M.Tech Program

Advanced Industry Integrated Programs

Jointly offered by University and LTIMindTree

Python for Data Science

Knowledge partner

LTIMindtree

Implementation partner

L&T EduTech

Python -Visualization

Modules to cover….

1. Matplotlib Basics
2. Understanding figure Object
3. Matplotlib Figure parameter & Styling
4. Matplotlib Advanced Commands
5. Seaborn
6. Seaborn-Scatter Plots, Distribution Plots, Categorical Plots

Learning Outcome…

- **Understand the Basics of Matplotlib:**
  - Learn the fundamental components and functionalities of Matplotlib
- **Implement and Customize Figures and Axes:**
  - Gain proficiency in creating and customizing figures and axes using Matplotlib
- **Master Advanced Matplotlib Commands:**
  - Explore advanced plotting techniques in Matplotlib
- **Explore Seaborn & Analyze Data with Seaborn :**
  - Understand the basics of Seaborn and its advantages over Matplotlib, Learn how to create and customize various plot types such as scatterplots, distribution plots, and categorical plots.

Python Visualization

Introduction to Matplotlib

- **Overview of Matplotlib:-**
  - It is a comprehensive library for creating static, animated, and interactive visualizations in Python.
  - It is widely used in the scientific Python community for data visualization.
- **History & Development:-**
  - **Created by:-** John D. Hunter in 2003.
  - **Purpose:-** Designed to replicate MATLAB's plotting capabilities in Python.
  - **Development:-** Actively developed and maintained by a large community of contributors.

Python Visualization

Installation & Setup

- **Installation via pip:**
  - python –m pip install –U pip
  - python –m pip install –U matplotlib
- **Installation via conda:**
  - conda install matplotlib
- **Importing Matplotlib in your Python script:**
  - import matplotlib.pyplot as plt

Python Visualization

Matplotlib Basics

- **Basic Components of a Matplotlib Plot:**
  - **Figure:** The overall window or page that everything is drawn on.
  - **Axes:** The area on which data is plotted, containing x and y (or theta in polar plots) Axis.
  - **Axis:** The part of the plot that includes tick marks and labels.

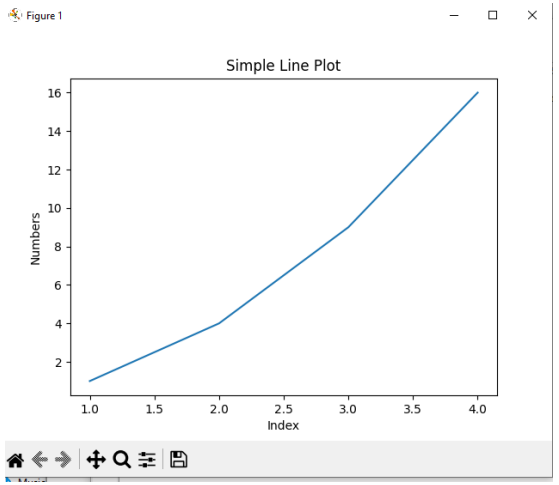
Python Visualization

Simple Plot Creation

Explaining the Code:

- plt.plot(): Creates a line plot with the provided x and y data.
- plt.ylabel(): Sets the label for the y-axis.
- plt.xlabel(): Sets the label for the x-axis.
- plt.title(): Adds a title to the plot.
- plt.show(): Displays the plot.

