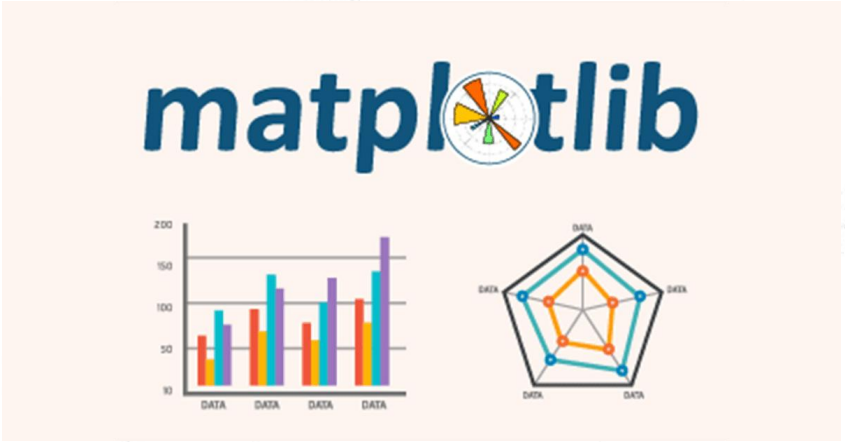







# Data Visualization with Python Matplotlib and Seaborn

## NIELIT Chandigarh/Ropar


*"In God we trust, all others must bring data." – W. Edwards Deming*

# Introduction to Matplotlib


- **What is Matplotlib?**
- Matplotlib is a Python library used for data visualization.
- It allows the creation of static, animated, and interactive plots.
- Highly customizable and widely used in data science and machine learning.

# Key Features

- Variety of plots: Line, Bar, Histogram, Scatter, etc.
- High level of customization (titles, labels, legends, etc.).
- Support for multiple output formats (PNG, PDF, etc.).
- Integrates with libraries like NumPy and Pandas.

# Installing Matplotlib

```
pip install matplotlib
```

**Basic Workflow**

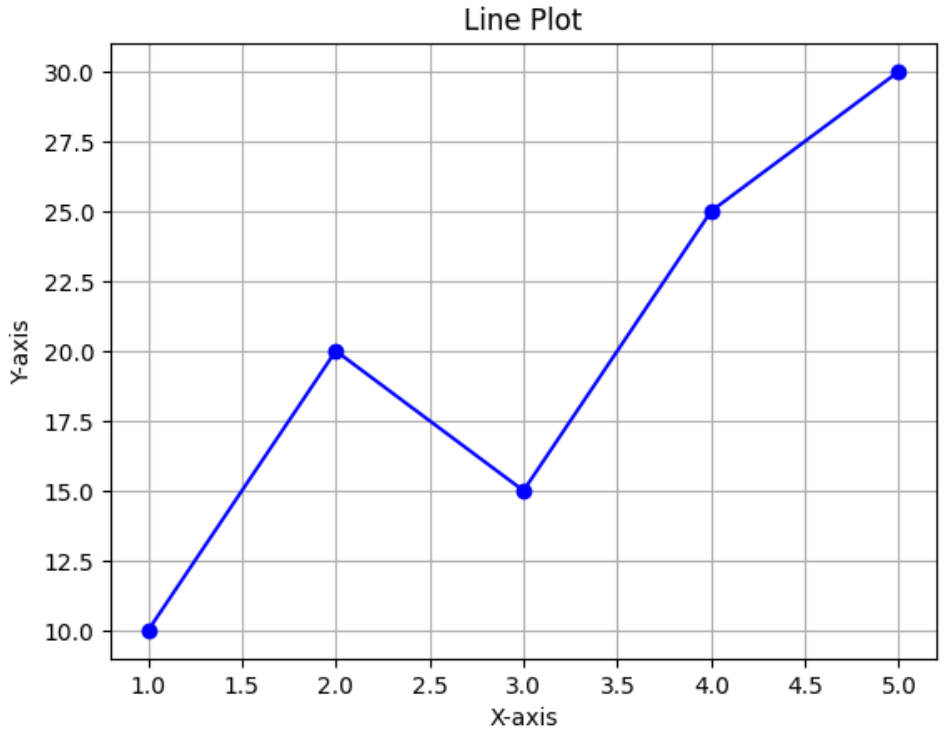
1. Import the library: **import matplotlib.pyplot as plt**
2. Prepare the data: Arrays or lists or load data from CSV
3. Create a plot using functions like **plt.plot()**.
4. Customize the plot: Titles, labels, legends.
5. Display or save the plot using **plt.show()** or **plt.savefig()**.

