plt.show()
```

Output:



Description:

In this code, time represents the data values, and labels are assigned using the labels parameter.

wedgeprops attribute is used to set the border width and color that separates the sections of the pie chart.

color array contains the colors to each element.

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:

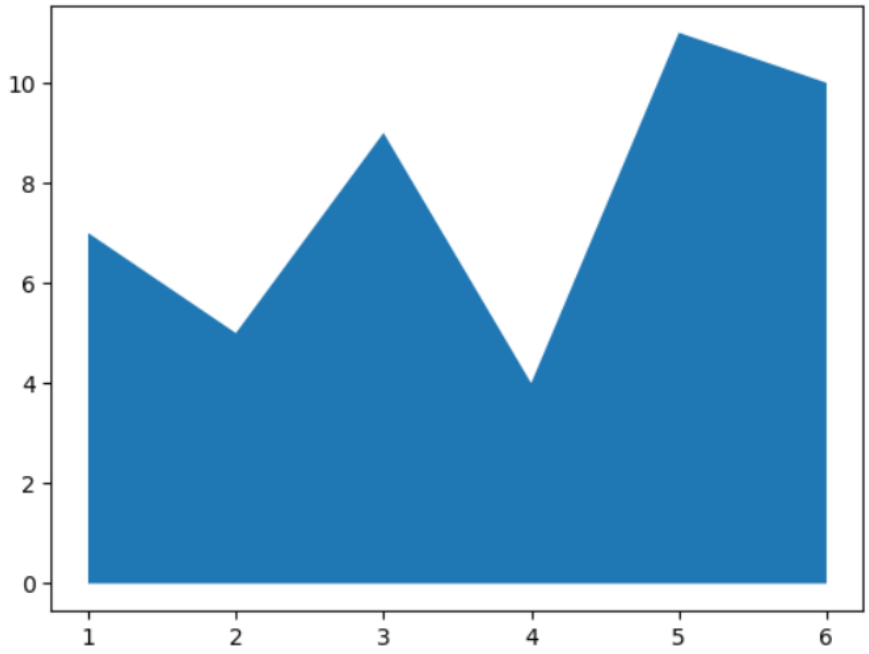
Using fill_between() function

