VISUALIZATION

1.Introduction :

Data visualization plays an essential role in data science and analytics. It helps us to represent large and complex data in a clear, visual form such as graphs, charts, and plots.
With the help of visualization, it becomes easier to identify trends, patterns, and relationships in data that may not be visible from raw numbers alone.

Python offers many powerful libraries for data visualization, and two of the most commonly used are **Matplotlib** and **Seaborn**.
This documentation explains both libraries in detail, their uses, examples of different chart types, and a comparison of their features.

2. Matplotlib

Overview

Matplotlib is one of the most basic and popular visualization libraries in Python. John D. Hunter developed it in 2003 to offer MATLAB-like plotting options in Python. You can create line plots, bar graphs, scatter plots, histograms, pie charts, and more, all with full customization.

Key Features

- Supports 2D and some 3D plotting.
- Highly customizable: you have control over color, labels, axes, and layout.
- Integrates easily with NumPy and Pandas.
- Serves as the main library for Seaborn and other high-level tools.

Pyplot:

Pyplot is a module in the **Matplotlib** library that is used to create different types of graphs and plots in Python. It provides a simple way to visualize data by offering functions that let you draw lines, bars, scatter plots, and many other charts.

Some Graph Types and Examples in Matplotlib :

2.1. Line Graph

Shows trends or changes over time.

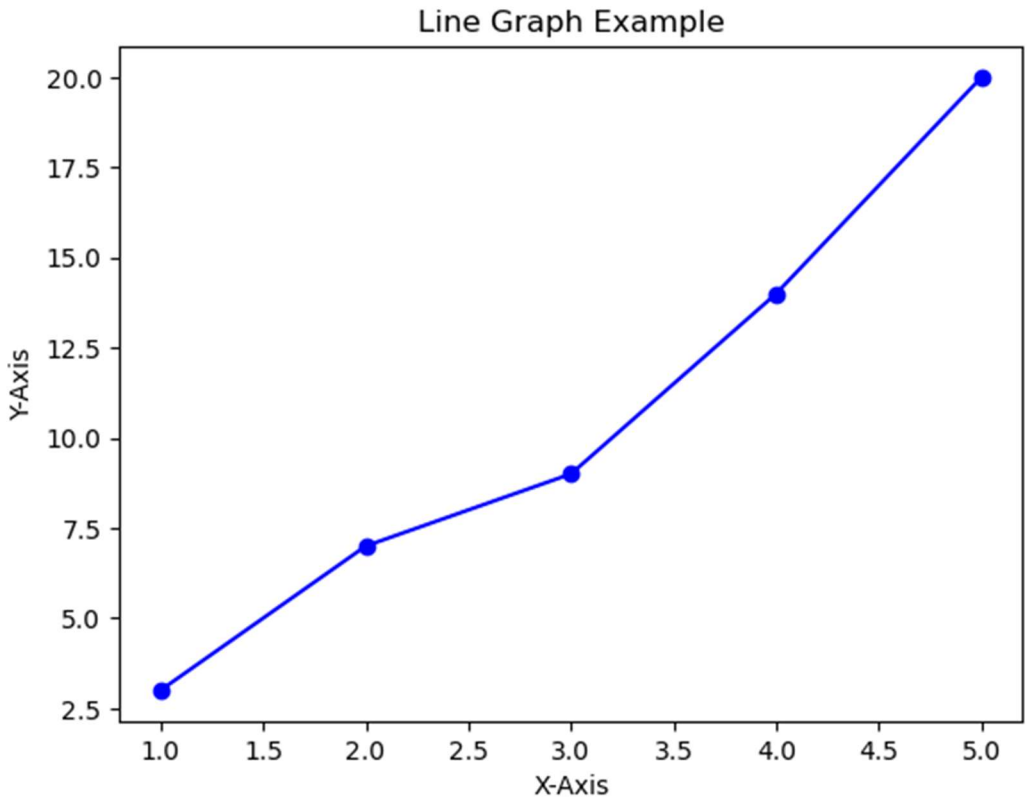
Code snippet:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [3, 7, 9, 14, 20]

plt.plot(x, y, color='blue', marker='o', linestyle='-')
plt.title("Line Graph Example")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

Output:



2.2. Bar Graph

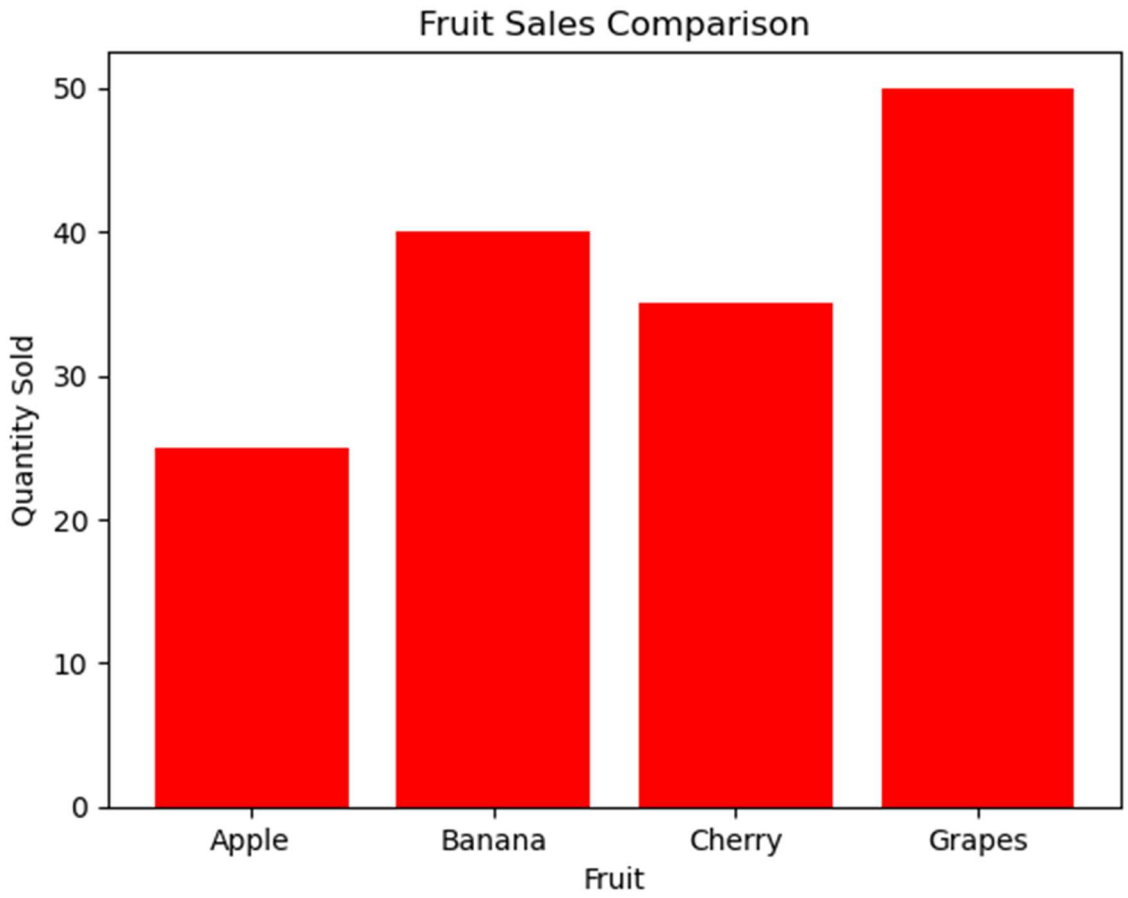
Compares quantities across categories.

Code snippet:

```
    categories = ['Apple', 'Banana', 'Cherry', 'Grapes']
values = [25, 40, 35, 50]

plt.bar(categories, values, color='red')
plt.title("Fruit Sales Comparison")
plt.xlabel("Fruit")
plt.ylabel("Quantity Sold")
plt.show()
```

Output:



2.3. Scatter Plot

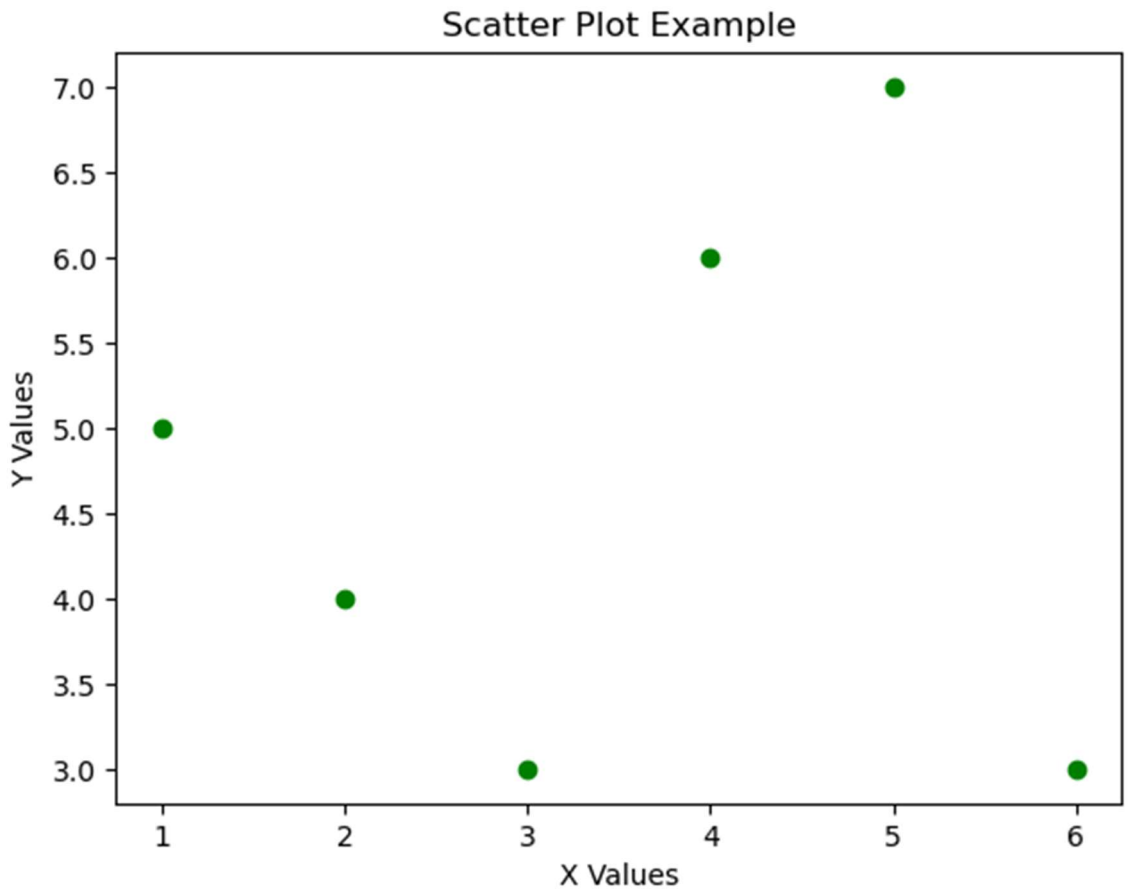
Shows the relationship between two numeric variables.