# Common Plot Types

## Line Plot

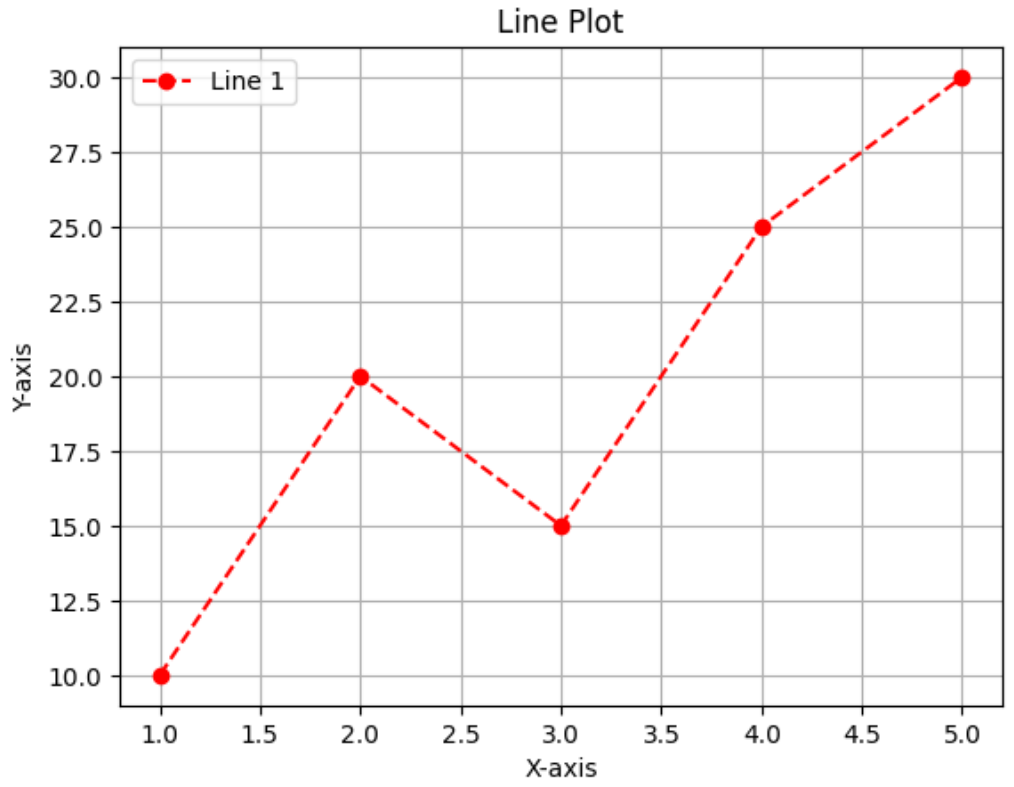
- Visualizes trends over time or sequences

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [10, 20, 15, 25, 30]
plt.plot(x, y, marker='o', linestyle='-', color='blue')
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.grid()
plt.show()
```



# Line Plot (Visualizes trends over time or sequences.)

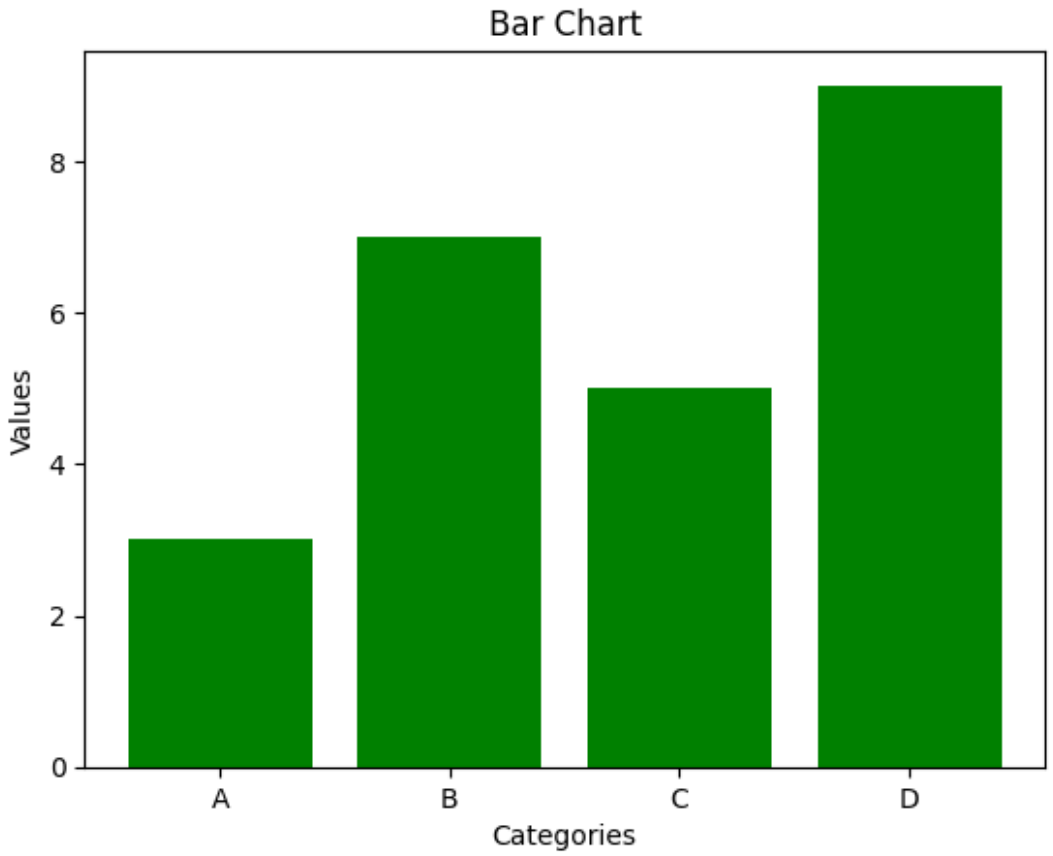
```
import matplotlib.pyplot as plt
# Sample Data