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.ylabel('Numbers')
plt.xlabel('Index')
plt.title('Simple Line Plot')
plt.show()
```



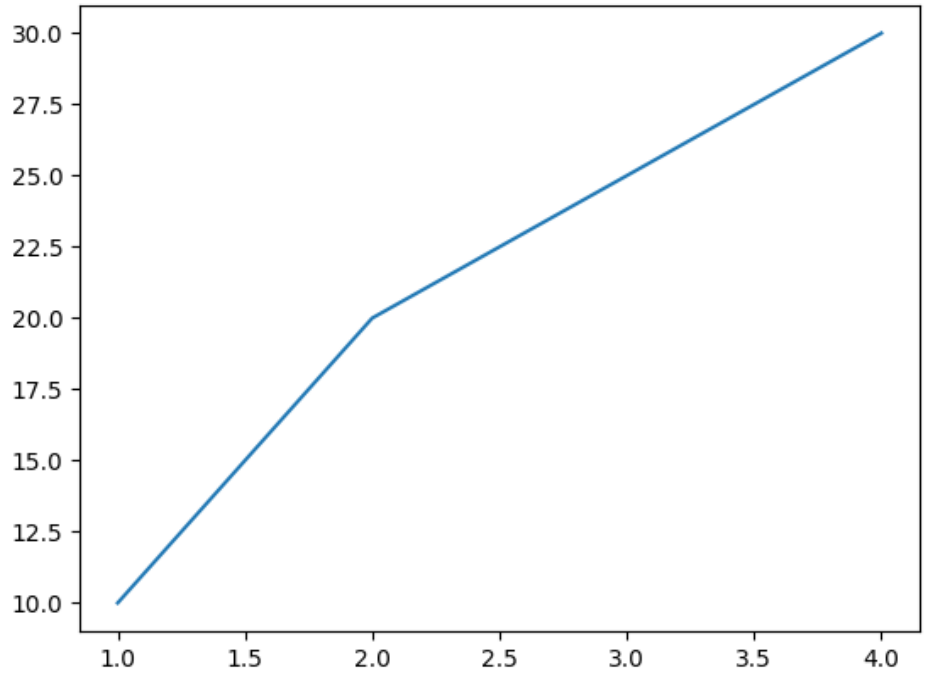
Python Visualization

Understanding the Figure Object

Definition and Role of Figure in Matplotlib:-

- **Figure:-** The container for all plot elements. It can contain multiple Axes (subplots), a title, and other plot elements.
- **Creating Figure:-**

```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([1, 2, 3, 4], [10, 20, 25, 30])
plt.show()
```



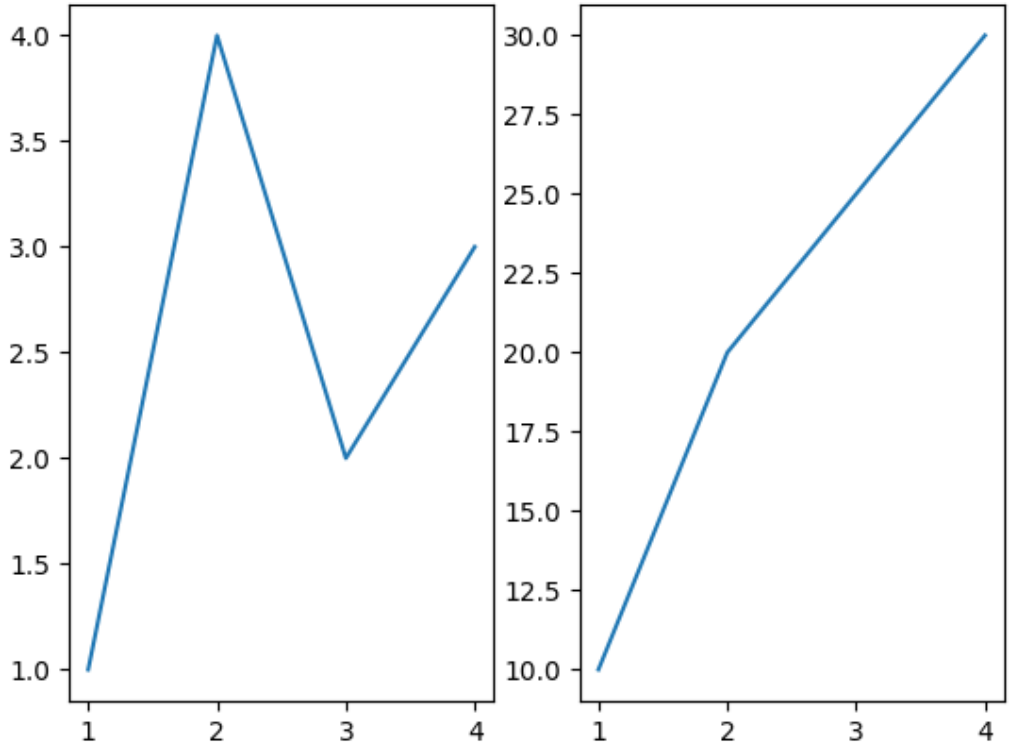
Python Visualization

Implementing Figures and Axes

Introduction to Axes:

- **Axes:** The area on which data is plotted. It is an individual plot within a Figure.
- **Adding Multiple Axes:**
  - **Example: Multiple Subplots**

