VISUALIZATION

Visualization is the process of creating representations of different concepts. Data Visualization is when you create representations of data to better understand it.

Python has various libraries which make performing visualization easier. There are multiple libraries like matplotlib, seaborn, plotly etc.

We shall discuss about the two of the most prominent libraries for visualization in python: Matplotlib and Seaborn.

Matplotlib

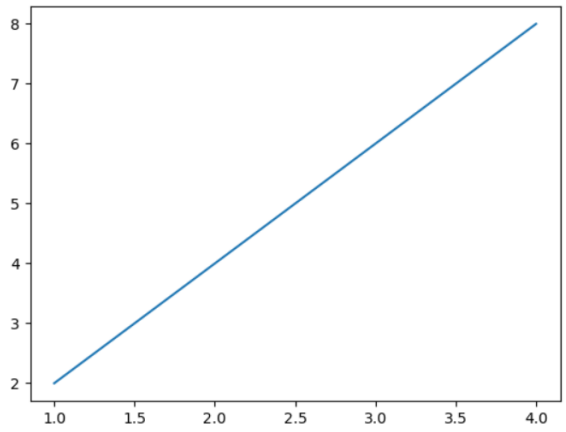
Matplotlib, introduced in 2002 by John Hunter, is a powerful library for creating static, animated, and interactive visualizations in Python. It is primarily used for plotting 2D plots and is built on top of NumPy.

It offers a wide range of plot types and customization options.

- ### Line Chart

A line chart is a basic plot where data points are connected by a continuous line. It is suitable for visualizing trends over time.

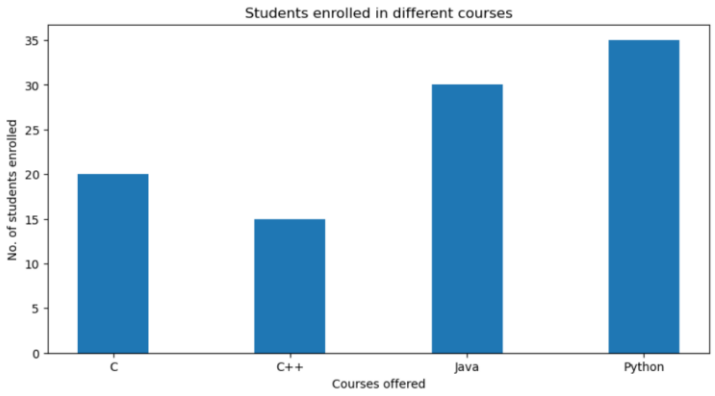
```
import matplotlib.pyplot as plt
import numpy as np
# Define X and Y variable data
x = np.array([1, 2, 3, 4])
y = x*2

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



- ### Bar Graph

A bar graph uses rectangular bars to represent data. It is effective for comparing quantities or frequencies across different categories.

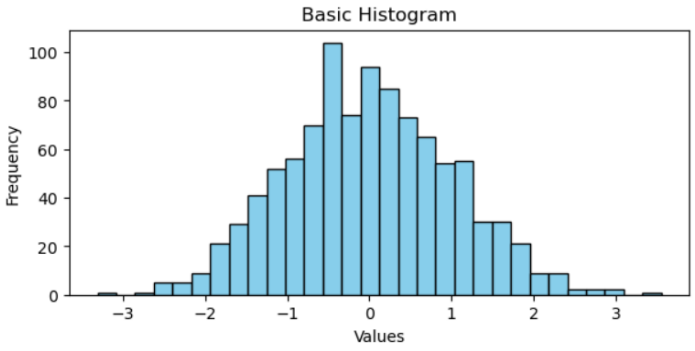
```
import numpy as np
import matplotlib.pyplot as plt

data = {'C':20, 'C++':15, 'Java':30,
        'Python':35}
courses = list(data.keys())
values = list(data.values())

fig = plt.figure(figsize = (10, 5))

plt.bar(courses, values,
        width = 0.4)