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

y = [10, 20, 15, 25, 30]
# Create Line Plot

plt.plot(x, y, marker='o', linestyle='--', color='r',
label='Line 1')

plt.title("Line Plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.legend()

plt.grid()

plt.show()
```



# Bar Chart

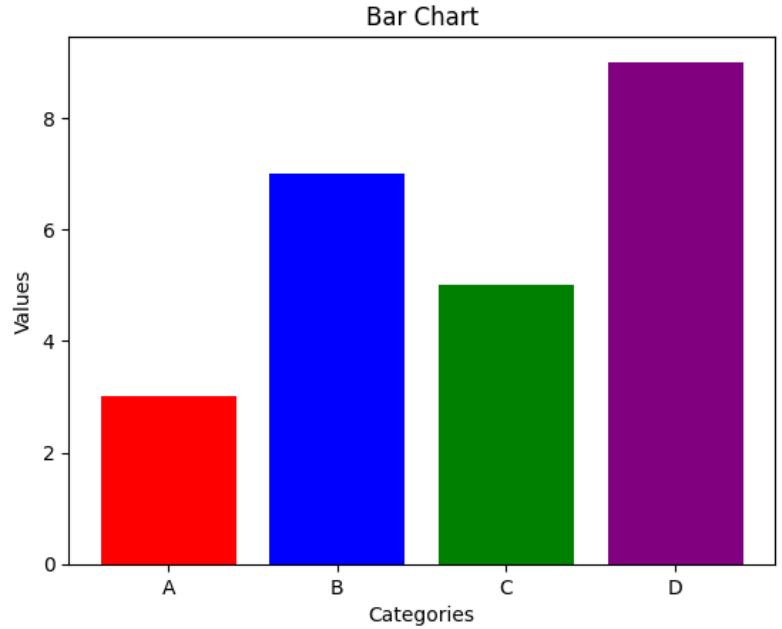
- Compares categorical data.

```
categories = ['A', 'B', 'C', 'D']
values = [3, 7, 5, 9]
plt.bar(categories, values, color='green')
plt.title("Bar Chart")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```



# Bar Chart (Compares categorical data.)

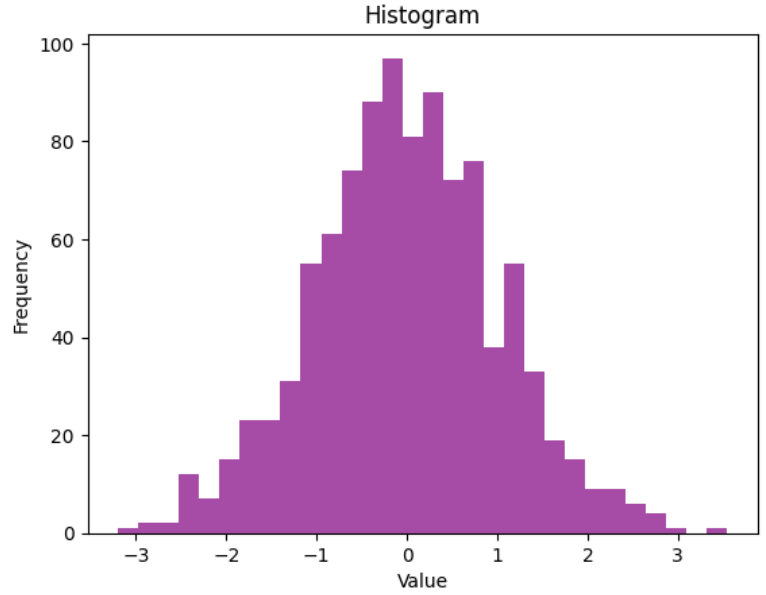
```
# Sample Data
categories = ['A', 'B', 'C', 'D']
values = [3, 7, 5, 9]

# Create Bar Chart
plt.bar(categories, values, color=['red', 'blue', 'green', 'purple'])
plt.title("Bar Chart")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```



# Histogram

- Displays the distribution of a dataset.

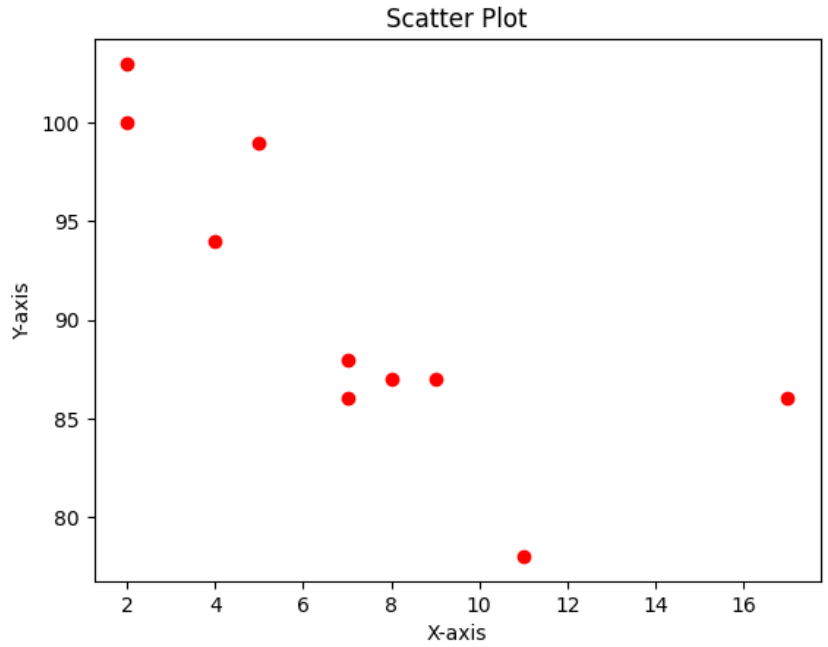
```
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, color='purple', alpha=0.7)