```
import matplotlib.pyplot as plt
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax2.plot([1, 2, 3, 4], [10, 20, 25, 30])
plt.show()
```

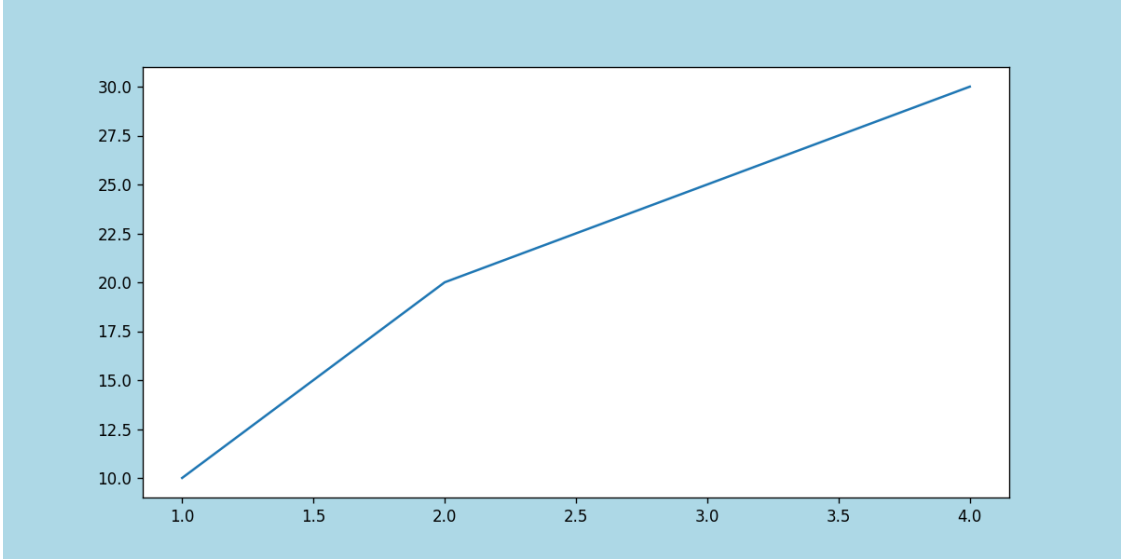


Python Visualization

Figure Parameters

- **Figure Size:** Adjust the overall size of the figure.
- **DPI (Dots Per Inch):** Controls the resolution of the figure.
- **Background Color:** Set the background color of the figure.
  - **Example: Custom Figure Size and DPI:**

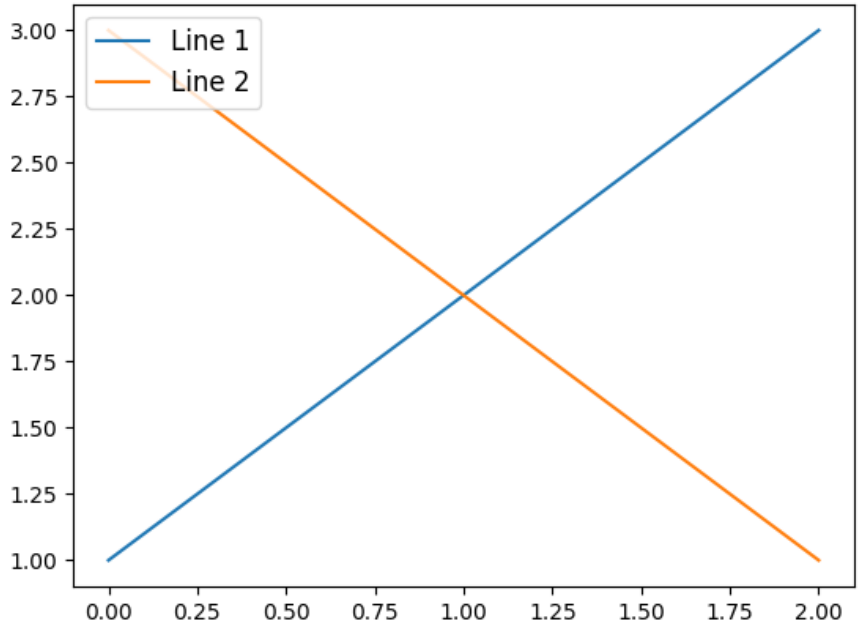
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(10, 5), dpi=120)
fig.patch.set_facecolor('lightblue')
ax = fig.add_subplot(111)
ax.plot([1, 2, 3, 4], [10, 20, 25, 30])
plt.show()
```



