Comprehensive Documentation Guide for Matplotlib

Library Overview

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Developed by John Hunter in 2003, Matplotlib provides a flexible way to generate a wide variety of plots and charts. It is particularly well-suited for creating publication-quality figures and is highly customizable.

Unique Features:

- Extensive plotting capabilities.
- Highly customizable plots.
- Ability to produce publication-quality figures.
- Wide range of output formats (PNG, PDF, SVG, etc.).

Typical Use Cases:

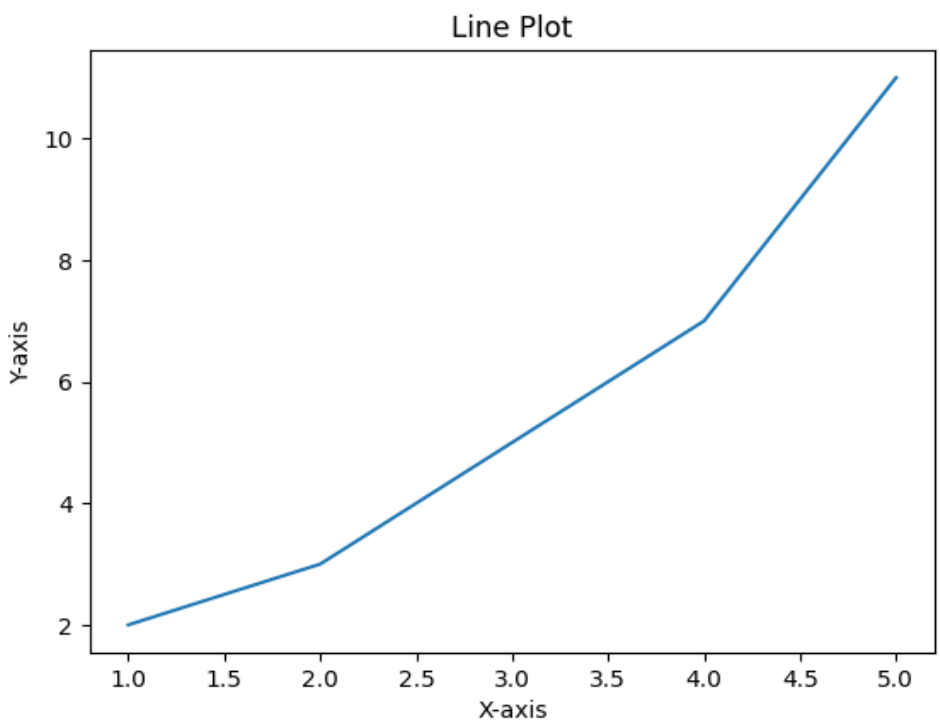
- Academic research and publication.
- Data analysis and exploration.
- Creating static plots for reports and presentations.

Graph Types

1. **Line Plot:** Displays data points connected by straight lines. Useful for showing trends over time.

   Code Example:

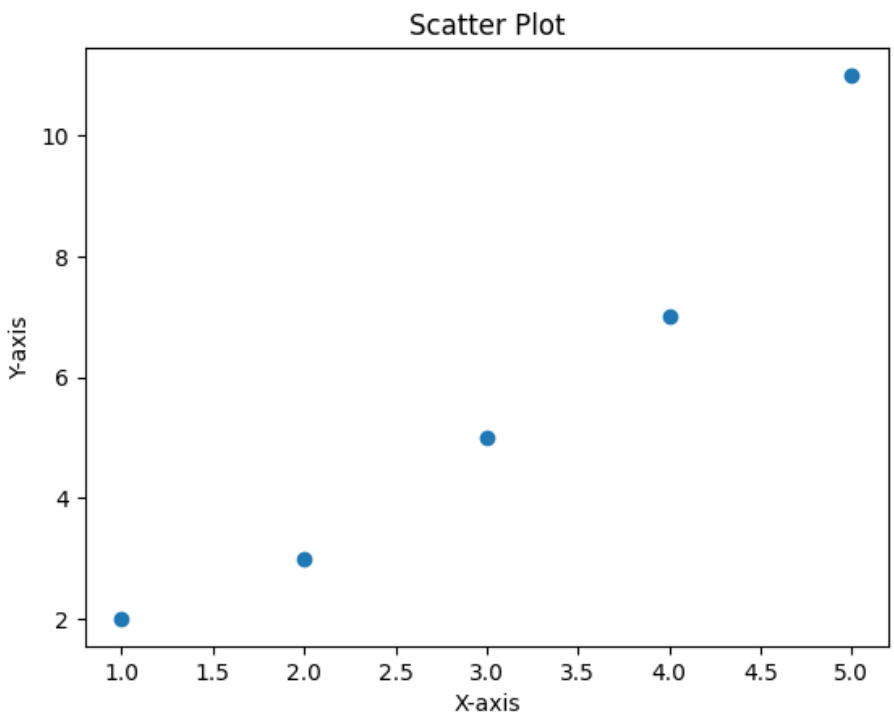
```
import matplotlib.pyplot as plt

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

plt.plot(x, y)
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



2. **Scatter Plot:** Uses dots to represent the values obtained for two different variables. Ideal for identifying relationships between variables.

   Code Example:

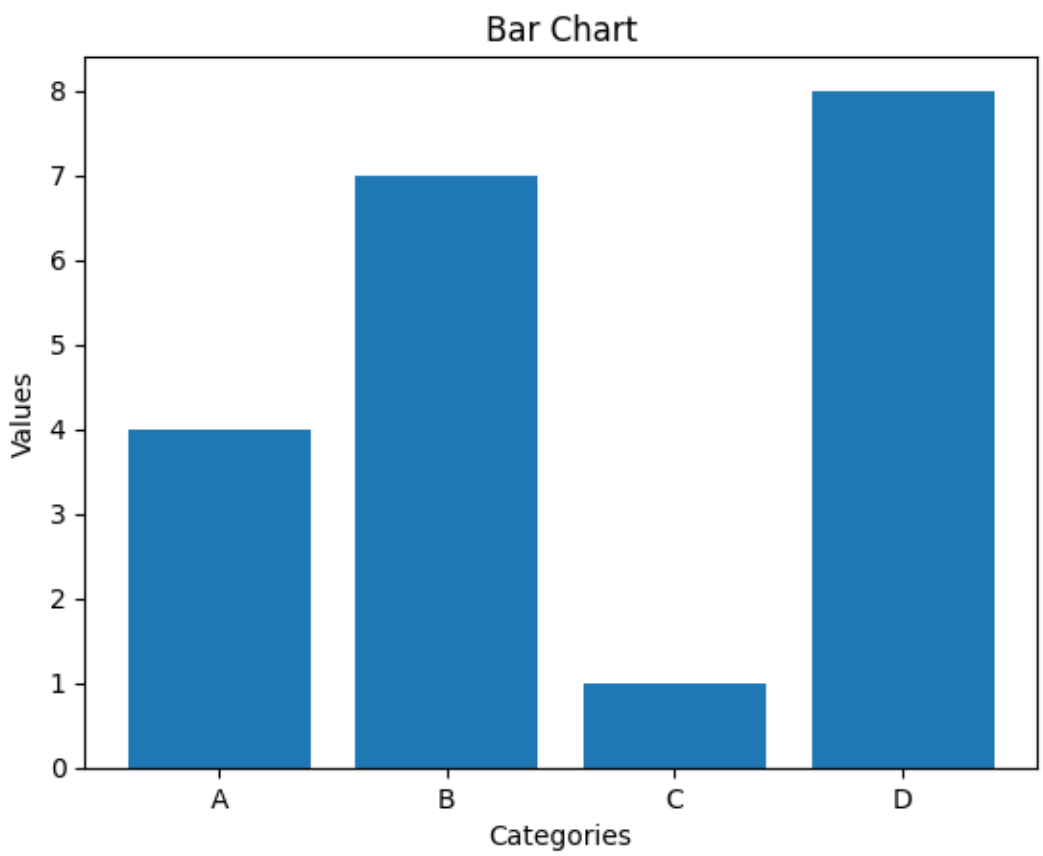
```python
import matplotlib.pyplot as plt

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

plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```



3. **Bar Chart:** Uses rectangular bars to represent data values. Good for comparing discrete categories.