```
[62]: 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()
```

Output:



Code snippet:

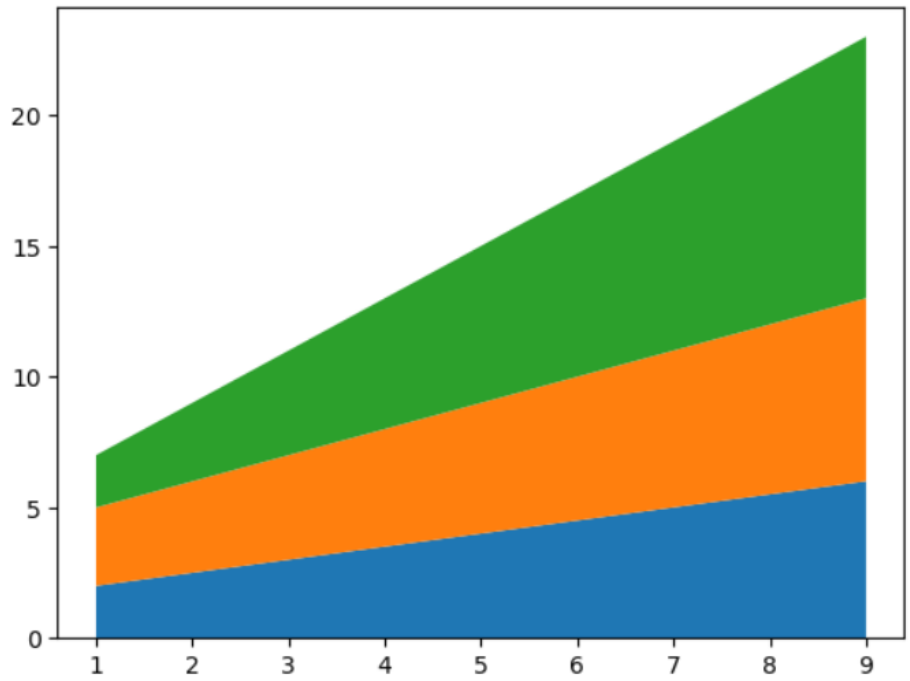
Using stackplot() function

[63]:
```
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()
```

Output:



Description:

fill_between() function uses x and y as parameters.

stackplot() function takes parameters x, y1, y2, y3, where y1, y2, y3 represent different data groups.

Legends can be added by specifying the labels parameter in stackplot() and positioning them using plt.legend() with the loc attribute.

SEABORN

Seaborn is a data visualization library in Python that is built on top of matplotlib. It is more advanced than matplotlib and provides built-in features such as default styles and color palettes. Seaborn offers various categories of plots, including relational plots, categorical plots, distribution plots, regression plots, and matrix plots.

Let us look at the different plots available under 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 Regression Plots:

- regplot()
- lmplot()

5 Matrix Plots:

- heatmap()
- clustermap()

Here are some sample codes for some of the graphs.

Scatter Plot

Scatter plot shows the relationship between two numerical variables using data points. Each point represents an observation, which helps in identifying patterns, trends, correlations, and outliers in the data.

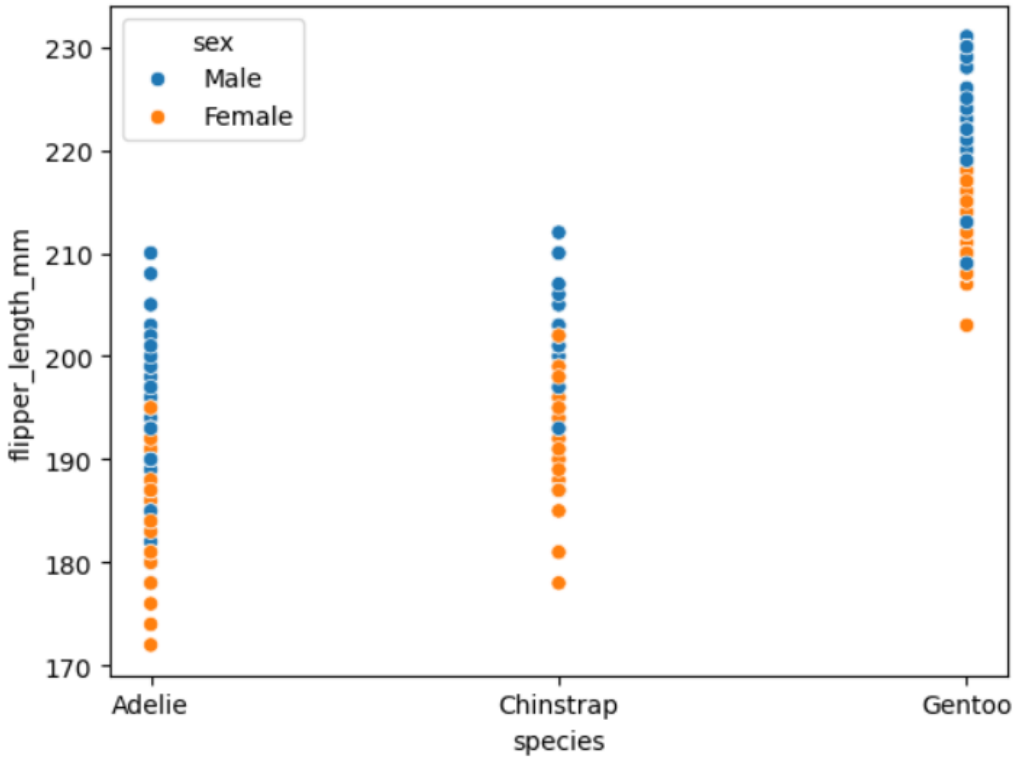
Code snippet:

```
[64]: import seaborn as sns
      import matplotlib.pyplot as plt
      import pandas as pd

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

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

Output:



Description:

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

LINE PLOT

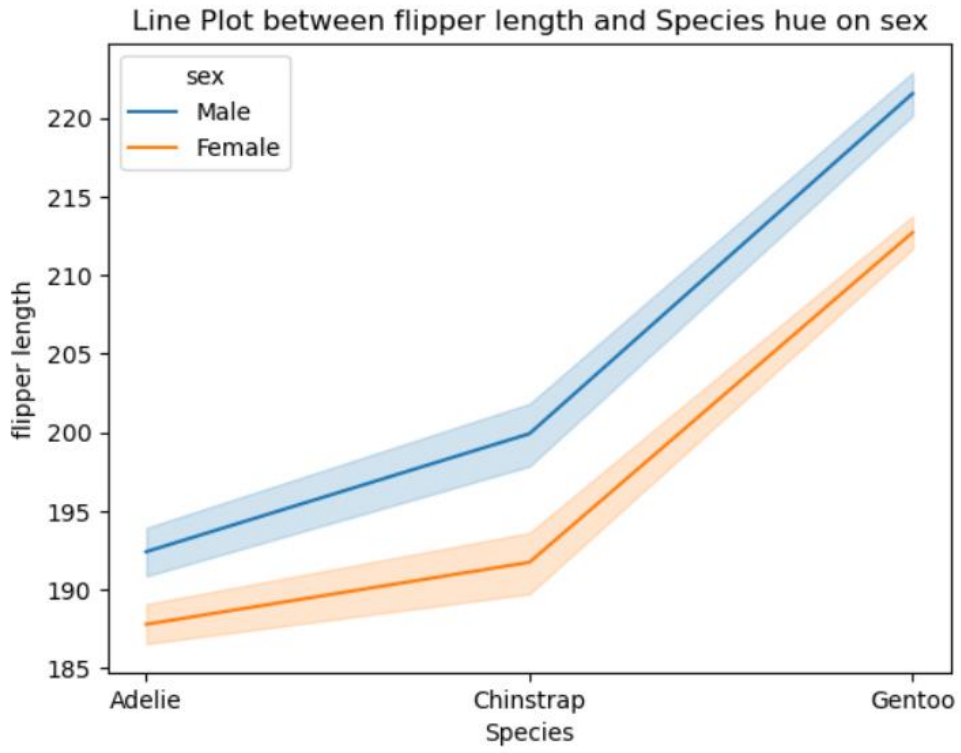
Line plot shows the change or trend of numeric data over a continuous interval, usually time. The data points are connected using straight lines, making it easy to observe increases, decreases, and overall patterns.

Code snippet:

```
[66]: 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:



Description:

In this plot, the x-axis represents species and the y-axis represents flipper length (mm).

The parameter y must be numeric as it shows measured values, while x represents categories.

The hue parameter (sex) is used to differentiate the lines for males and females, helping to compare flipper length trends across species based on sex.

BAR PLOT

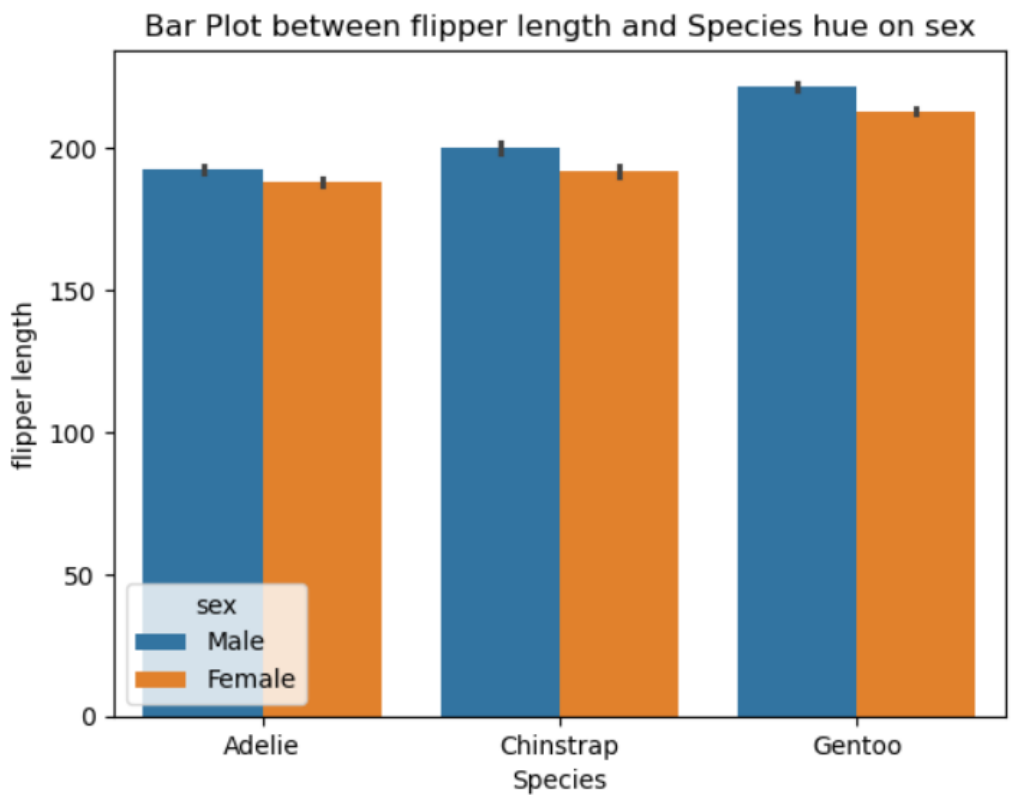
Bar plot shows the comparison of numeric values across different categories. The height of each bar represents the value of the numeric variable for a specific category.

Code snippet:

```
[67]: 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:



Description:

This bar plot compares the average flipper length (mm) across different species.