Python Visualization

Matplotlib Styling - Legends

- **Adding and Customizing Legends:**
  - Legends help in identifying different data series on the plot.
- import matplotlib.pyplot as plt
- plt.plot([1, 2, 3], label='Line 1')
- plt.plot([3, 2, 1], label='Line 2')
- plt.legend(loc='upper left',fontsize='large')
- plt.show()

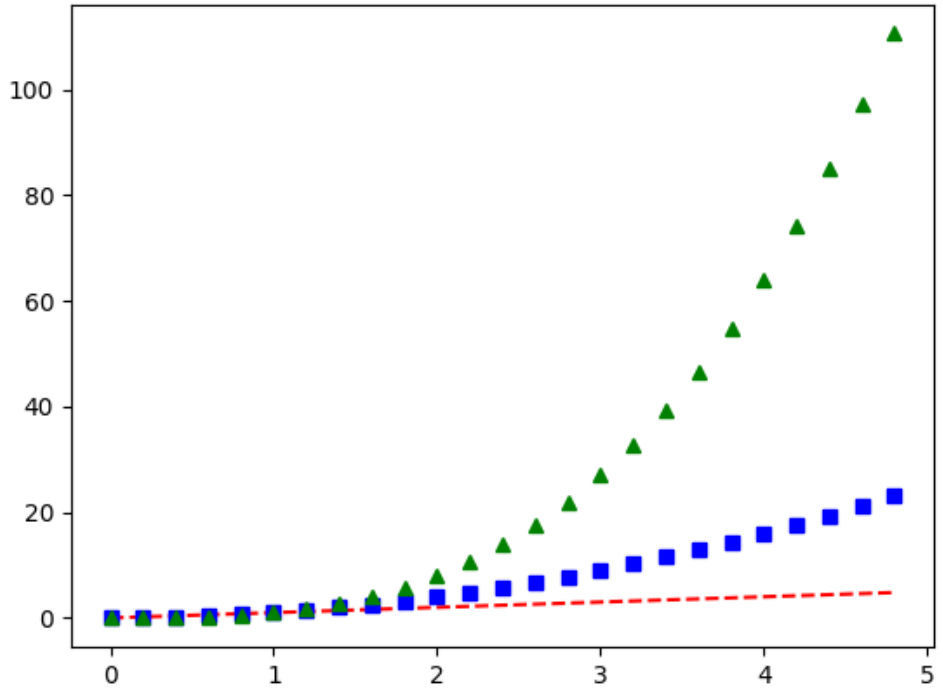


Python Visualization

Matplotlib Styling - Colors and Styles

- **Customizing Plot Colors and Styles:**
  - **Line Styles:** Change the style of the lines (solid, dashed, etc.)

```
import numpy as np
import matplotlib.pyplot as plt
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



Python Visualization

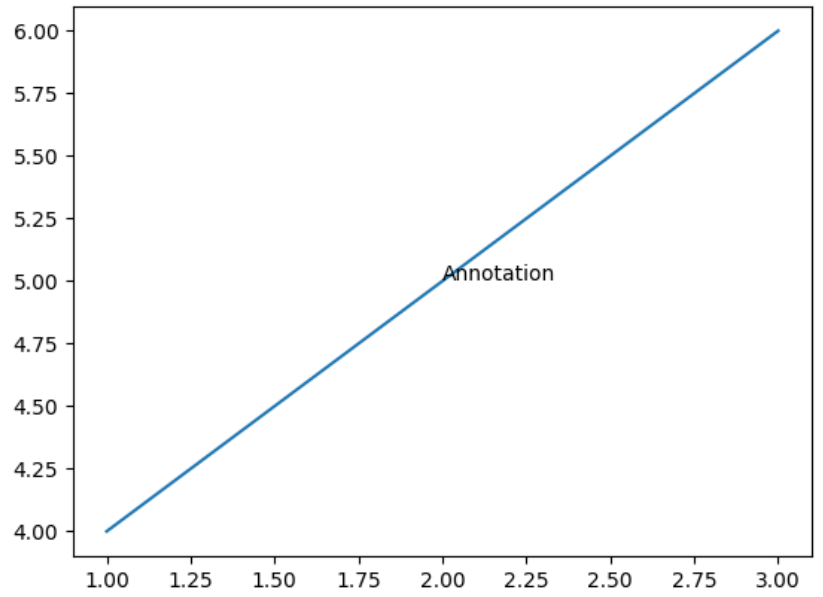
Advanced Matplotlib Commands

- **Advanced Plotting Techniques:**
  - **Annotations and Text:** Adding text annotations to the plot.

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.text(2, 5, 'Annotation')
plt.show()
```