plt.xlabel("Courses offered")
plt.ylabel("No. of students enrolled")
plt.title("Students enrolled in different courses")
plt.show()
```



- ## Histogram

A histogram displays the distribution of a continuous variable by dividing the data into bins and counting the number of observations in each bin.

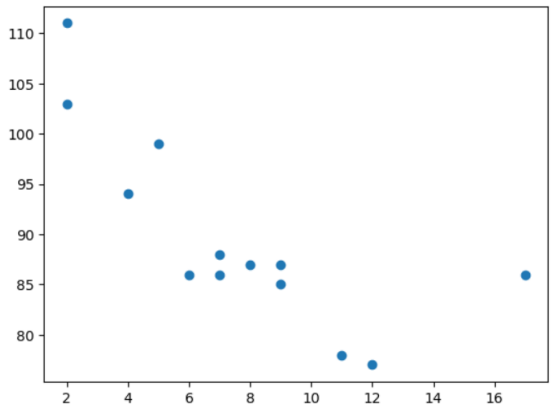
```
data = np.random.randn(1000)

plt.hist(data, bins=30,
         color='skyblue',
         edgecolor='black')

plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')

plt.show()
```



- ## Scatter Plot

A scatter plot uses dots to represent the relationship between two or more variables. It is useful for identifying patterns or correlations in data.

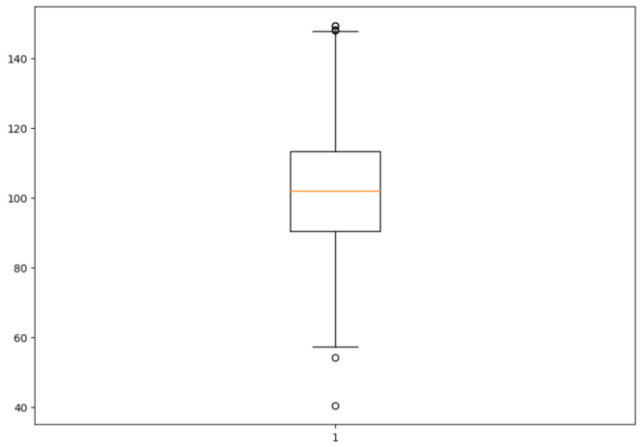
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

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



- ## Box Plot

A scatter plot uses dots to represent the relationship between two or more variables. It is useful for identifying patterns or correlations in data.

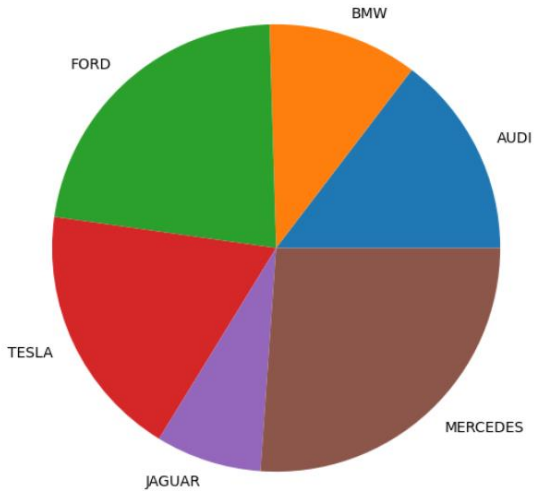
```
np.random.seed(10)
data = np.random.normal(100, 20, 200)

fig = plt.figure(figsize =(10, 7))

plt.boxplot(data)

plt.show()
```



- ## Pie Chart

A scatter plot uses dots to represent the relationship between two or more variables. It is useful for identifying patterns or correlations in data.

```
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]

fig = plt.figure(figsize=(10, 7))
plt.pie(data, labels=cars)