The hue parameter (sex) divides each species into male and female groups, allowing comparison of flipper length between sexes within the same species.

COUNT PLOT

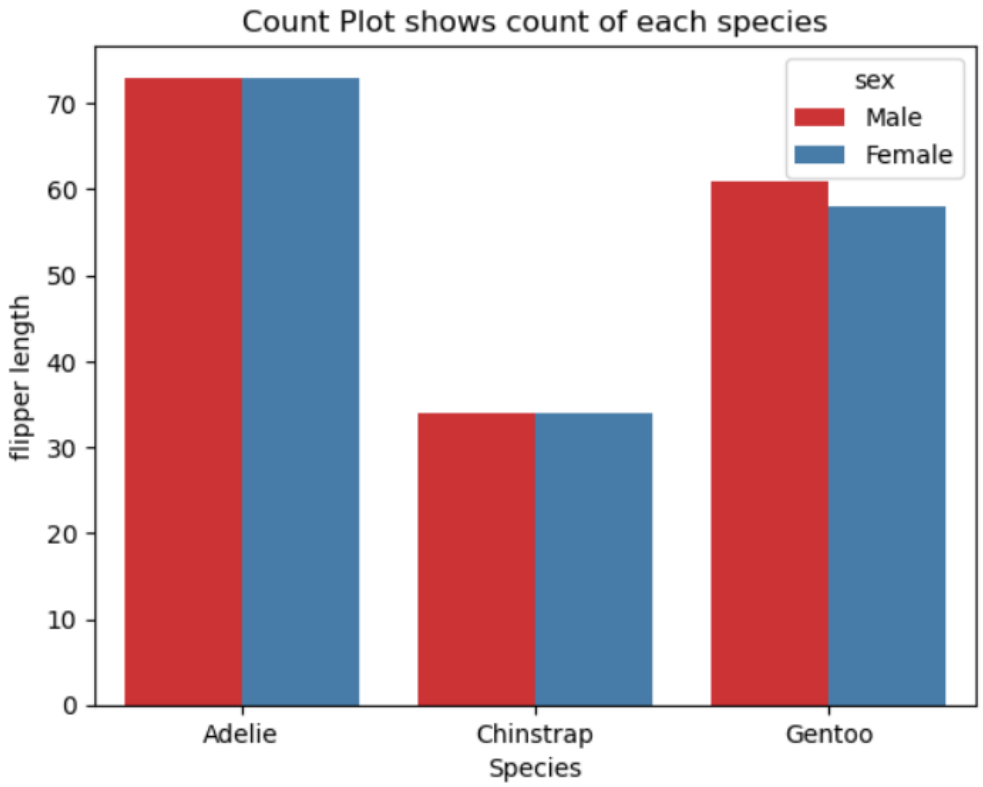
Count plot shows the number of observations in each category. It is mainly used to visualize how data is distributed across categorical variables.

Code snippet:

```
[75]: import matplotlib.pyplot as plt
      import pandas as pd
      import seaborn as sns

      sns.countplot(x='species', hue='sex', data=df, palette="Set1")
      plt.title('Count Plot shows count of each species')
      plt.ylabel('flipper length')
      plt.xlabel('Species')
      plt.show()
```

Output:



Description:

The count plots give the count of the data passed.

In this plot, x represents species, hue is set to sex, and the palette used is Set1.

The palette parameter controls the colors, and Set1 is the default color palette in Seaborn.