plt.title("Histogram")

plt.xlabel("Value")

plt.ylabel("Frequency")

plt.show()
```



# Scatter Plot

- Displays relationships between two variables.

```
x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11]

y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]


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



# Customizing Plots

- **Adding Titles and Labels**

```
plt.title("Title Here")
plt.xlabel("X-axis Label")
plt.ylabel("Y-axis Label")
```

- **Adding Legends**

```
plt.plot(x, y, label="Line 1") plt.legend()
```

- **Adding Grids**

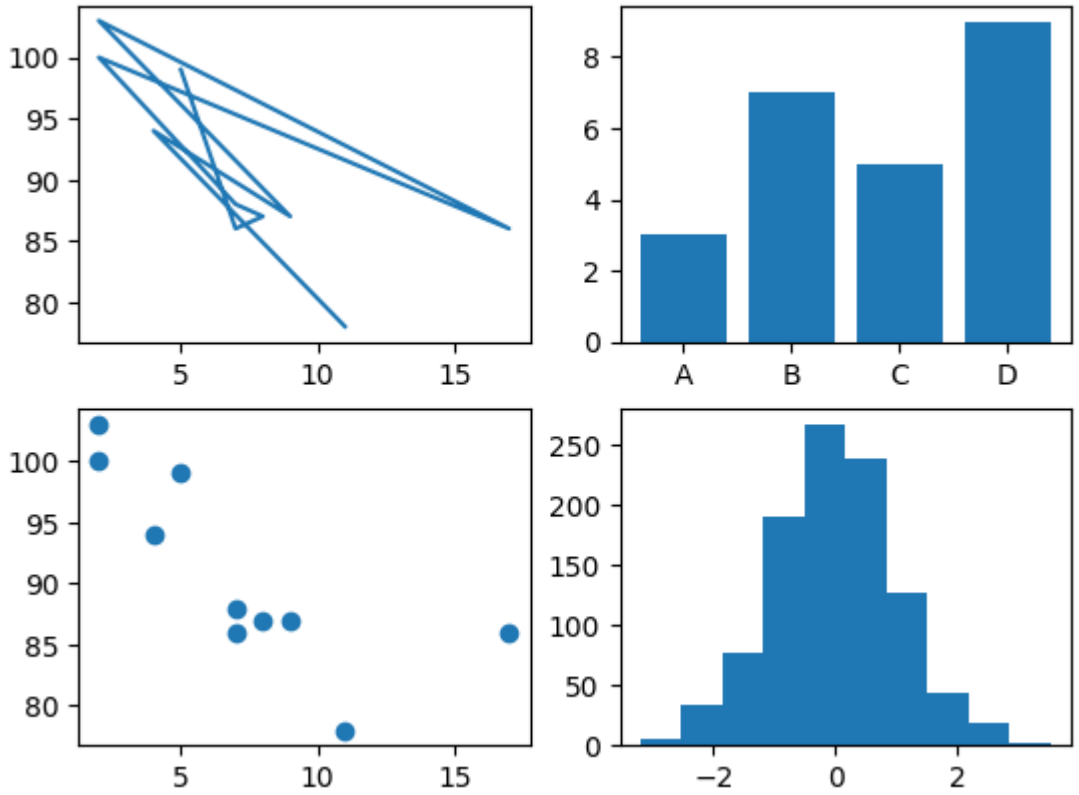
```
plt.grid()
```

# Advanced Features

## Subplots

- Multiple plots in one figure.

```
fig, axs = plt.subplots(2, 2)
axs[0, 0].plot(x, y)
axs[0, 1].bar(categories, values)
axs[1, 0].scatter(x, y)
axs[1, 1].hist(data)
plt.show()
```



# Saving Plots

```
plt.savefig("plot.png")
```

# Save the plot as a PDF `plt.savefig('plot.pdf')`

**When to Use Matplotlib?**

- When creating simple and static visualizations.
- When you need detailed customization.
- Ideal for data exploration and reporting.