plt.show()
```



Seaborn

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

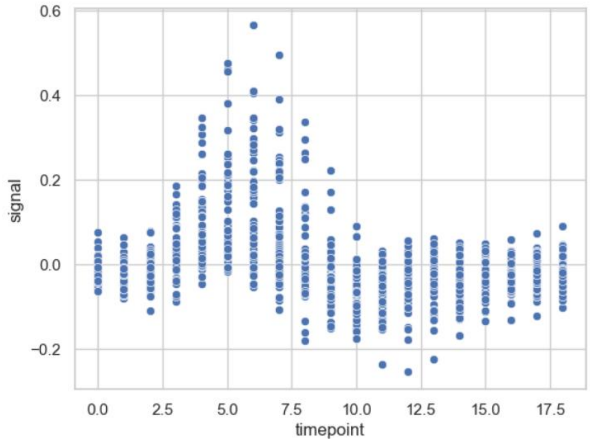
• Relational Plots

Relational plots are used to visualize the relationship between variables. They are effective for exploring correlations and trends in data.

Examples:

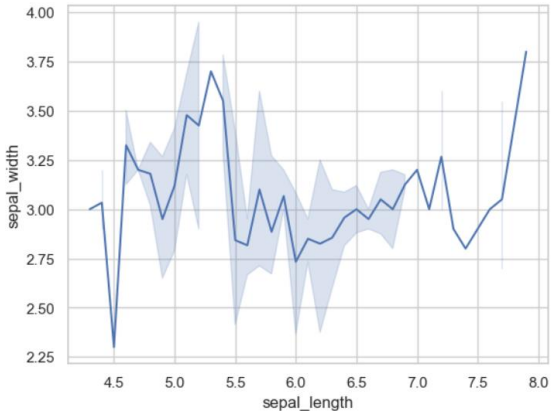
- Scatter Plot: Scatter plots are used to visualize the relationship between two continuous variables. Each point represents an observation in the dataset, and the position of the point on the plot corresponds to the values of the two variables.

```
import seaborn


seaborn.set(style='whitegrid')
fmri = seaborn.load_dataset("fmri")

seaborn.scatterplot(x="timepoint",
                    y="signal",
                    data=fmri)
```



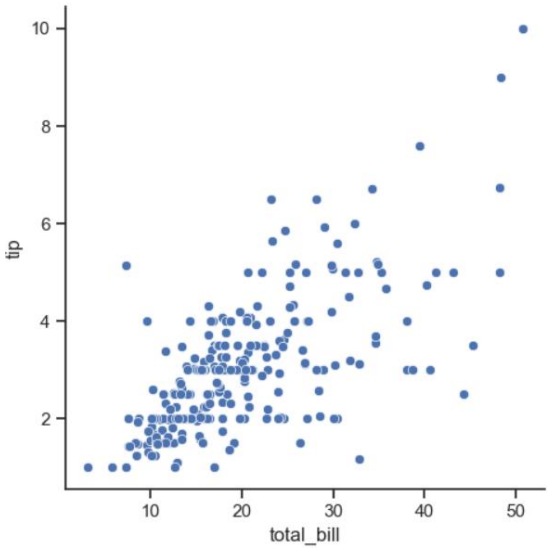
- Line Plot: Line plots are used to visualize trends or patterns over time or another continuous variable. They are particularly useful for visualizing time series data or data collected at regular intervals.

```
import seaborn as sns

data = sns.load_dataset("iris")

sns.lineplot(x="sepal_length",
             y="sepal_width",
             data=data)
```



- Relplot: Relational plots are a generalization of scatter plots and line plots. They can display the relationship between two variables using different markers or lines, depending on the value of a third variable.

```
import seaborn as sns
sns.set(style ="ticks")

