Matplotlib

Prof. Pranay Kumar Saha

January 30, 2025

What is Matplotlib?

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- Widely used for data visualization and exploratory analysis.

Basic Line Plot

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 8, 7]

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

Basic Line Plot

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 8, 7]

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

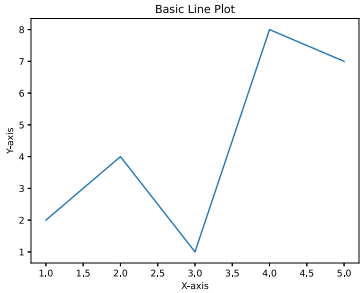
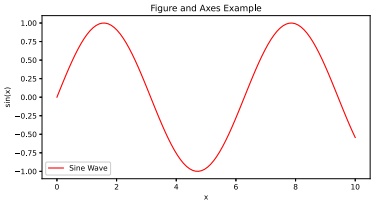


Figure and Axes Objects

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

x = np.linspace(0, 10, 100)
y = np.sin(x)

fig, ax = plt.subplots(figsize=(8,4))
ax.plot(x, y, color='red', label='Sine Wave')
ax.set_title("Figure and Axes Example")
ax.set_xlabel("x")
ax.set_ylabel("sin(x)")
ax.legend()
plt.show()
```

Figure and Axes Objects

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

x = np.linspace(0, 10, 100)
y = np.sin(x)

fig, ax = plt.subplots(figsize=(8,4))
ax.plot(x, y, color='red', label='Sine Wave
ax.set_title("Figure and Axes Example")
ax.set_xlabel("x")
ax.set_ylabel("sin(x)")
ax.legend()
plt.show()
```

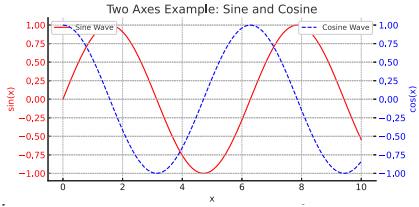


Multiple Axes (shared x-axis)

```
x = np.linspace(0, 10, 100)
y1 = np.sin(x)  # Sine wave
y2 = np.cos(x)  # Cosine wave
fig, ax1 = plt.subplots(figsize=(8, 4))
ax1.plot(x, y1, color='red', label='Sine Wave')
ax1.set_xlabel("x")
ax1.set_ylabel("sin(x)", color='red')
ax1.tick_params(axis='y', labelcolor='red')
# Create a second y-axis sharing the same x-axis
ax2 = ax1.twinx()
ax2.plot(x, y2, color='blue', linestyle='dashed', label='Cosine Wave')
ax2.set_ylabel("cos(x)", color='blue')
ax2.tick_params(axis='y', labelcolor='blue')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
```

Multiple Axes (shared x-axis)

```
x = np.linspace(0, 10, 100)
y1 = np.sin(x)  # Sine wave
y2 = np.cos(x)  # Cosine wave
fig, ax1 = plt.subplots(figsize=(8, 4))
ax1.plot(x, y1, color='red', label='Sine Wave')
ax1.set_xlabel("x")
ax1.set_ylabel("sin(x)", color='red')
ax1.tick_params(axis='y', labelcolor='red')
# Create a second y-axis sharing the same x-axis
ax2 = ax1.twinx()
ax2.plot(x, y2, color='blue', linestyle='dashed'
ax2.set_ylabel("cos(x)", color='blue')
ax2.tick_params(axis='y', labelcolor='blue')
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
```

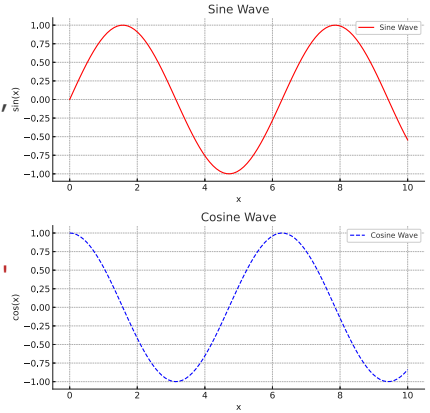


Multiple Axes (Separate x-axis)

```
x = np.linspace(0, 10, 100)
y1 = np.sin(x)  # Sine wave
y2 = np.cos(x)  # Cosine wave
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8, 8))
ax1.plot(x, y1, color='red', label='Sine Wave')
ax1.set_title("Sine Wave")
ax1.set_xlabel("x")
ax1.set_ylabel("sin(x)")
ax1.legend()
ax2.plot(x, y2, color='blue', linestyle='dashed', label='Cosine Wave')
ax2.set_title("Cosine Wave")
ax2.set_xlabel("x")
ax2.set_ylabel("cos(x)")
ax2.legend()
plt.tight_layout()
```