**Limitations**

- Not as aesthetically modern as Seaborn or Plotly.
- Requires more effort for advanced visualizations.

# Best Practices & Conclusion

- Label axes and add titles for clarity.
- Use legends for multiple datasets.
- Choose appropriate plot types for your data.
- Maintain consistent styles for readability.

**Conclusion**

- Matplotlib is a powerful and versatile library for visualizing data.
- Its wide range of features and customization options make it a go-to tool for data scientists and analysts.
- **Explore more at:** Matplotlib Documentation

# Introduction to Seaborn

## What is Seaborn?

- Seaborn is a Python data visualization library built on top of Matplotlib.
- Provides a high-level interface for drawing attractive and informative statistical graphics.
- Works seamlessly with Pandas DataFrames and NumPy arrays.

# Key Features

- Easy creation of complex visualizations with minimal code.
- Built-in themes for better aesthetics.
- Statistical visualization capabilities (e.g., histograms, KDE plots).
- Integration with Pandas for handling data.
- Support for multi-plot grids.

**Installing Seaborn**

```
pip install seaborn
```

# Basic Workflow

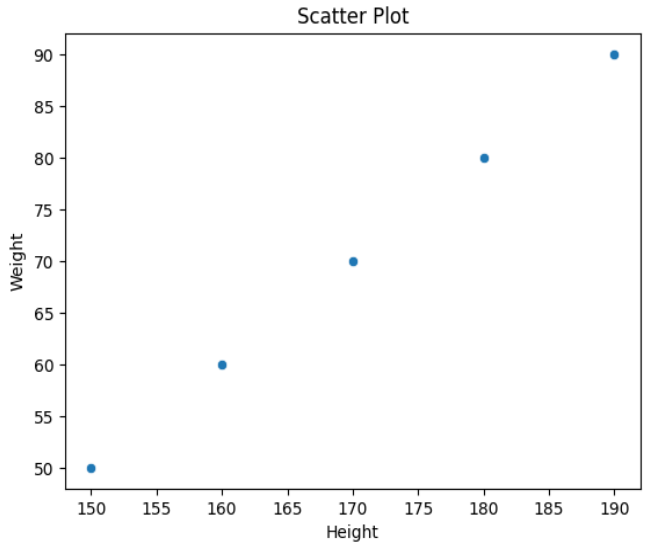
1. Import the library: import seaborn as sns.
2. Load or prepare your dataset (e.g., Pandas DataFrame).
3. Use Seaborn functions like sns.barplot() or sns.heatmap() to create plots.
4. Customize the plots using arguments, themes, and Matplotlib tools.
5. Display the plot using plt.show().

# Common Plot Types

## 1. Scatter Plot

- Visualizes relationships between two variables.

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

# Sample Data
data = {
    'Height': [150, 160, 170, 180, 190],
    'Weight': [50, 60, 70, 80, 90]
}

sns.scatterplot(x='Height', y='Weight', data=data)
plt.title("Scatter Plot")
plt.show()
```



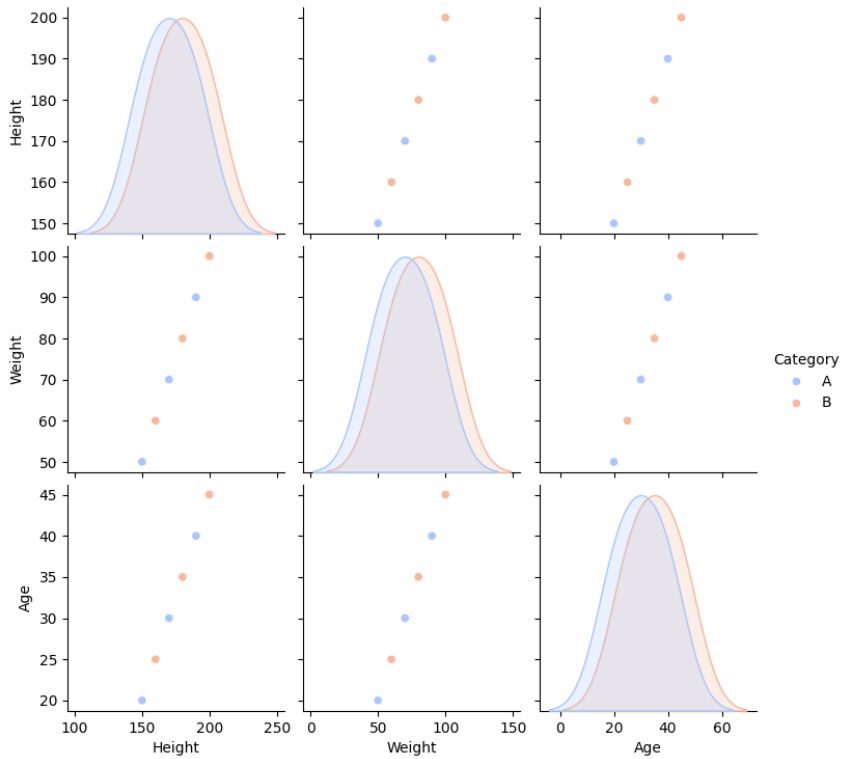
# 2. Pair Plot

**Pair Plot**: Use for relationships between multiple variables.

Creates pairwise scatter plots for all numeric columns in a dataset.

**Example**: Height, Weight, and Age with a categorical column.

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

# Sample Data (Custom Dataset)
data = {
    'Height': [150, 160, 170, 180, 190, 200],
    'Weight': [50, 60, 70, 80, 90, 100],
    'Age': [20, 25, 30, 35, 40, 45],
    'Category': ['A', 'B', 'A', 'B', 'A', 'B']
}

df = pd.DataFrame(data)

# Create Pair Plot
sns.pairplot(df, hue='Category', palette='coolwarm')
plt.show()
```