tips = sns.load_dataset('tips')
sns.relplot(x ="total_bill",
            y ="tip",
            data = tips)
```



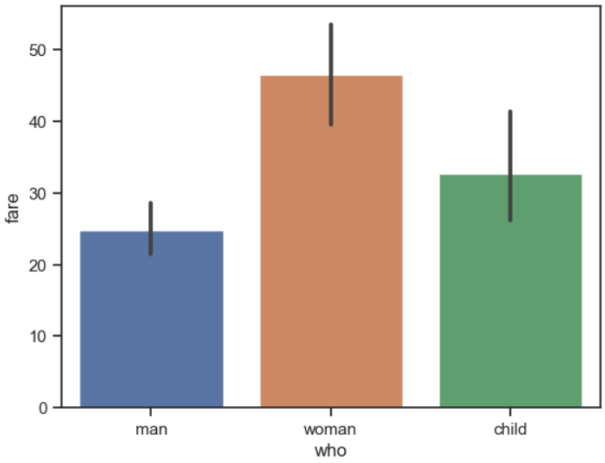
- ## Categorical Plots

Categorical plots are used to visualize the distribution of categorical variables. They are useful for comparing groups or categories within the data.

Examples:

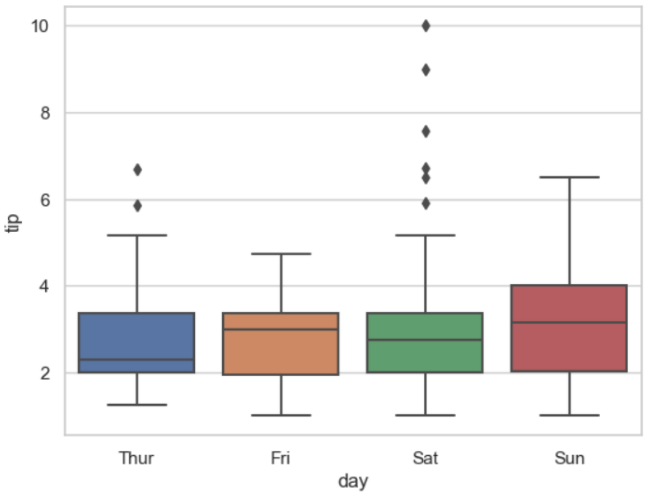
- Bar Plot: Bar plots are used to visualize the distribution of a categorical variable by displaying the frequencies or proportions of each category as rectangular bars.

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

df = sns.load_dataset('titanic')

sns.barplot(x = 'who',
            y = 'fare',
            data = df)

plt.show()
```



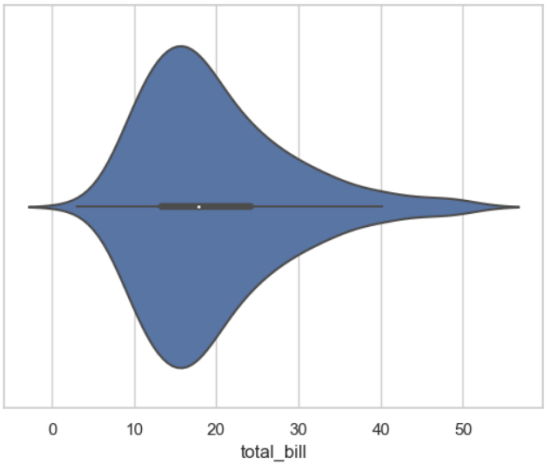
- Box Plot: Box plots, also known as box-and-whisker plots, are used to visualize the distribution of a continuous variable within different categories. They provide a visual summary of the central tendency, spread, and skewness of the data.

```
import seaborn
seaborn.set(style='whitegrid')
tip = seaborn.load_dataset('tips')
seaborn.boxplot(x='day',
                y='tip',
                data=tip)
```



- Violin Plot: Violin plots are used to visualize the distribution of a continuous variable across different categories. They combine elements of a box plot and a kernel density plot to provide insights into the data distribution.

```
import seaborn

seaborn.set(style="whitegrid")

tips = seaborn.load_dataset("tips")

seaborn.violinplot(x=tip["total_bill"])
```



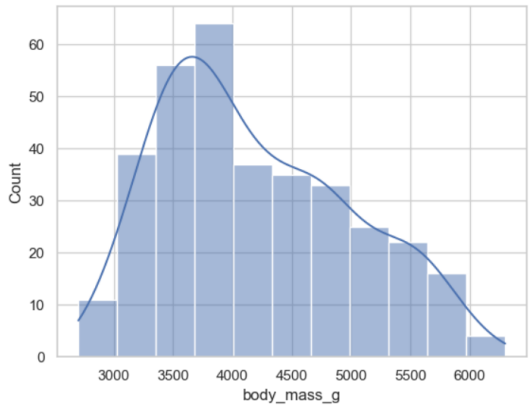
• Distribution Plots

Distribution plots are used to visualize the distribution of a single variable. They help understand the underlying distribution of the data.