BOX PLOT

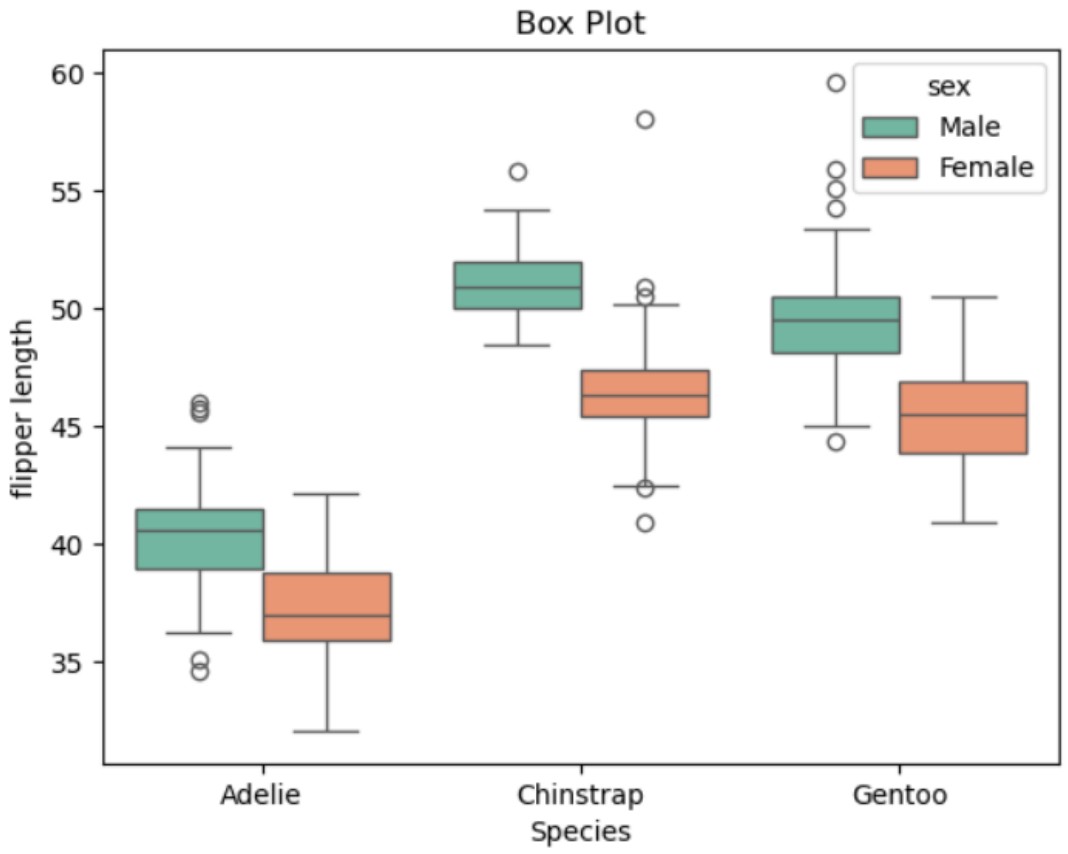
Box plot represents the quartiles of the data, where the interquartile range is shown inside the box. Data points that lie beyond the whiskers are outliers.

Code snippet:

```
[79]: 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.

HISTPLOT

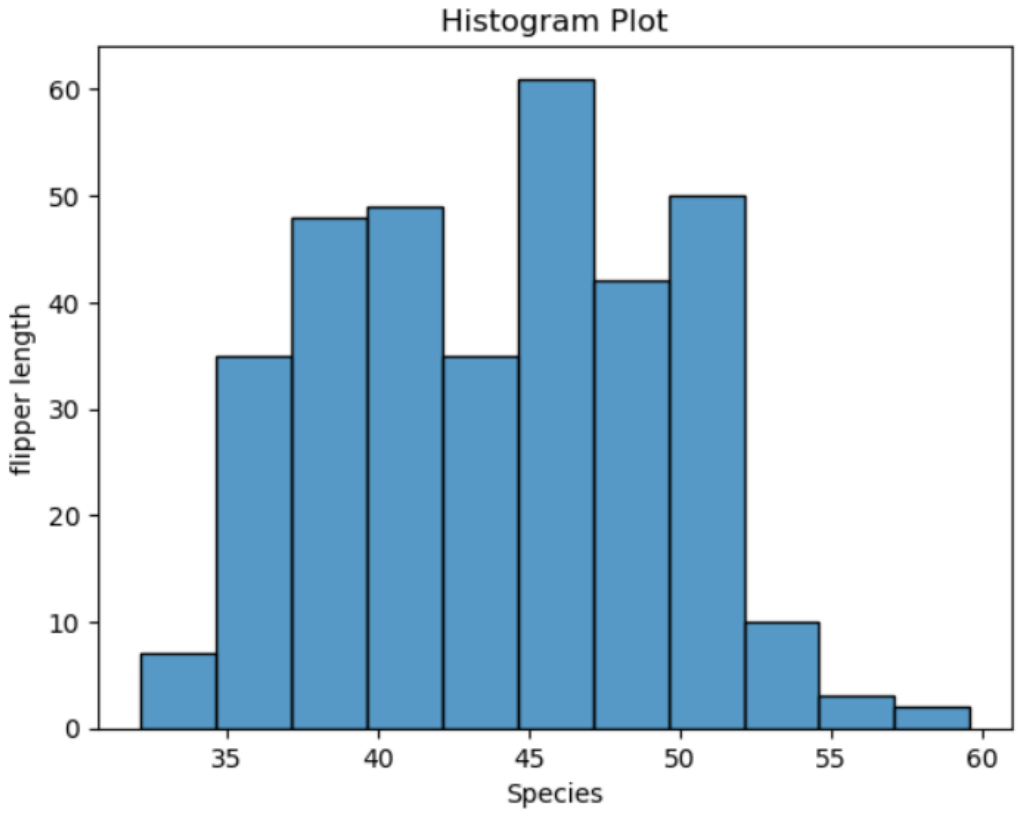
Histogram shows the distribution of a numeric variable by grouping data into bins(intervals). It helps in understanding the frequency, spread, central tendency and skewness of the data.

Code snippet:

```
[88]: 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:



Description:

The parameter x should be numeric.

The plot shows the frequency of values in different bins.

Useful for identifying the central tendency, spread, and skewness of the data.

KDEPLOT

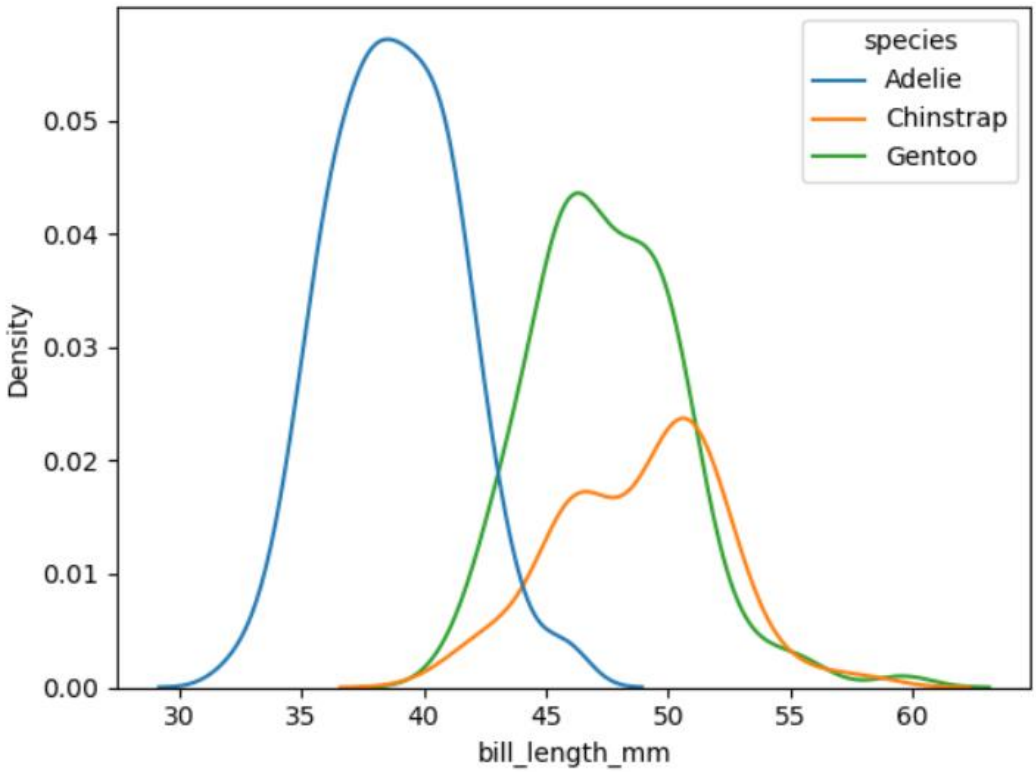
KDE stands for Kernel Density Estimate. It creates a curve as based on probability. It creates single graph for multiple data samples.