Code snippet:

```
x = [1, 2, 3, 4, 5, 6]
y = [5, 4, 3, 6, 7, 3]

plt.scatter(x, y, color='Green')
plt.title("Scatter Plot Example")
plt.xlabel("X Values")
plt.ylabel("Y Values")
plt.show()
```

Output:



2.4. Histogram

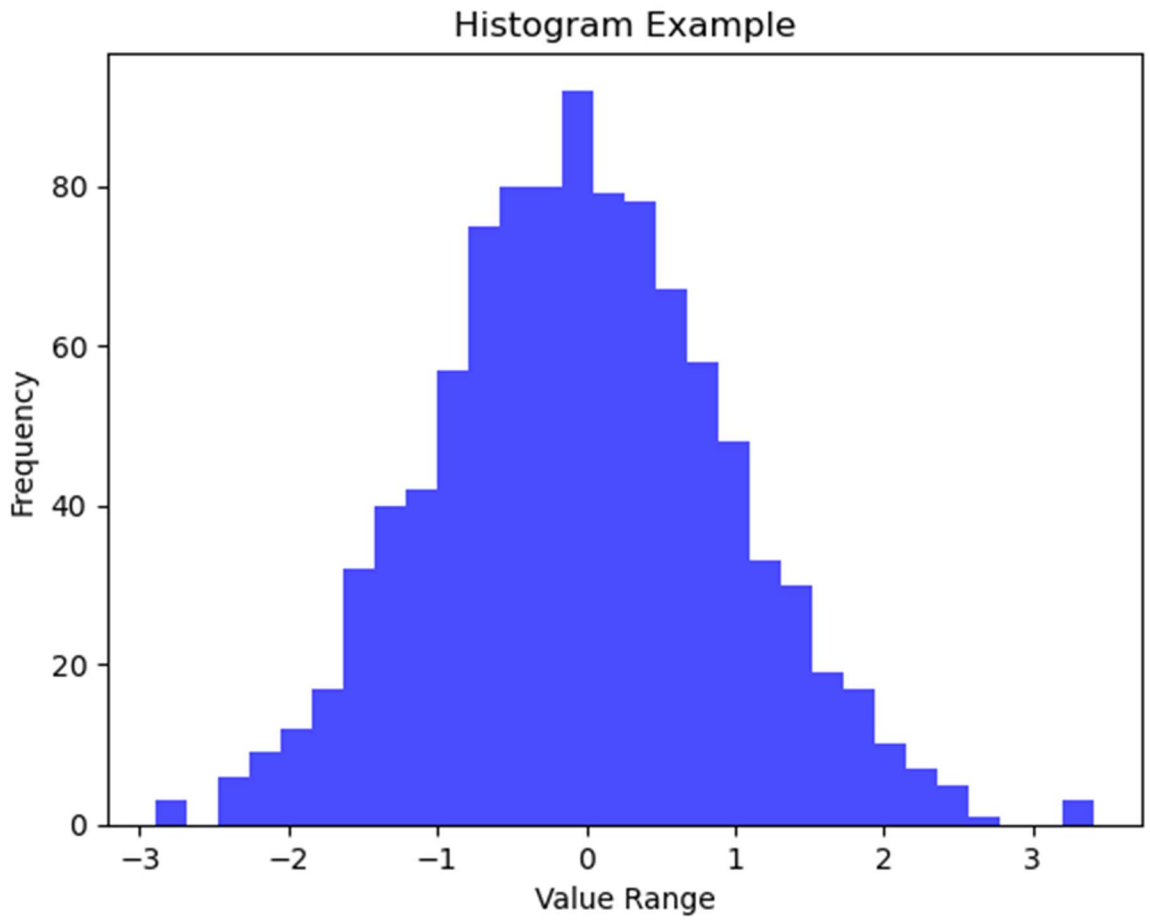
Displays data distribution and frequency.

Code snippet:

```
import numpy as np
data = np.random.randn(1000)

plt.hist(data, bins=30, color='blue', alpha=0.7)
plt.title("Histogram Example")
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.show()
```

Output:



2.5. Pie Chart

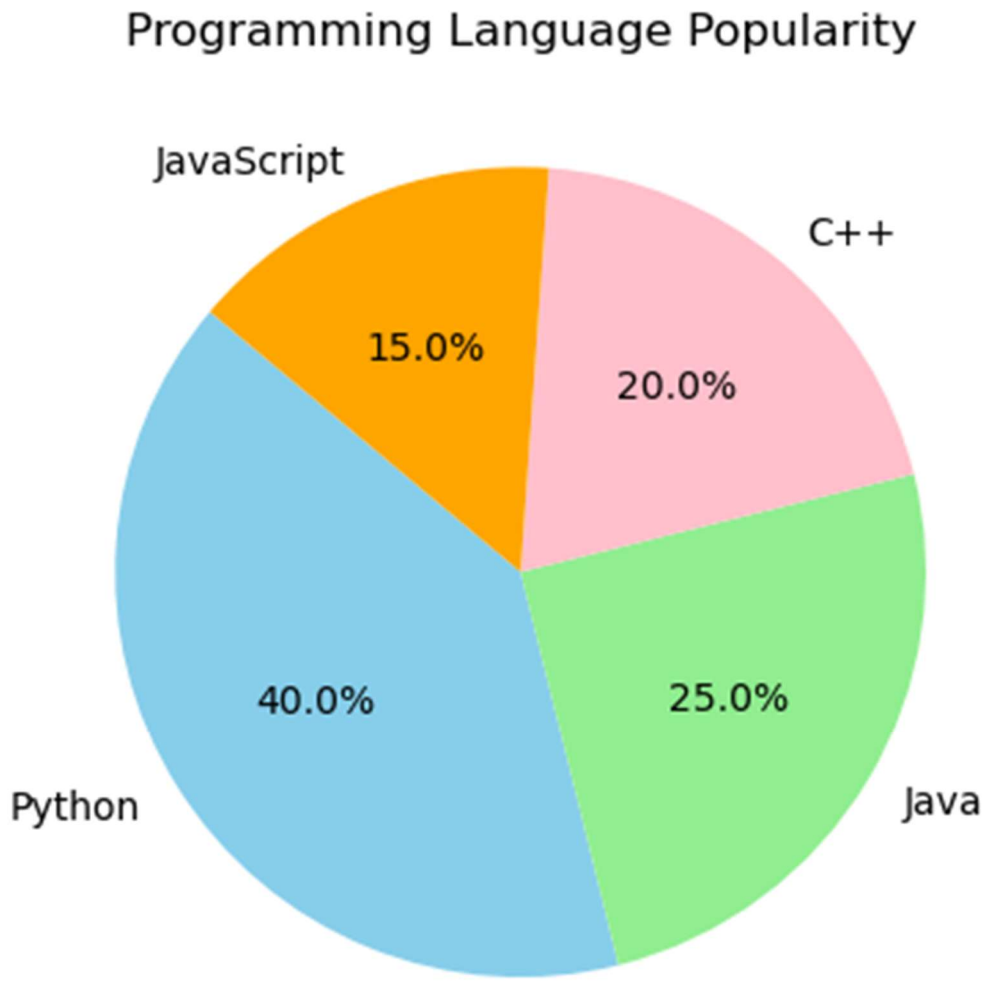
Represents proportions or percentage distribution.

Code snippet:

```
labels = ['Python', 'Java', 'C++', 'JavaScript']
sizes = [40, 25, 20, 15]
colors = ['skyblue', 'lightgreen', 'pink', 'orange']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140
plt.title("Programming Language Popularity")
plt.show()
```

Output:



2.6. Area Chart

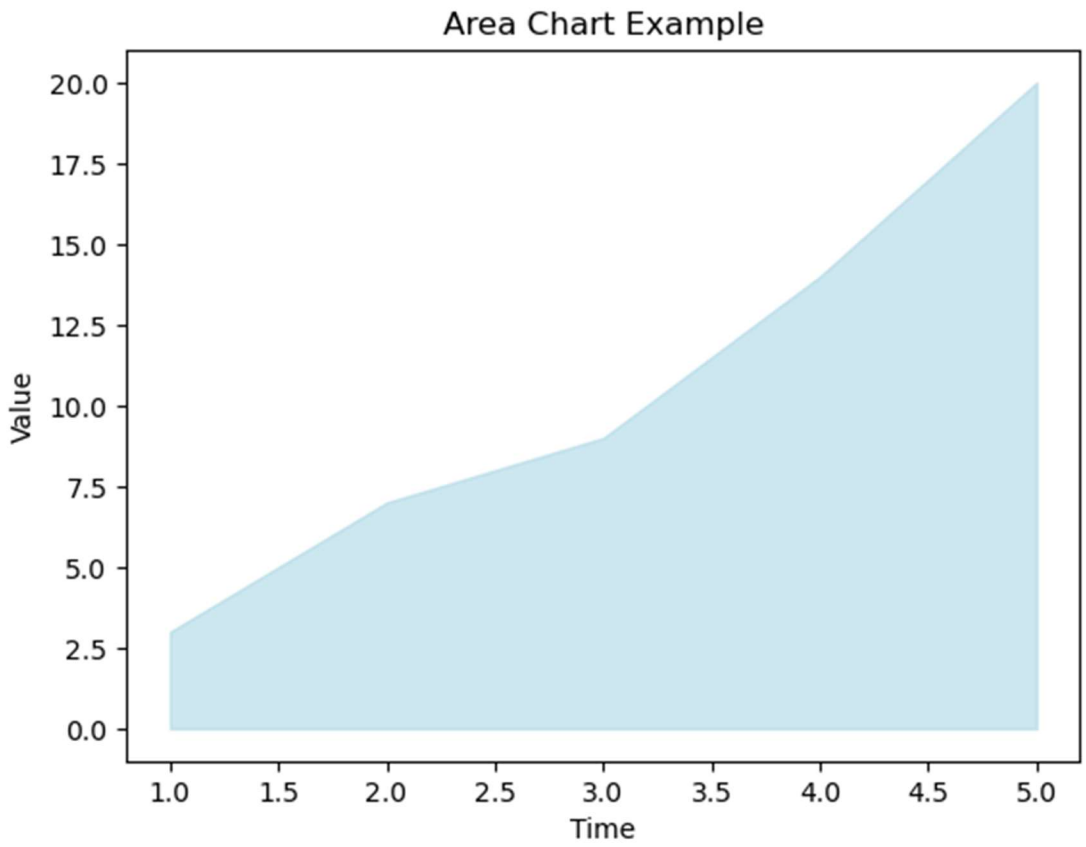
Used to display quantitative data over a time period.

Code snippet:

```
x = [1, 2, 3, 4, 5]
y = [3, 7, 9, 14, 20]

plt.fill_between(x, y, color='lightblue', alpha=0.6)
plt.title("Area Chart Example")
plt.xlabel("Time")
plt.ylabel("Value")
plt.show()
```

Output:



3. Seaborn

Overview

Seaborn is a high-level visualization library that is built on top of Matplotlib. In 2014, Michael Waskom created this visualization library to make statistical visualization easier and more visually appealing. Seaborn automatically applies nice themes and color palettes, so you can make graphs look clean and professional with minimal code

Key Features

- Built on Matplotlib (can be combined easily).
- Integrated with Pandas DataFrames.
- Provides advanced plots like heatmaps, KDE plots, pair plots, and regression plots.
- Automatically adds statistical summaries and styles.