   Code Example:

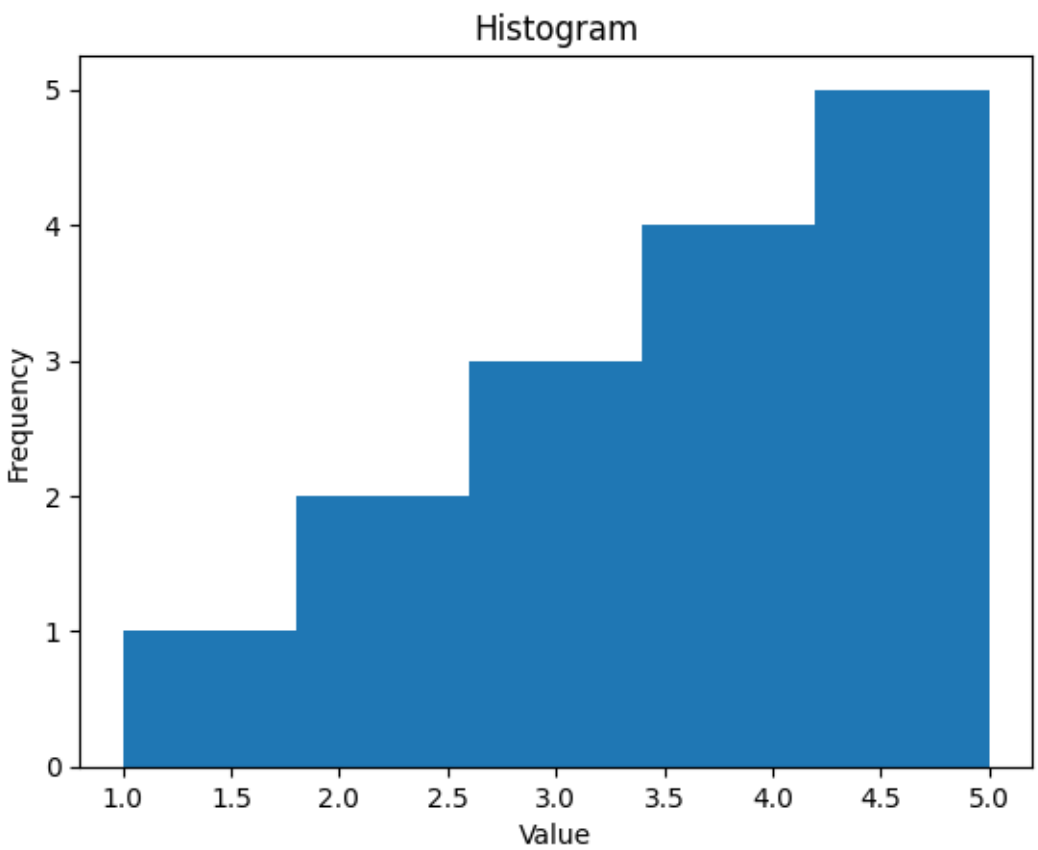
```python
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D']
values = [4, 7, 1, 8]

plt.bar(categories, values)
plt.title("Bar Chart")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```



4. **Histogram**: Shows the distribution of a dataset. Useful for understanding the distribution of numerical data.

   Code Example:

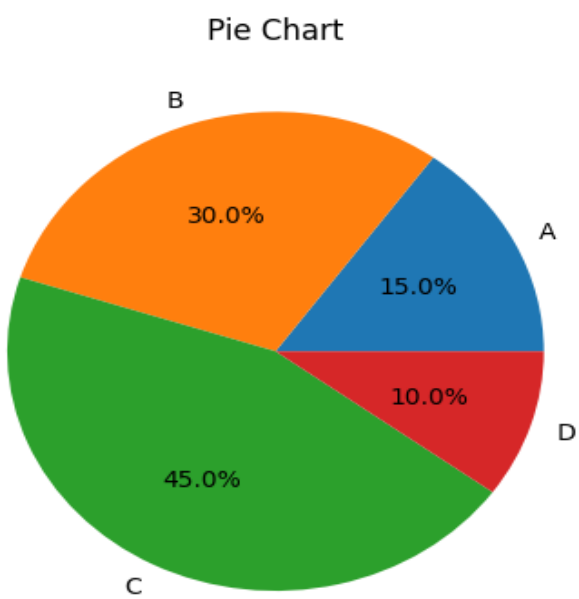
```
import matplotlib.pyplot as plt

data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

plt.hist(data, bins=5)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```



5. **Pie Chart:** Circular chart divided into sectors to illustrate numerical proportions.