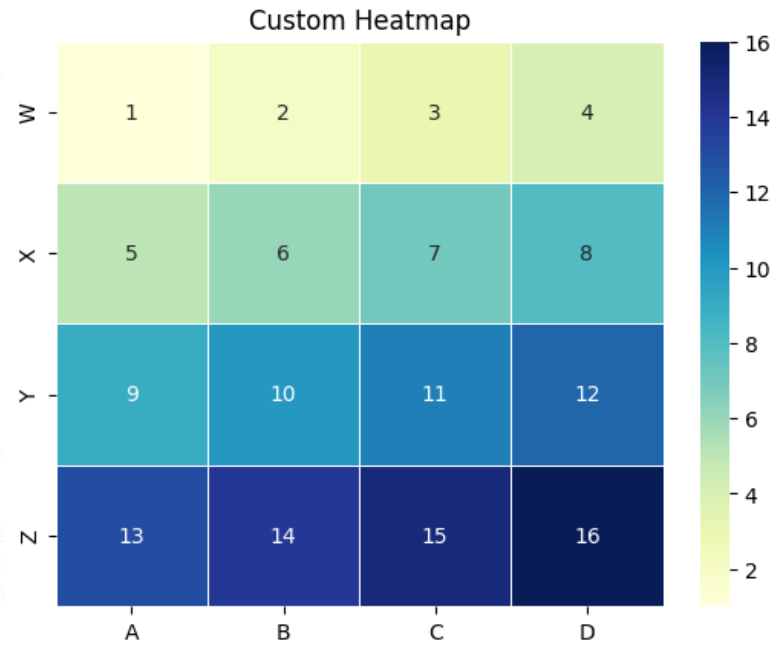


# 3. Heatmap

- Visualize matrix-like data.

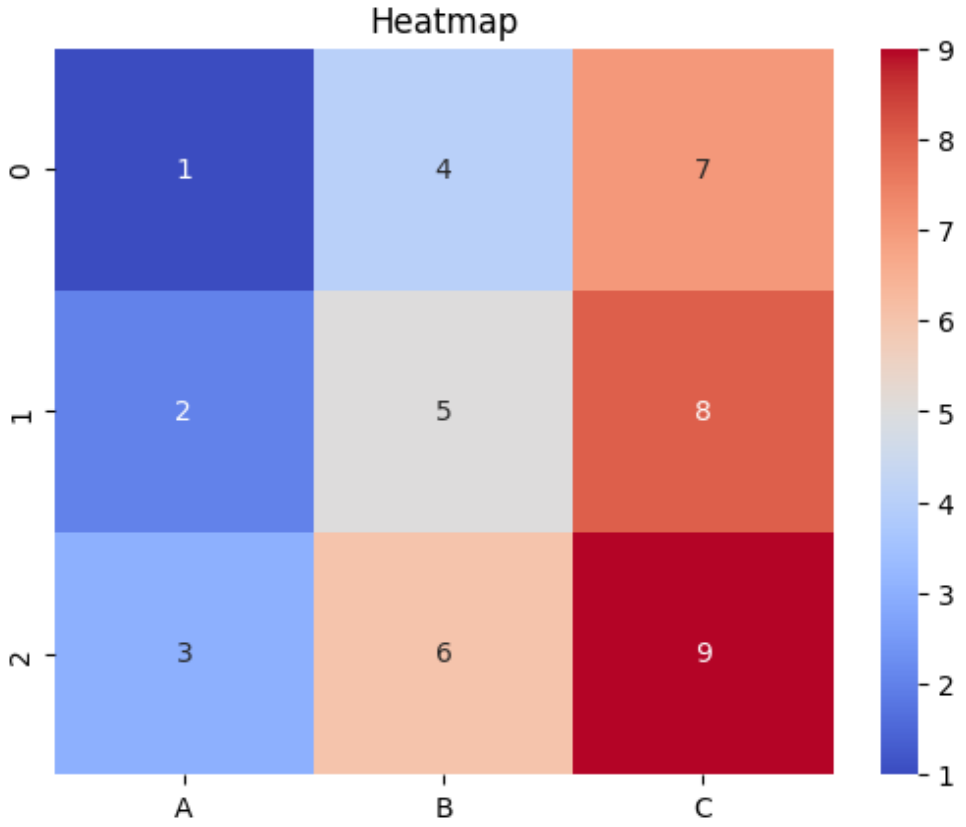
Example: Tabular data with rows and columns.