Code snippet:

```
[89]: 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:



Description:

The parameter x should be numeric, and hue can be used to separate distributions by categories.

KDE plots are useful for comparing distributions, observing skewness, and identifying peaks in the data.

HEATMAP

Heatmaps represents data using a correlation matrix for visualization. Values with higher magnitude are displayed with brighter colors, whereas lower values are shown using darker colors.

Code snippet:

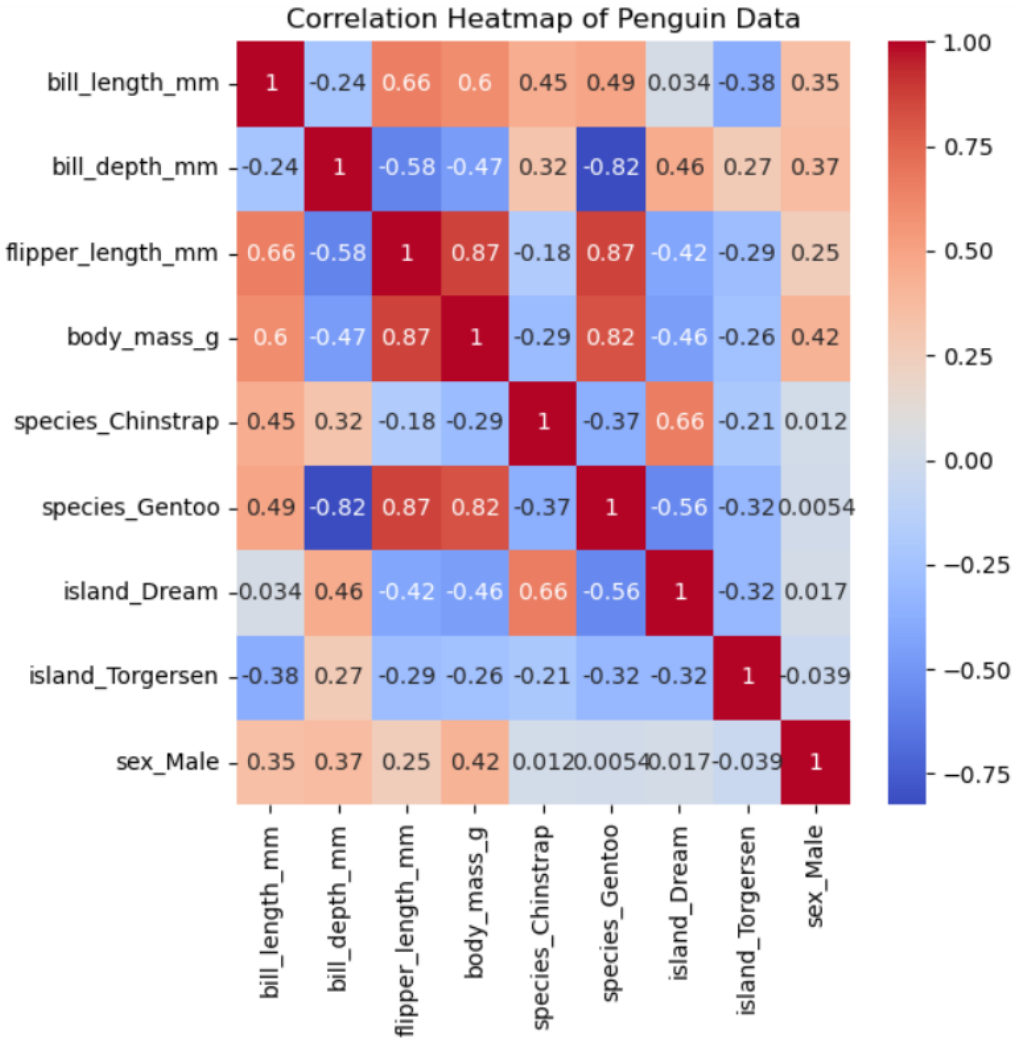
```
[91]: import seaborn as sns
      import matplotlib.pyplot as plt
      import pandas as pd

      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:



Description:

Categorical variables (species, island, sex) are converted into numeric form using one-hot encoding to enable correlation calculation.

The heatmap shows the correlation matrix, where color intensity indicates the strength and direction of relationships.

annot=True displays correlation values inside each cell, and cmap="coolwarm" highlights positive and negative correlations clearly.

REGPLOT

Regression plots shows the relationship between variables along with the regression line.