   Code Example:

```
import matplotlib.pyplot as plt

sizes = [15, 30, 45, 10]
labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title("Pie Chart")
plt.show()
```



Comparison

Strengths	Weaknesses
Highly customizable	Steeper learning curve for beginners
Extensive documentation and user community	Requires more code for simple plots.
Supports a wide range of output formats	Limited default aesthetics.
Suitable for creating complex and publication-quality figures	

Resources

Matplotlib Quick Start Guide

Comprehensive Documentation Guide for Seaborn

Library Overview

Seaborn

Seaborn is a Python data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. Developed by Michael Waskom, Seaborn simplifies complex visualizations with less code and integrates well with Pandas data structures.

Unique Features:

- Beautiful default styles and color palettes.
- Simplified syntax for complex visualizations.
- Integrated with Pandas data structures.
- Specialized plots for statistical analysis.

Typical Use Cases:

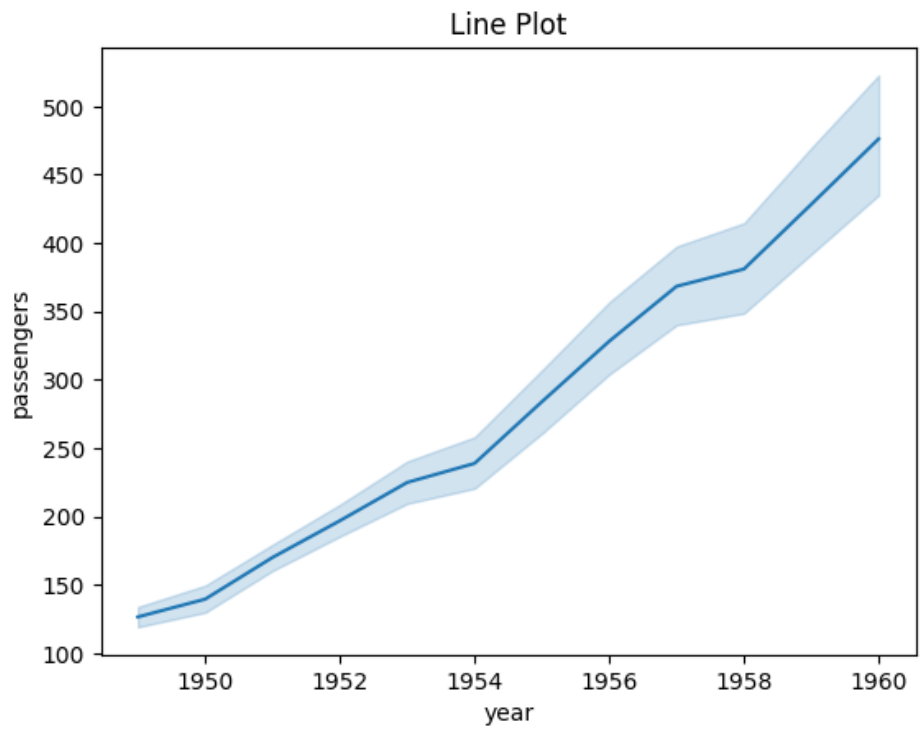
- Exploratory data analysis.
- Statistical data visualization.
- Creating aesthetically pleasing plots with minimal effort.

Graph Types

1. **Line Plot:** Similar to Matplotlib's line plot but with additional statistical features.

   Code Example:

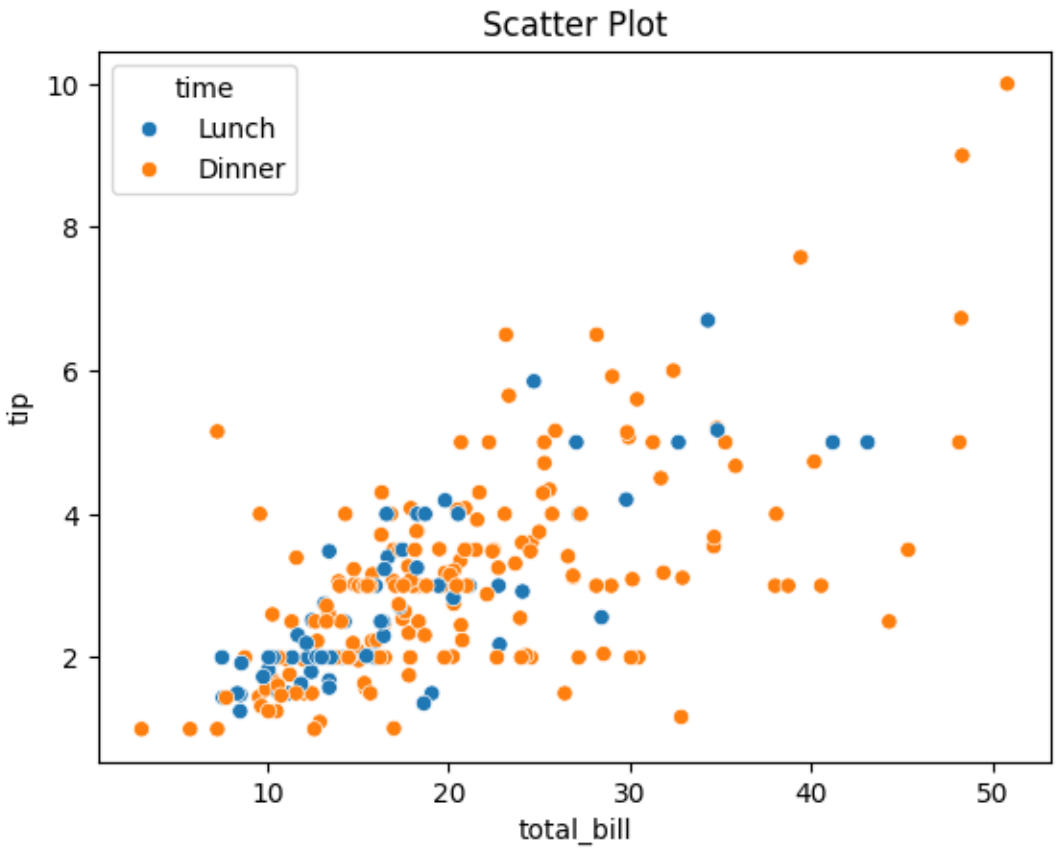
```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("flights")
sns.lineplot(x="year", y="passengers", data=data)
plt.title("Line Plot")
plt.show()
```



2. **Scatter Plot:** Enhanced scatter plot with additional features like regression lines.

   Code Example:

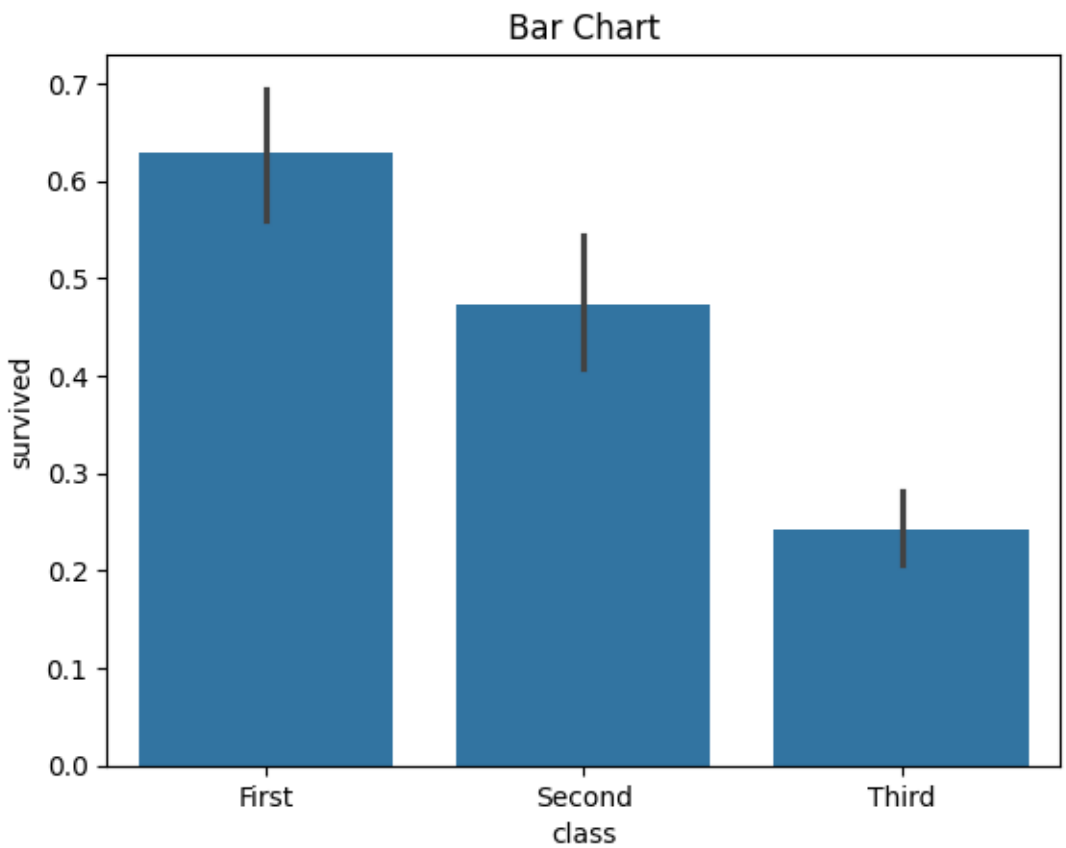
```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("tips")
sns.scatterplot(x="total_bill", y="tip", hue="time", data=data)
plt.title("Scatter Plot")
plt.show()
```