Examples:

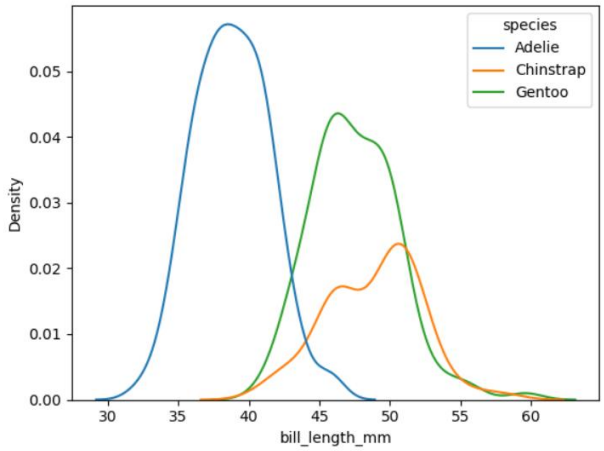
- Histogram: Histograms are used to visualize the frequency distribution of a continuous variable by dividing the data into bins and displaying the count or density of observations within each bin.

```
import numpy as np
import pandas as pd
import seaborn as sns

penguins = sns.load_dataset("penguins")

sns.histplot(data = penguins, x = "body_mass_g", kde = True)
```



- Kernel Density Estimation (KDE) Plot: KDE plots are used to estimate the probability density function of a continuous variable. They provide a smooth curve that represents the distribution of the data.

```
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()
```

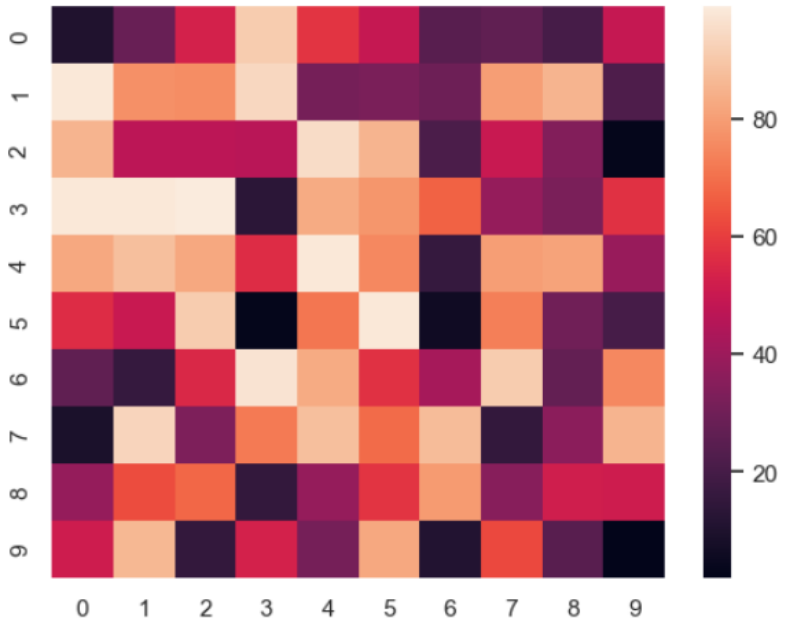


• Heatmap

Heatmaps are used to visualize two-dimensional data in a matrix format. They use colors to represent the values of the data, making it easy to identify patterns and relationships.

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

data = np.random.randint(low = 1,
                         high = 100,
                         size = (10, 10))
print("The data to be plotted:\n")
print(data)

hm = sn.heatmap(data = data)