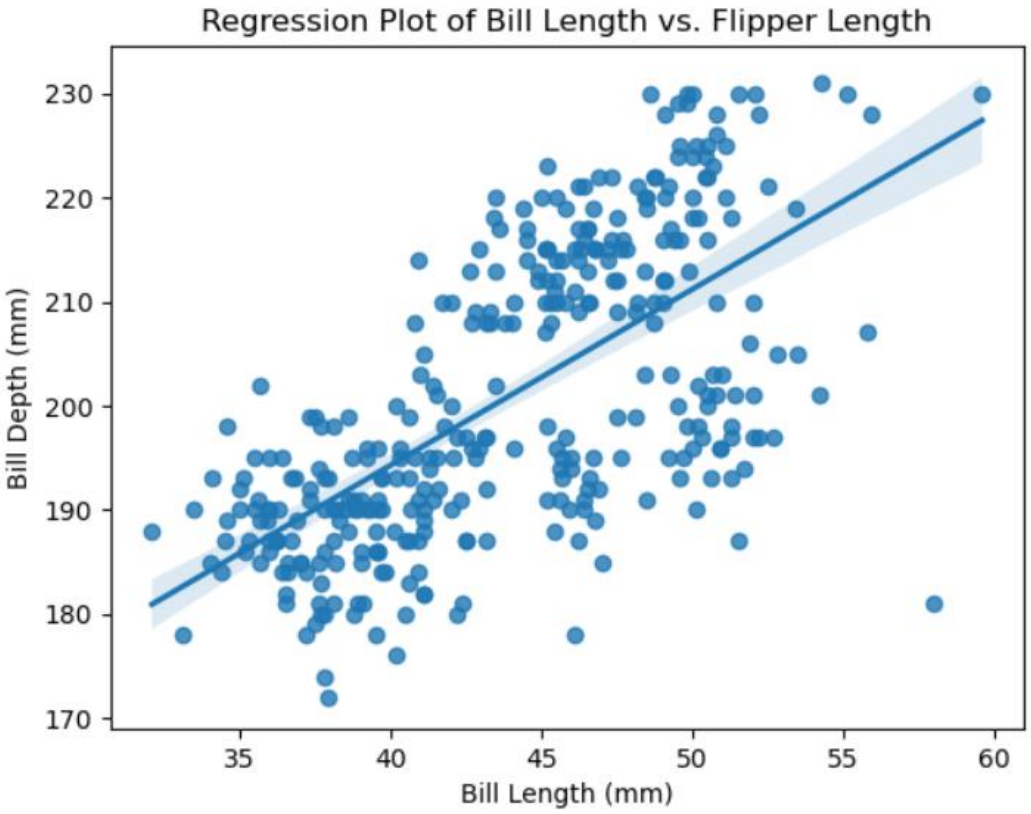
Code snippet:

[93]:
```
import seaborn as sns
import matplotlib.pyplot as plt

penguins = sns.load_dataset("penguins")

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:



Description:

Both x and y parameters must be numeric as the plot shows how one variable changes with respect to another.

The regression line indicates the direction and strength of the relationship between bill length (bill_length_mm) and flipper length (flipper_length_mm).

COMPARISON OF MATPLOTLIB AND SEABORN

Matplotlib:

- Core Library: Matplotlib is a powerful library used to create static, animated, and interactive visualizations in Python. It offers a low-level interface, giving detailed control over every component of a figure.
- Flexibility: It supports a wide variety of plots, ranging from simple line and scatter plots to advanced 3D visualizations and geographical maps.
- Mature: Being one of the oldest visualization libraries in Python, Matplotlib is stable and widely adopted. Many libraries, including Seaborn, are built on top of it.
- Customization: Matplotlib allows extensive customization, such as modifying colors, line styles, markers, fonts, annotations, and other plot elements.
- Integration: Matplotlib can be easily integrated with other libraries and frameworks, such as NumPy, Pandas, SciPy, making it a flexible and widely used option for data visualization in different scenarios.

Advantages of Matplotlib:

- High level of control and customization over plots.
- Supports a wide variety of plot types and styles.
- Strong community support with detailed documentation.
- Suitable for producing publication-quality graphs and figures.

Seaborn:

- High-Level Interface: Seaborn is developed on top of Matplotlib and offers a high-level interface to create clear and visually appealing statistical graphics. It reduces complexity by providing default styles and easy-to-use functions for common visualization tasks.
- Statistical Plotting: Seaborn focuses on statistical visualization and includes many plot types designed to show 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:

- Simple syntax with high-level functions for creating complex plots.
- Inbuilt support for statistical visualization and data analysis.
- Attractive default aesthetics and color palettes.
- Easy integration with Pandas dataframes for data visualization.
- Best suited for exploratory data analysis and quick visualization of data relationships.

Overall, selecting between Matplotlib and Seaborn depends on the needs of the data visualization task. Matplotlib is commonly chosen for its flexibility and detailed control over plot elements, whereas Seaborn is preferred for its simplicity, statistical plotting features, and visually appealing default styles. Many users combine both libraries and use the strengths of each as required.

RESOURCES

Matplotlib: https://matplotlib.org/stable/users/explain/quick_start.html#quick-start

Seaborn: https://seaborn.pydata.org/tutorial/introduction.html