3. **Bar Chart:** Bar plot with additional statistical aggregations.

Code Example:

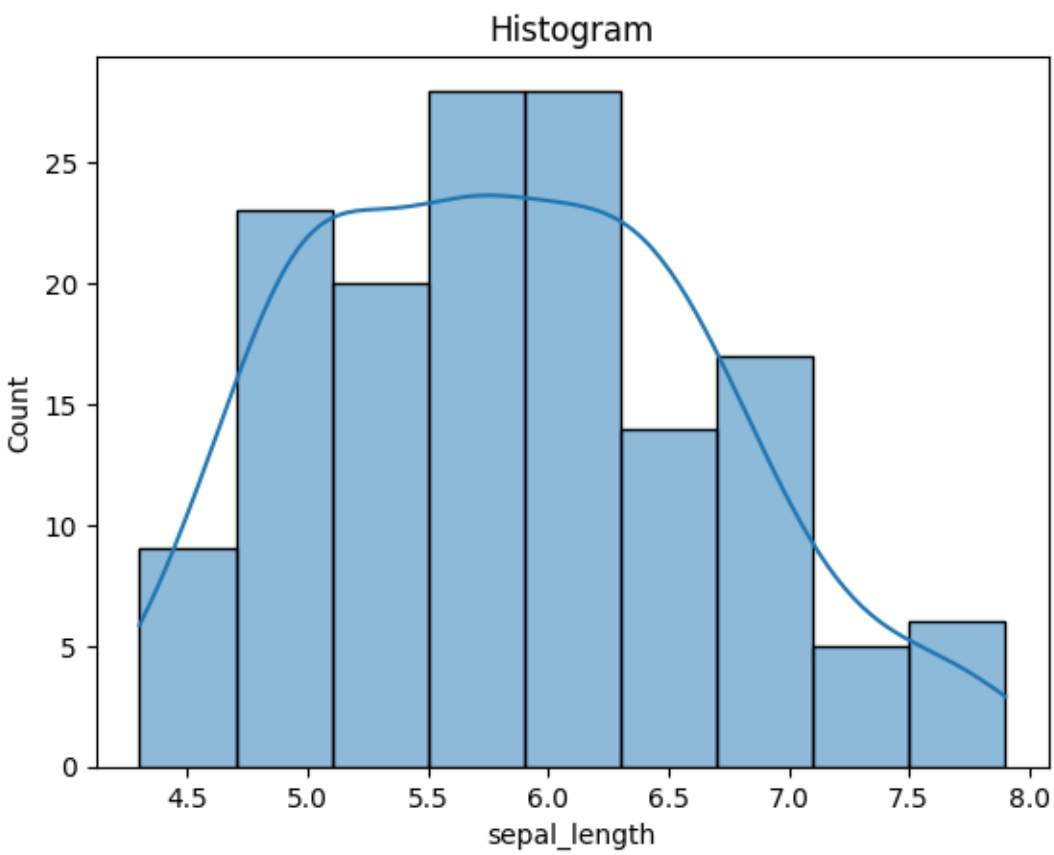
```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("titanic")
sns.barplot(x="class", y="survived", data=data)
plt.title("Bar Chart")
plt.show()
```



4. **Histogram:** Enhanced histogram with kernel density estimate.