plt.show()
```

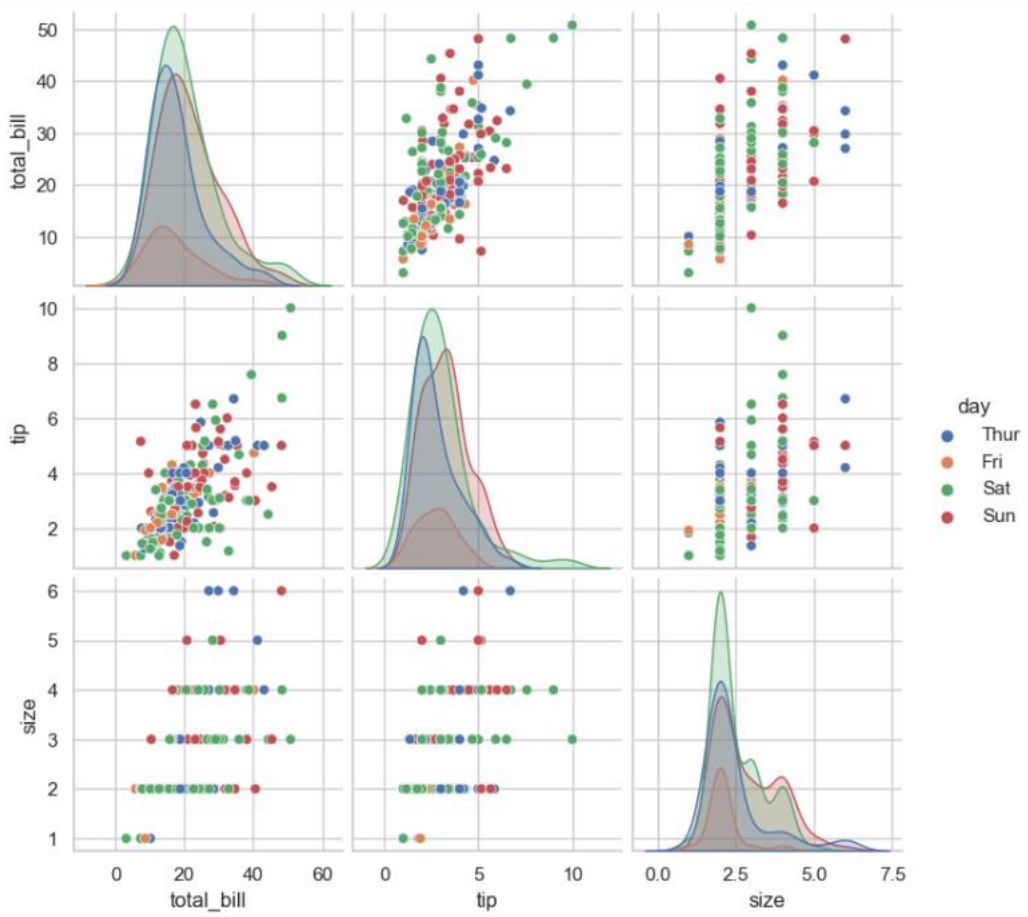
```
The data to be plotted:

[[10 28 53 91 58 49 24 26 20 49]
 [98 77 76 94 31 32 29 80 85 22]
 [85 47 47 46 95 85 21 50 34  3]
 [98 98 99 13 83 78 67 38 32 57]
 [82 88 82 56 98 75 16 80 81 39]
 [56 50 91  3 71 98  6 73 30 20]
 [26 16 55 97 83 57 42 91 27 75]
 [ 9 93 33 72 88 69 87 15 36 85]
 [38 63 68 15 38 58 79 35 52 51]
 [51 86 15 53 31 82 11 62 24  2]]
```



- ## Pair Plot

Pair plots are used to visualize pairwise relationships between variables in a dataset. They create a grid of scatterplots for each pair of variables, along with histograms for each variable on the diagonal.

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

df = seaborn.load_dataset('tips')

seaborn.pairplot(df, hue ='day')

plt.show()
```



- Comparison between matplotlib and seaborn libraries

Matplotlib

Core library: Matplotlib is a comprehensive plotting library with low-level control over plot elements, making it highly customizable.

Versatility: It offers a wide range of plotting functions and styles, allowing users to create virtually any type of plot, including line plots, scatter plots, bar charts, and many more.

Steep learning curve: Matplotlib has a steep learning curve, especially for beginners, due to its extensive functionality and complex API. Users may need to invest time in learning the library's syntax, concepts, and best practices for creating effective plots.

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

Integration with pandas: Seaborn seamlessly integrates with pandas data frames, allowing users to directly pass data from pandas structures to seaborn plotting functions.

Attractive defaults: Seaborn comes with aesthetically pleasing default styles and color palettes, making it easy to create visually appealing plots without extensive customization.

High-level interface: Seaborn's high-level API simplifies the process of creating complex statistical plots by abstracting away the tedious details of data manipulation and plot configuration.

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 data frames for data visualization.
- Ideal for exploratory data analysis and quick visualization of relationships in data.

In summary, Matplolib provides low-level control and extensive customization options, making it suitable for creating highly customized plots. On the other hand, seaborn offers a high-level interface and attractive defaults, making it ideal for quickly creating complex statistical visualizations with minimal code.