  - **Arrow Props:** Customize the arrow style in annotations

```
plt.annotate('Note', xy=(2, 5), xytext=(3, 4),
arrowprops=dict(facecolor='black', shrink=0.05))
plt.show()
```



Python Visualization

Introduction to Seaborn

Overview of Seaborn:

- Seaborn is a Python data visualization library based on Matplotlib.
- Provides a high-level interface for drawing attractive and informative statistical graphics.

Differences and Advantages Over Matplotlib:

- **Simpler Syntax:** Easier to create complex visualizations.
- **Built-in Themes:** Attractive default styles and color palettes.
- **Integration:** Works well with Pandas data structures.
- **Statistical Plots:** Simplifies the creation of complex plots like violin plots and pair plots.

Python Visualization

Installation Seaborn

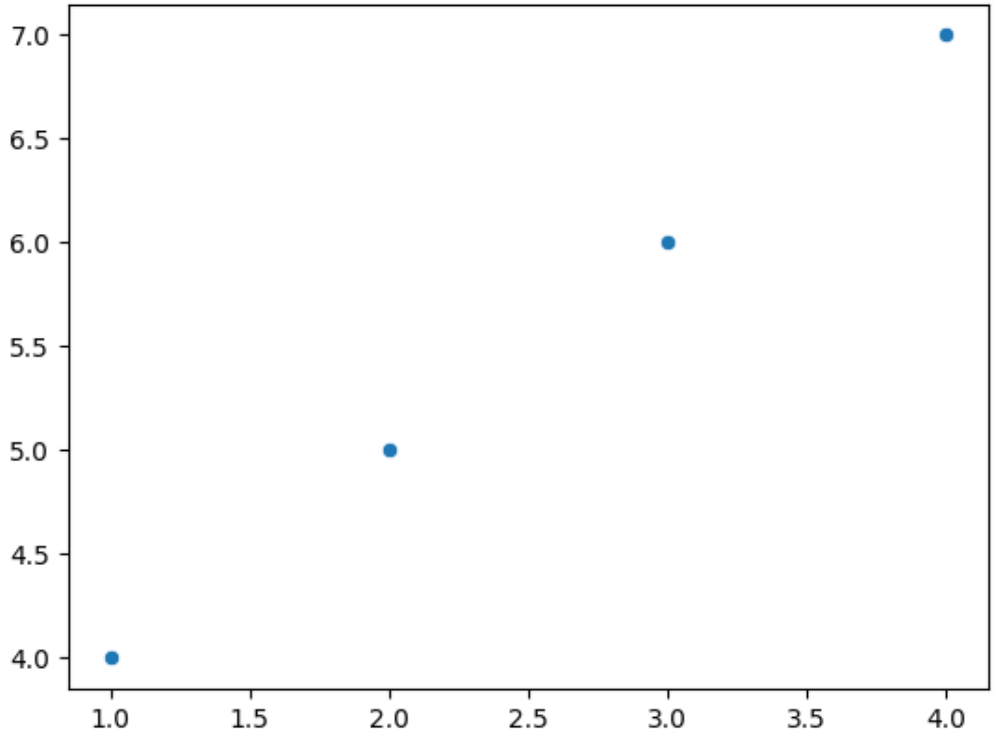
- **Installation via pip:**
  - pip install seaborn
- **Installation via conda (Anaconda distribution):**
  - conda install seaborn
- **Basic Setup:**
  - import seaborn as sns