Code Example:

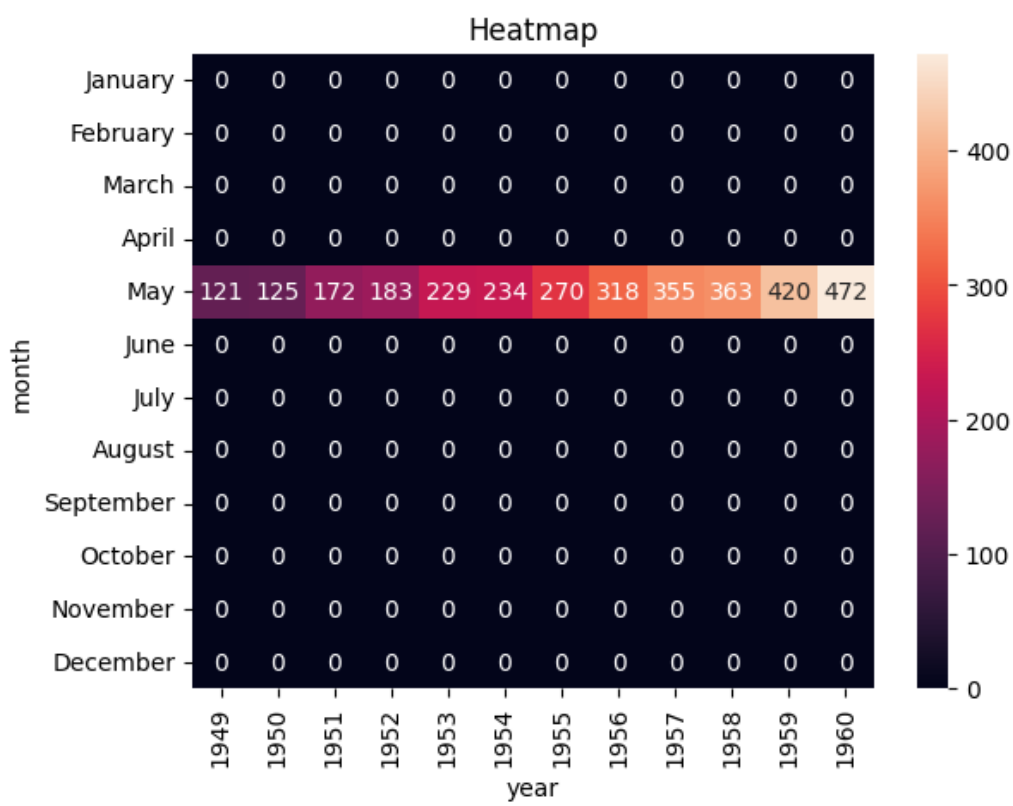
```
import seaborn as sns
import matplotlib.pyplot as plt

data = sns.load_dataset("iris")
sns.histplot(data["sepal_length"], kde=True)
plt.title("Histogram")
plt.show()
```



5. **Heatmap**: Displays matrix data as a color-encoded grid. Useful for correlation matrices and cluster analysis.

   Code Example:

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

data = sns.load_dataset("flights")

data['month'] = pd.Categorical(data['month'], categories=['January', 'February', 'March', 'April', 'May', 'June',
    'July', 'August', 'September', 'October', 'November', 'December'], ordered=True)

duplicates = data.duplicated(subset=["year", "month"], keep=False)
if duplicates.any():
    print("Duplicates found, aggregating duplicate values.")
    data = data.groupby(["year", "month"], as_index=False).sum()

try:
    data_pivot = data.pivot(index="month", columns="year", values="passengers")
except Exception as e:
    print(f"Error in pivoting the data: {e}")
    print("Data types of columns:")
    print(data.dtypes)
    print("Unique values in 'month' column:")
    print(data['month'].unique())
    raise

sns.heatmap(data_pivot, annot=True, fmt="d")

plt.title("Heatmap")

plt.show()
```



Comparison

STRENGTHS	WEAKNESSES
Beautiful default aesthetics.	Less flexible than Matplotlib for highly customized plots.
Simplifies the process of creating complex visualizations.	Limited control over low-level plot details.
Integrates well with Pandas data structures.	Depends on Matplotlib for rendering
Great for statistical data visualization.	

Resources

Seaborn Introduction