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Custom Data
data = np.array([
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12],
    [13, 14, 15, 16]
])

# Create a DataFrame for better labeling
df = pd.DataFrame(data, columns=['A', 'B', 'C', 'D'], index=['W', 'X', 'Y', 'Z'])

# Create Heatmap
sns.heatmap(df, annot=True, cmap='YlGnBu', linewidths=0.5)
plt.title("Custom Heatmap")
plt.show()
```



# Heatmap example 2

```
import seaborn as sns
import pandas as pd

# Sample Data
data = pd.DataFrame({
        'A': [1, 2, 3],
        'B': [4, 5, 6],
        'C': [7, 8, 9]
})

sns.heatmap(data, annot=True,
cmap='coolwarm’)

plt.title("Heatmap")
plt.show()
```

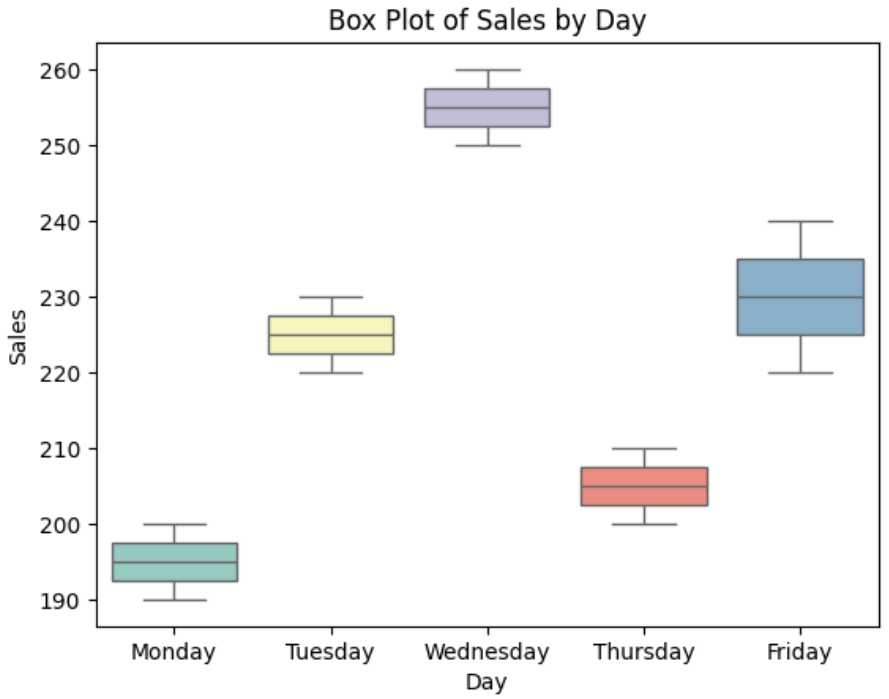


# 4. Box Plot

- Analyze distributions and detect outliers.

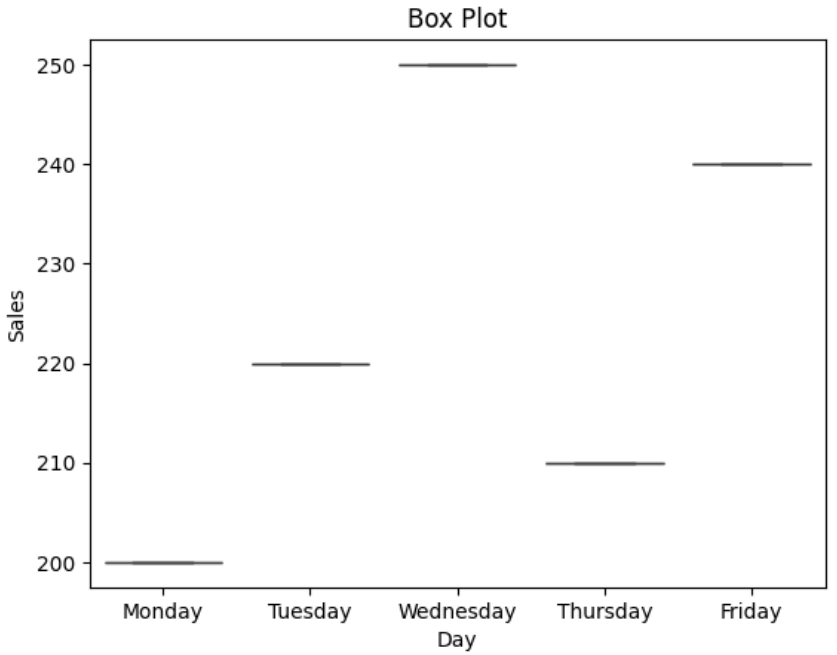
Example: Sales data over different days.

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

# Custom Data
data = {
    'Day': ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
            'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
    'Sales': [200, 220, 250, 210, 240, 190, 230, 260, 200, 220]
}

df = pd.DataFrame(data)

# Create Box Plot
sns.boxplot(x='Day', y='Sales', data=df, palette='Set3')
plt.title("Box Plot of Sales by Day")
plt.show()
```