To **import Seaborn** in Python:

```
import seaborn as sns
```

- Seaborn is great for visualizing relationships between different variables using plots like scatter plots, line plots, and pair plots.
- It helps you explore and understand how your data is distributed through histograms, box plots, and violin plots.
- You can easily visualize categorical data with bar plots, count plots, and swarm plots.
- Seaborn also makes it easy to style your charts with customized themes and color palettes, giving them a clean and professional look.

Some Graph Types and Examples in Seaborn :

3.1. Bar Plot

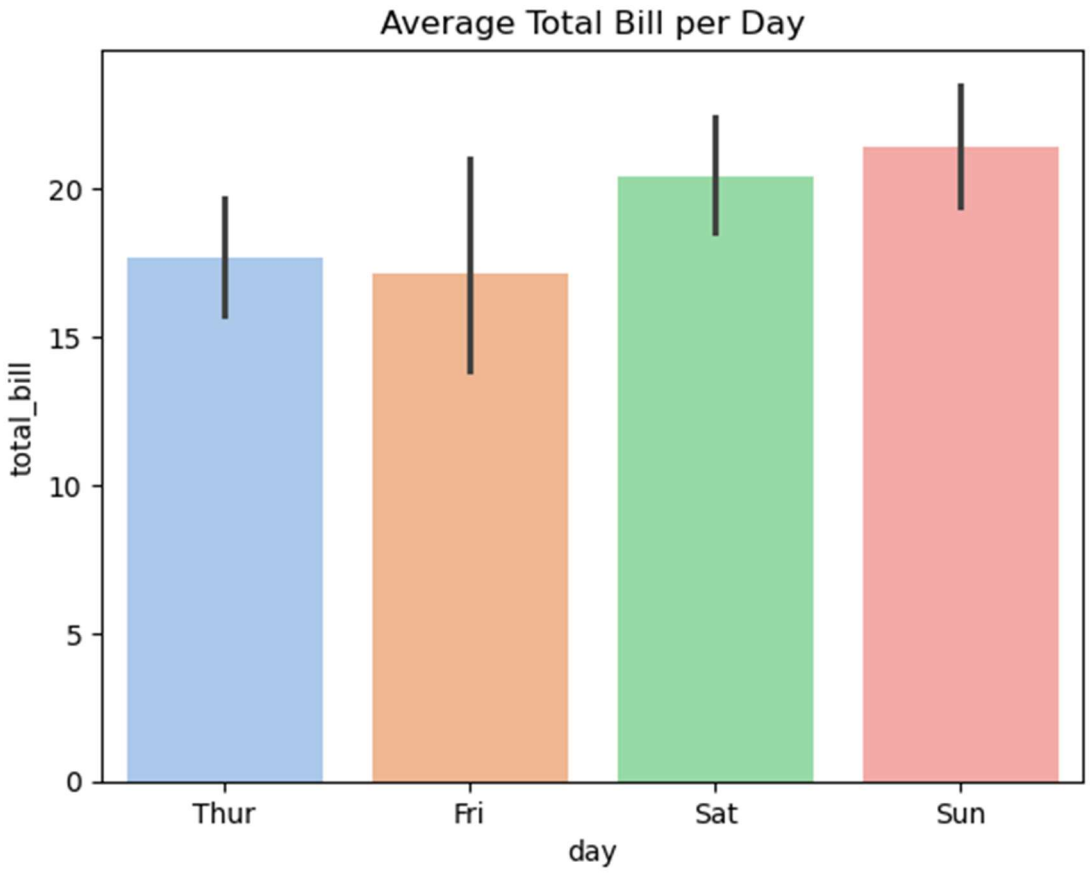
Shows mean values or category-based comparisons

Code snippet:

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

tips = sns.load_dataset("tips")
sns.barplot(x="day", y="total_bill", data=tips, palette="pastel", hue="day", legend=False)
plt.title("Average Total Bill per Day")
plt.show() # Assuming you're using plt.show() after all plot setup
```

Output:



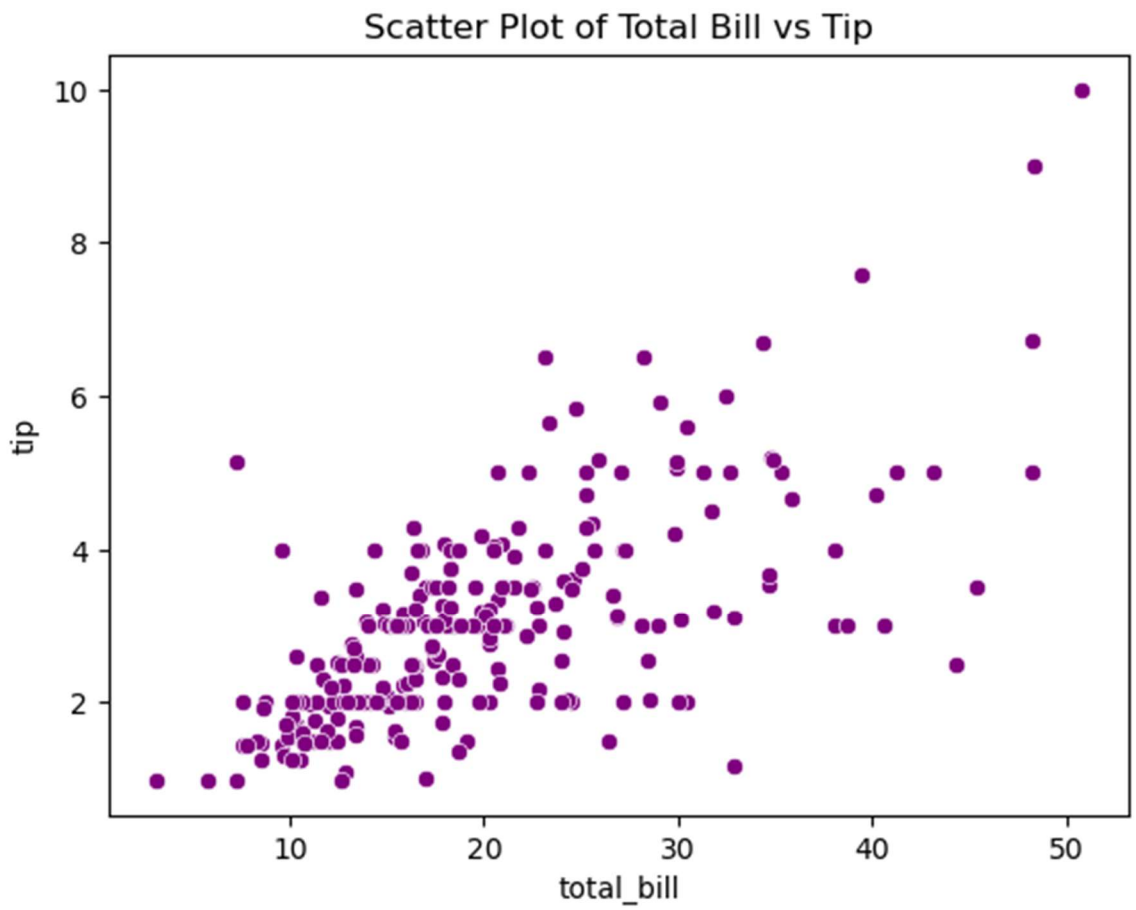
3.2. Scatter Plot

Displays relationships between two continuous variables.

Code snippet:

```
sns.scatterplot(x="total_bill", y="tip", data=tips, color="purple")
plt.title("Scatter Plot of Total Bill vs Tip")
plt.show()
```

Output:



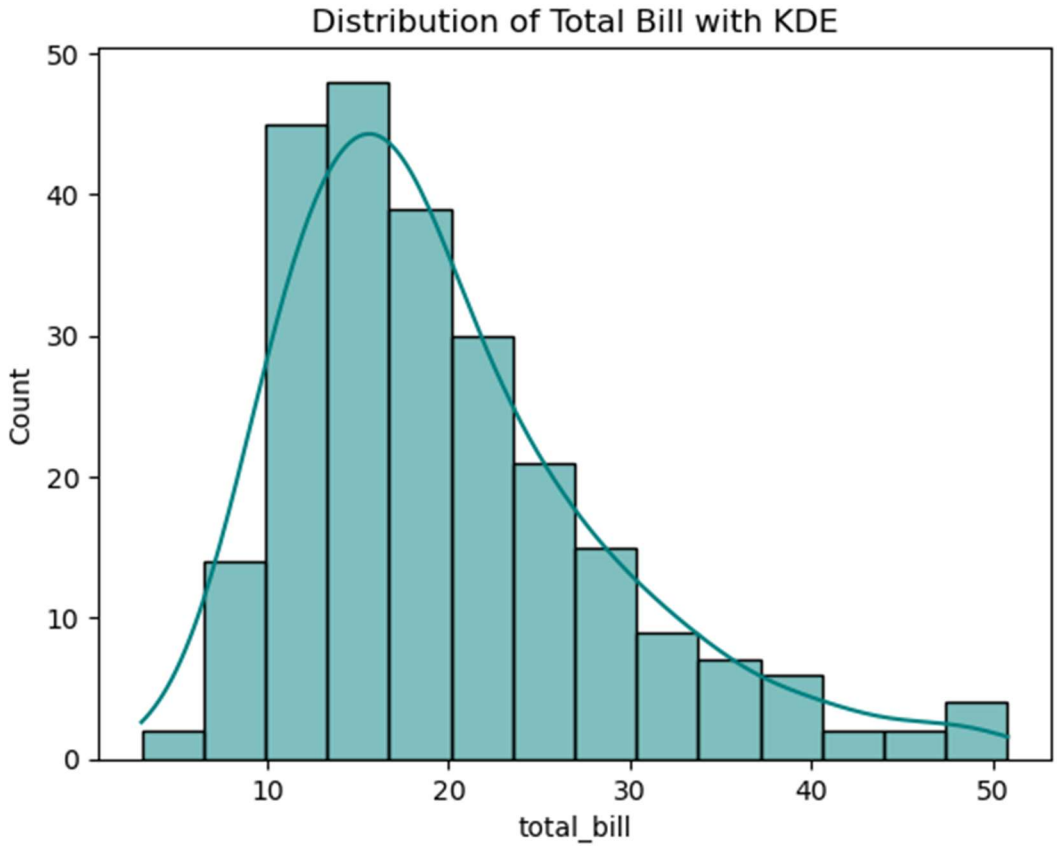
3.3. Histogram & KDE Plot

KDE (Kernel Density Estimate) shows a smooth distribution curve

Code snippet:

```
sns.histplot(tips['total_bill'], kde=True, color="teal")
plt.title("Distribution of Total Bill with KDE")
plt.show()
```

Output:



3.4. Box Plot

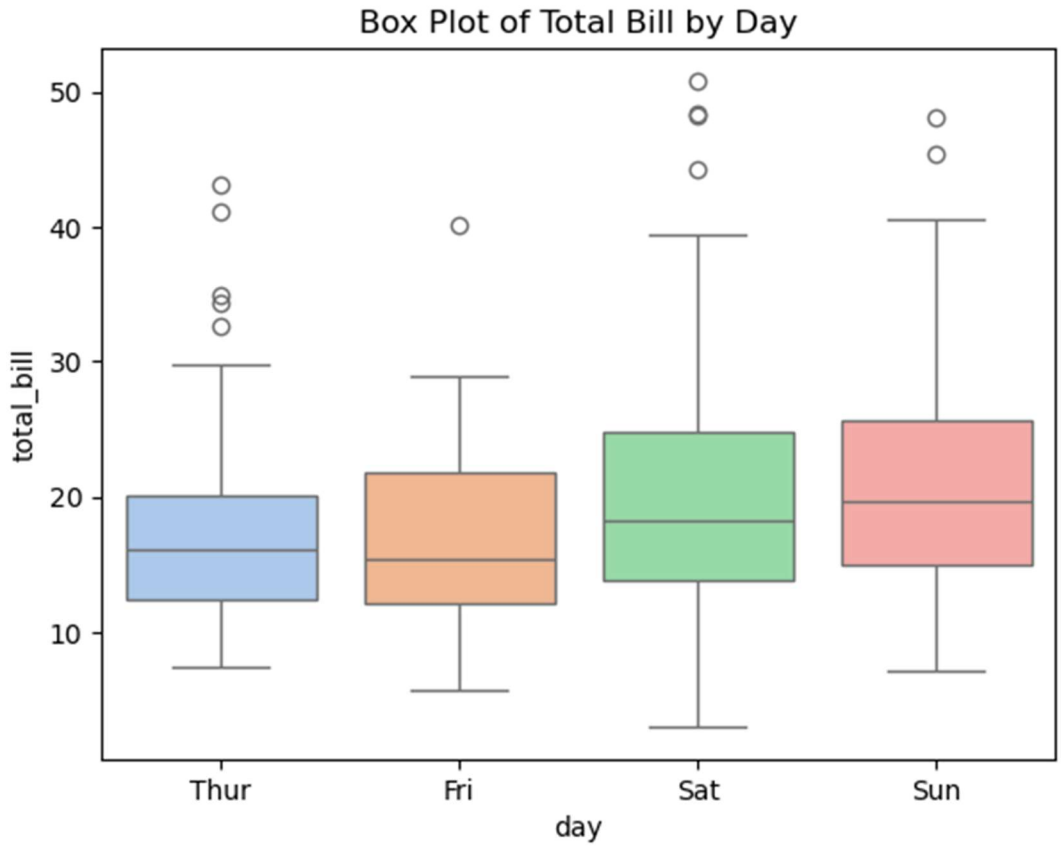
Used to visualize data spread and detect outliers.

Code snippet:

```
sns.boxplot(x="day", y="total_bill", data=tips, palette="pastel", hue="day", legend=False)

plt.title("Box Plot of Total Bill by Day")
plt.show()
```

Output:



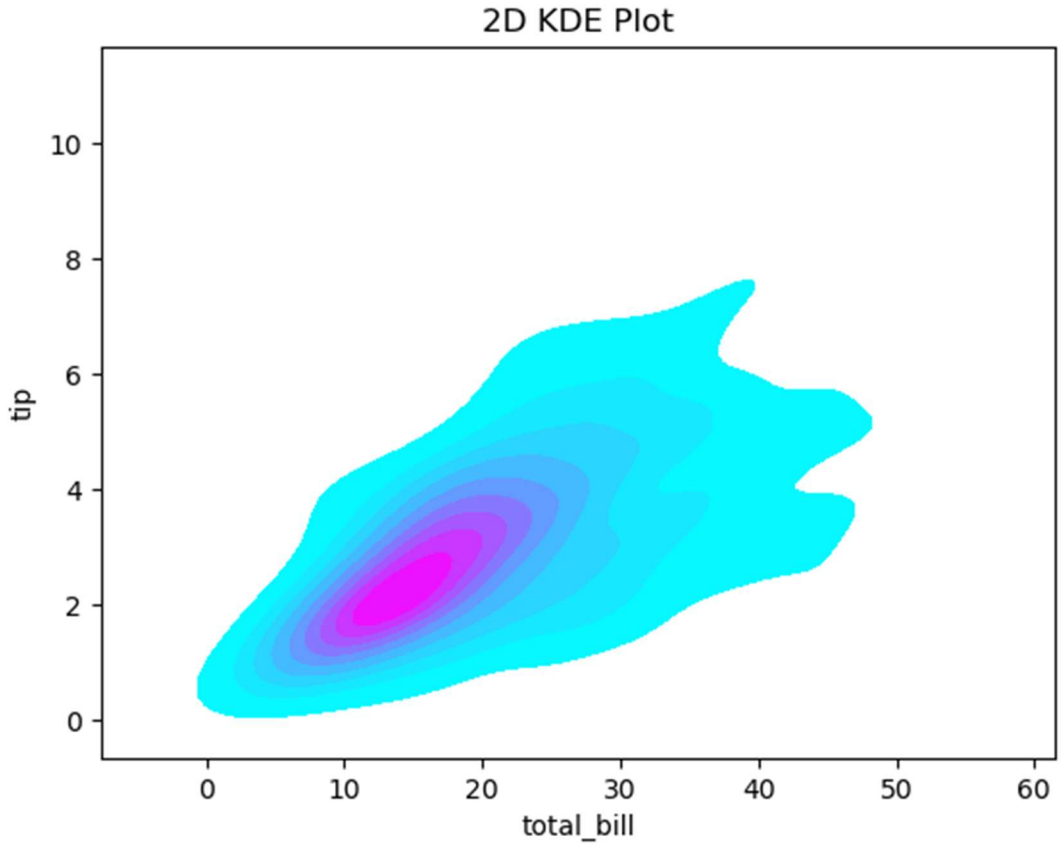
3.5. KDE Plot (2D Density)

Used to show probability density for two variables.

Code snippet:

```
sns.kdeplot(data=tips, x="total_bill", y="tip", fill=True, cmap="cool")
plt.title("2D KDE Plot")
plt.show()
```

Output:



3.6. Heatmap

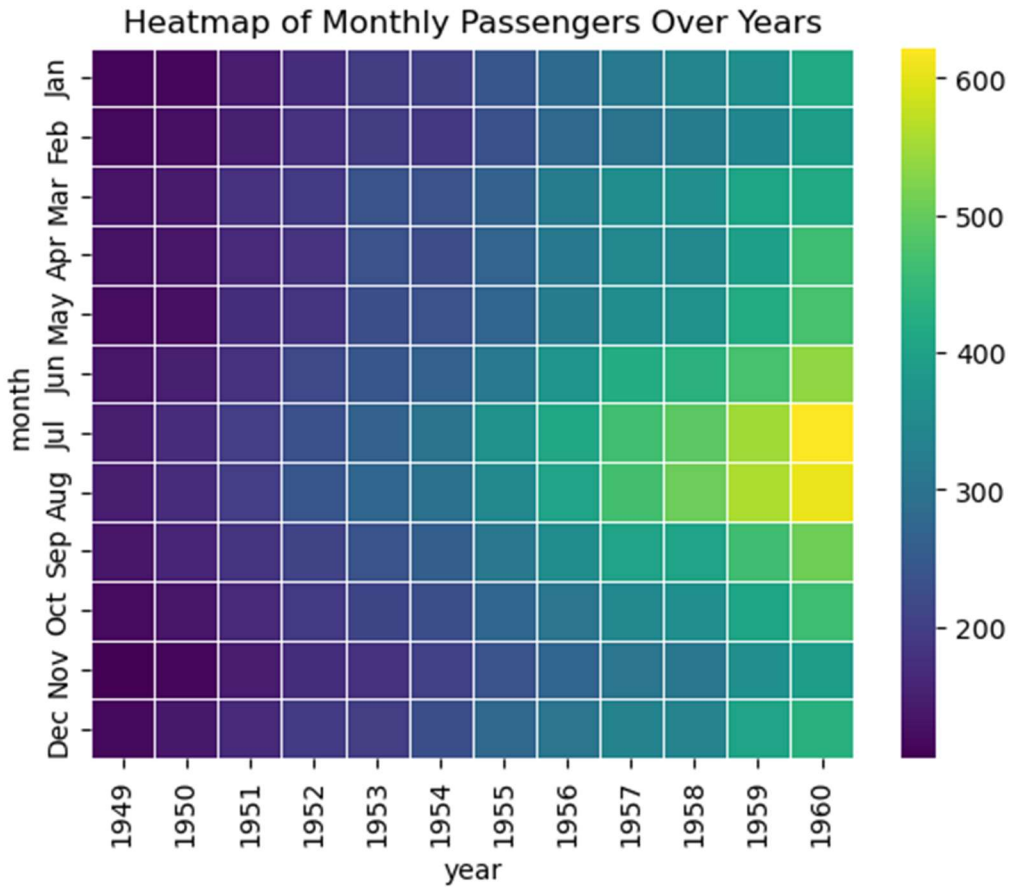
Visualizes matrix-like data using colors.

Code snippet:

```
flights = sns.load_dataset("flights")
pivot = flights.pivot_table(
    values="passengers",
    index="month",
    columns="year",
    observed=False
)

sns.heatmap(pivot, cmap="viridis", linewidths=0.5)
plt.title("Heatmap of Monthly Passengers Over Years")
plt.show()
```

Output:



4. Comparison Between Matplotlib and Seaborn

Feature	Matplotlib	Seaborn
Level	A low-level library where you build everything step by step.	A high-level library that does most of the work automatically.
Design Style	Simple and plain by default, but can be customized a lot.	Comes with attractive built-in styles and color themes.
Ease of Use	Requires more code and effort to make visually appealing plots.	Very easy to use — creates beautiful plots with minimal code.
Default Look	The plots look basic at first.	The plots look polished and professional right away.
Data Input	Works well with lists, arrays, or NumPy data.	Works directly with Pandas DataFrames, which makes it beginner-friendly.
Interactivity	Mostly static visuals.	Works smoothly in Jupyter Notebooks and supports limited interactivity.
Customization	You can control every detail — titles, colors, axes, labels, etc.	Has fewer options, but you can still use Matplotlib commands for extra control.
Best For	When you want complete control or publication-quality charts.	When you want to make quick, attractive graphs for analysis or reports.
Performance	Slightly faster and lighter.	A bit slower because of built-in styling and processing.
Developed By	John D. Hunter (2003).	Michael Waskom (2012).

5.Conclusion

Both **Matplotlib** and **Seaborn** play an important role in making data easier to understand through visuals.
Matplotlib offers detailed control over every element, while Seaborn makes the process simple and visually appealing.
Together, they help create clear, attractive, and meaningful graphs that make data analysis more effective and enjoyable.