Multiple Axes (Separate x-axis)

```
x = np.linspace(0, 10, 100)
y1 = np.sin(x)  # Sine wave
y2 = np.cos(x)  # Cosine wave
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(8,
ax1.plot(x, y1, color='red', label='Sine Wave')
ax1.set_title("Sine Wave")
ax1.set_xlabel("x")
ax1.set_ylabel("sin(x)")
ax1.legend()
ax2.plot(x, y2, color='blue', linestyle='dashed'
ax2.set_title("Cosine Wave")
ax2.set_xlabel("x")
ax2.set_ylabel("cos(x)")
ax2.legend()
plt.tight_layout()
```

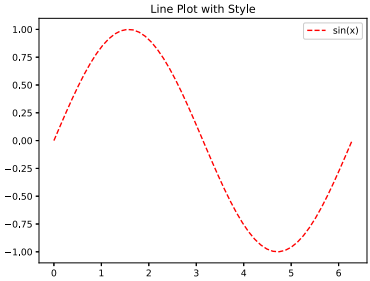


Line Plot Styles

```
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)

plt.plot(x, y, 'r--', label='sin(x)')
# red dashed line
plt.title("Line Plot with Style")
plt.legend()
plt.show()
```

Line Plot Styles

```
x = np.linspace(0, 2*np.pi, 50)
y = np.sin(x)

plt.plot(x, y, 'r--', label='sin(x)')
# red dashed line
plt.title("Line Plot with Style")
plt.legend()
plt.show()
```

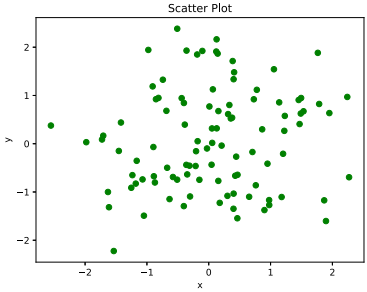


Scatter Plot

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

np.random.seed(0)
x = np.random.randn(100)
y = np.random.randn(100)

plt.scatter(x, y, c='green', alpha=0.5)
plt.title("Scatter Plot")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Scatter Plot

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

np.random.seed(0)
x = np.random.randn(100)
y = np.random.randn(100)

plt.scatter(x, y, c='green', alpha=0.5)
plt.title("Scatter Plot")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

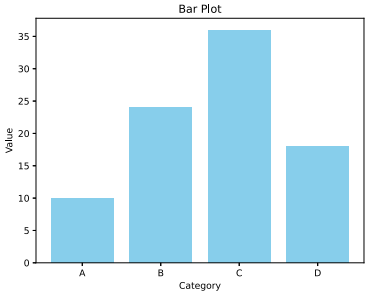


Bar Plot

```
categories = ['A', 'B', 'C', 'D']
values = [10, 24, 36, 18]

plt.bar(categories, values, color='skyblue')
plt.title("Bar Plot")
plt.xlabel("Category")
plt.ylabel("Value")
plt.show()
```

Bar Plot

```
categories = ['A', 'B', 'C', 'D']
values = [10, 24, 36, 18]

plt.bar(categories, values, color='skyblue'
plt.title("Bar Plot")
plt.xlabel("Category")
plt.ylabel("Value")
plt.show()
```

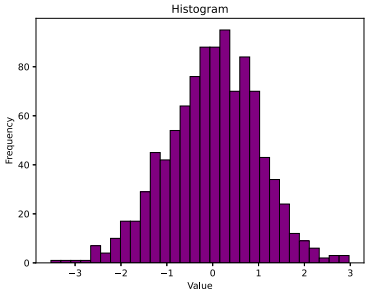


Histogram

```
data = np.random.normal(0, 1, 1000)
plt.hist(data, bins=30, color='purple',
            edgecolor='black', alpha=0.7)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

Histogram

```
data = np.random.normal(0, 1, 1000)
plt.hist(data, bins=30, color='purple',
            edgecolor='black', alpha=0.7)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

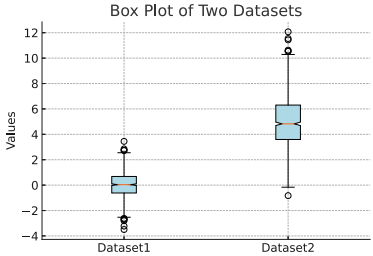


Box Plot

```
dataset1 = np.random.normal(0, 1, 1000)
dataset2 = np.random.normal(5, 2, 1000)

plt.boxplot([dataset1, dataset2],
        labels=["Dataset1", "Dataset2"])

plt.title("Box Plot")
plt.show()
```