# Box Plot Example 2

```
import seaborn as sns

# Sample Data
data = {
    'Day': ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],
    'Sales': [200, 220, 250, 210, 240]
}

sns.boxplot(x='Day', y='Sales', data=data)
plt.title("Box Plot")
plt.show()
```



# Customizing Seaborn Plots

## Themes

- Available themes: darkgrid, whitegrid, dark, white, ticks.
- sns.set_theme(style="whitegrid")

## Adding Titles and Labels

- plt.title("Title Here")
- plt.xlabel("X-axis Label")
- plt.ylabel("Y-axis Label")

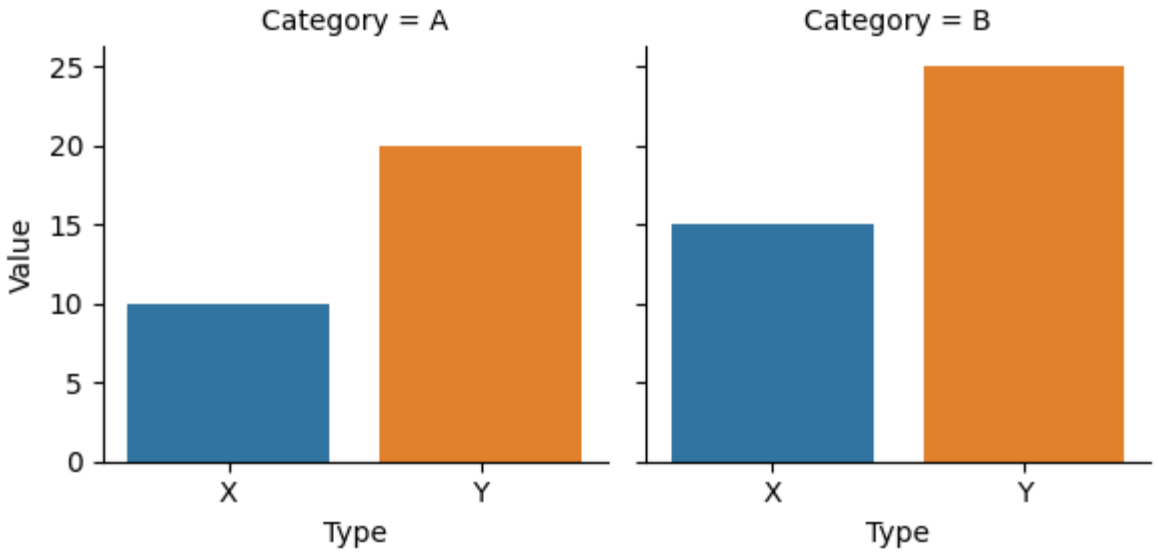
## Adjusting Color Palettes

- sns.set_palette("pastel")

# Advanced Features

- **FacetGrid**
- Allows creation of multiple plots based on subsets of data.

```
import seaborn as sns
import pandas as pd

# Sample Data
data = pd.DataFrame({
    'Category': ['A', 'A', 'B', 'B'],
    'Value': [10, 20, 15, 25],
    'Type': ['X', 'Y', 'X', 'Y']
})

facet = sns.FacetGrid(data, col="Category", hue="Type")
facet.map(sns.barplot, "Type", "Value")
plt.show()
```



# When to Use Seaborn?


- When creating aesthetically pleasing visualizations quickly.
- When visualizing statistical data with complex relationships.
- For integrating with Pandas for data analysis workflows

# Limitations & Best Practices

## Limitations

- Not as flexible as Matplotlib for very customized plots.
- Requires Matplotlib for certain advanced features.

## Best Practices

- Use themes to maintain consistency.
- Choose the right plot type for your data.
- Combine Seaborn with Matplotlib for advanced customization.

# Conclusion


- Seaborn simplifies statistical data visualization in Python.
- Its ease of use and integration with Pandas make it ideal for data scientists.

**Explore more at:** Seaborn Documentation


DATA SCIENCE
MACHINE LEARNING
ANALYTICS
DATA MINING
PREDICTIVE
PROCESSING
MODELS
WEB MARKETING
PROJECTS
PATTERN
ENGINEERING
VISUALIZATION
STATISTICS
BIG DATA
KDD

# Thank You! ☺