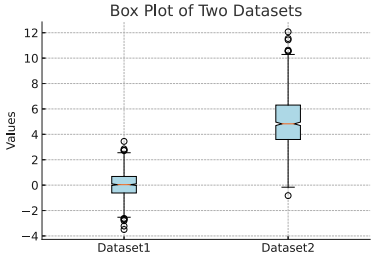
Box Plot

```
dataset1 = np.random.normal(0, 1, 1000)
dataset2 = np.random.normal(5, 2, 1000)

plt.boxplot([dataset1, dataset2],
        labels=["Dataset1", "Dataset2"])

plt.title("Box Plot")
plt.show()
```



Box Plot

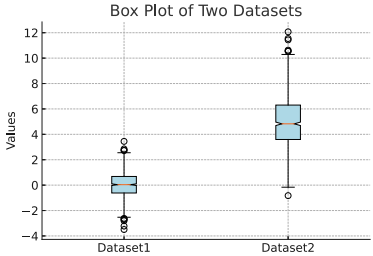
- **Median (Q2)** –Thick line inside the box.
- **Interquartile Range (IQR)** –The box height from Q1 to Q3.
- **Whiskers** –Extend up to 1.5 times the IQR.
- **Outliers** –Points outside the whiskers (plotted as dots).



Box Plot

Dataset1 (left box):

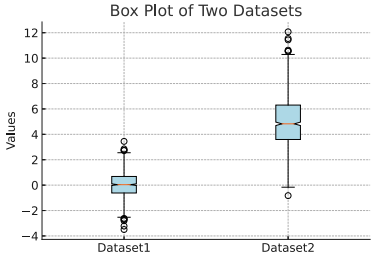
- **Median (Q2)** should be around 0.
- The **IQR** (Q1 to Q3) should be around $[-1, 1]$.
- **Whiskers** extend roughly $[-3, 3]$.
- Any extreme values beyond this range appear as **outliers** (dots).



Box Plot

Dataset2 (right box):

- **Median (Q2)** should be around 5.
- The **IQR** (Q1 to Q3) should be around $[3, 7]$.
- **Whiskers** extend roughly $[1, 9]$.
- Some points beyond this range appear as **outliers**.



Titles, Labels, Legends

- `plt.title("Title")` or `ax.set_title("Title")`
- `plt.xlabel("X Label")` or `ax.set_xlabel("X Label")`
- `plt.ylabel("Y Label")` or `ax.set_ylabel("Y Label")`
- `plt.legend()` or `ax.legend()`

Customization Example

```
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

plt.figure(figsize=(8,5))
plt.plot(x, y1, label='sin(x)', color='blue', linestyle='--')
plt.plot(x, y2, label='cos(x)', color='red', marker='o', markevery=10)
plt.title("Sine and Cosine")
plt.xlabel("x")
plt.ylabel("Value")
plt.grid(True)
plt.legend()
plt.show()
```

Creating Subplots

```
x = np.linspace(0, 2*np.pi, 100)
y_sin = np.sin(x)
y_cos = np.cos(x)
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(6,8))
# First subplot
axes[0].plot(x, y_sin, 'b-')
axes[0].set_title("Sine")
axes[0].set_xlabel("x")
axes[0].set_ylabel("sin(x)")
# Second subplot
axes[1].plot(x, y_cos, 'r--')
axes[1].set_title("Cosine")
axes[1].set_xlabel("x")
axes[1].set_ylabel("cos(x)")
plt.tight_layout()
plt.show()
```

Saving Figures

```
plt.savefig("myplot.png", dpi=300)
```

- Saves the current figure to a file named `myplot.png`
- `dpi=300` sets the resolution (dots per inch)
- With the object-oriented approach:

```
fig.savefig("myplot.pdf")
```

Additional Features

- **Styles**: Use `plt.style.use('ggplot')` or any style in `plt.style.available`
- **3D Plots**: Use `mpl_toolkits.mplot3d`
- **Animations**: See `matplotlib.animation`
- **Interactive Widgets**: In Jupyter, try `ipympl` or `ipywidgets`

Summary

- Installation and Setup
- Basic Plotting (line, scatter, bar, histogram, box)
- Figure and Axes
- Customization (titles, labels, legends, styles)
- Subplots
- Saving Figures
- Advanced/Optional Features (3D, animations, etc.)

Check the official Matplotlib documentation for more:

`https://matplotlib.org/`

Matplotlib

Thank You for Listening!