Python Visualization

Scatterplots with Seaborn

Basic Scatterplot Example:-

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x=[1, 2, 3, 4], y=[4, 5, 6, 7])
plt.show()
```



Python Visualization

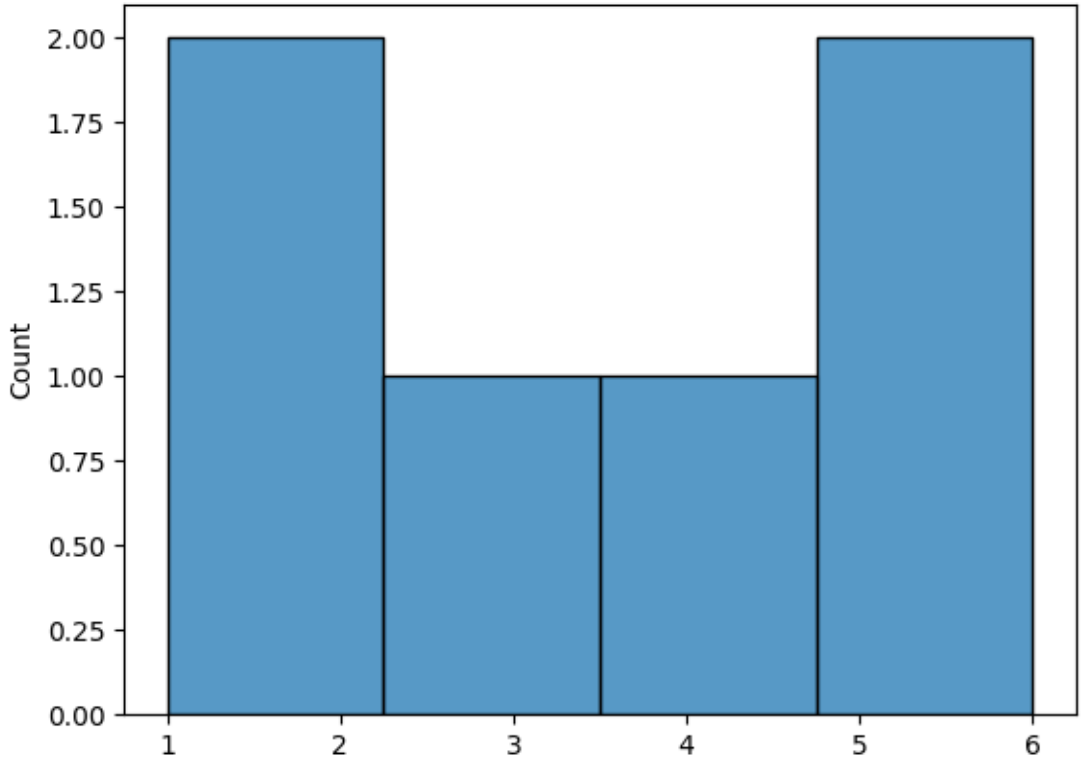
Distribution Plots - Part One

Introduction to Distribution Plots:

- **Purpose:** Visualize the distribution of a dataset.
- **Types:** Histograms, KDE plots, and Rug plots.

Histogram Example:

```
sns.histplot(data=[1, 2, 3, 4, 5, 6])
plt.show()
```



Python Visualization

Understanding Plot Types (Distribution Plots)

Overview of Distribution Plot Types:

- **Histogram:** Shows the frequency of data points in bins.
- **KDE Plot:** Kernel Density Estimate, smoothes the histogram into a continuous line.
- **Rug Plot:** Places a tick mark at each data point along the x-axis

When to Use Each Type:

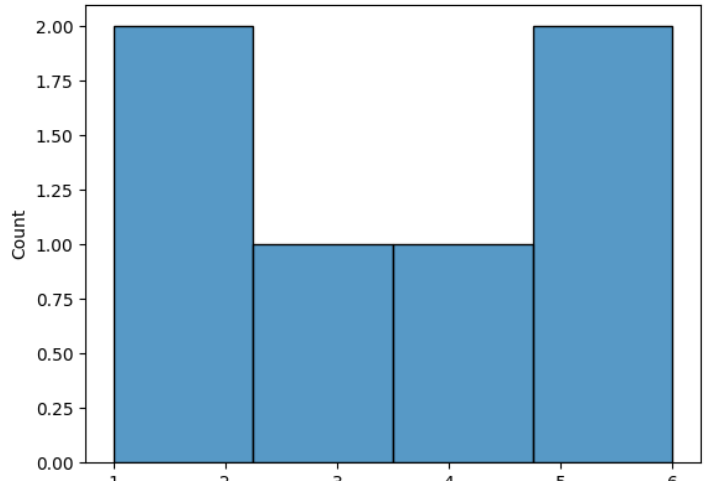
- **Histogram:** Good for visualizing the frequency distribution of numerical data.
- **KDE Plot:** Useful for estimating the probability density function of a random variable.
- **Rug Plot:** Enhances other plots by showing individual data points.

Python Visualization

Advanced Distribution Plots:

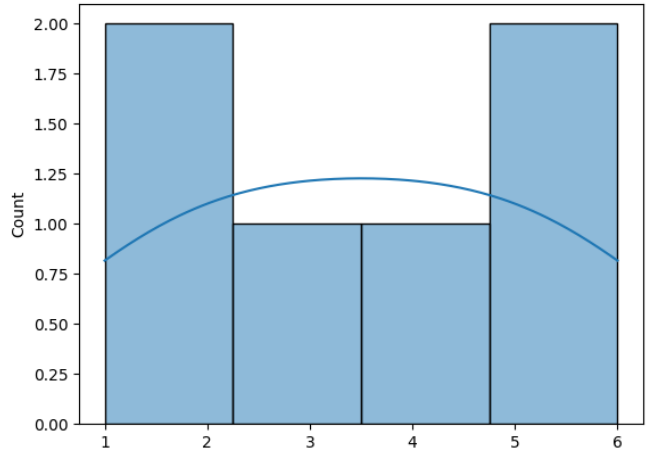
- **KDE Plot Example:**

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.kdeplot(x=[1, 2, 3, 4, 5, 6], fill=True)
plt.show()
```



- **Histogram Example:**

```
sns.histplot(x=[1, 2, 3, 4, 5, 6], kde=True) plt.show()
```



Python Visualization

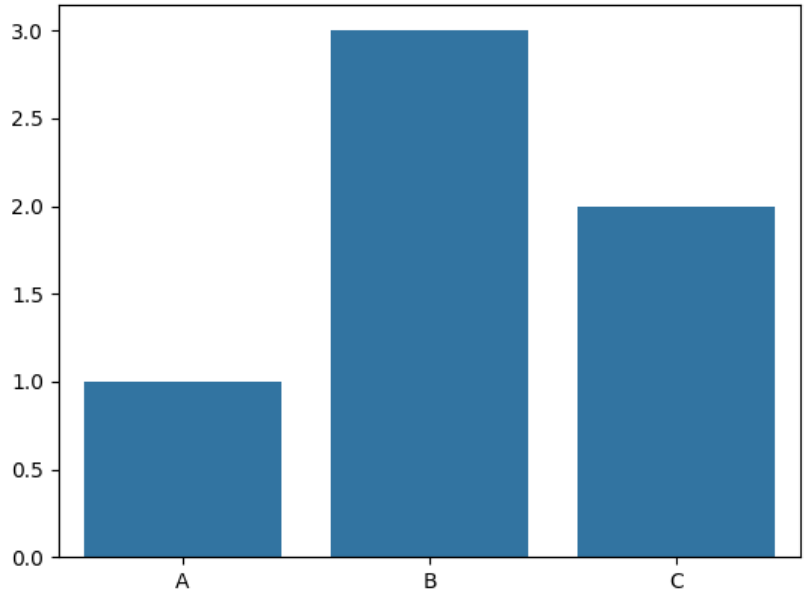
Coding with Seaborn - Categorical Plots

- **Introduction to Categorical Plots:**

Purpose: Visualize the relationship between categorical variables and continuous variables.

Types: Bar plots, count plots, box plots, violin plots, etc.

```
sns.barplot(x=['A', 'B', 'C'], y=[1, 3, 2])
plt.show()
```



- **Explaining the Code:**

sns.barplot(x=['A', 'B', 'C'], y=[1, 3, 2]):

Creates a bar plot with specified categories and values.

Python Visualization

Different Types of Categorical Plots:

- **Bar Plot:** Displays the mean (or other statistic) of a numerical variable for different categories.
- **Count Plot:** Shows the count of observations in each categorical bin using bars.
- **Box Plot:** Visualizes the distribution of data across categories using quartiles.
- **Violin Plot:** Combines a box plot and a KDE plot to show data distribution.

Python Visualization

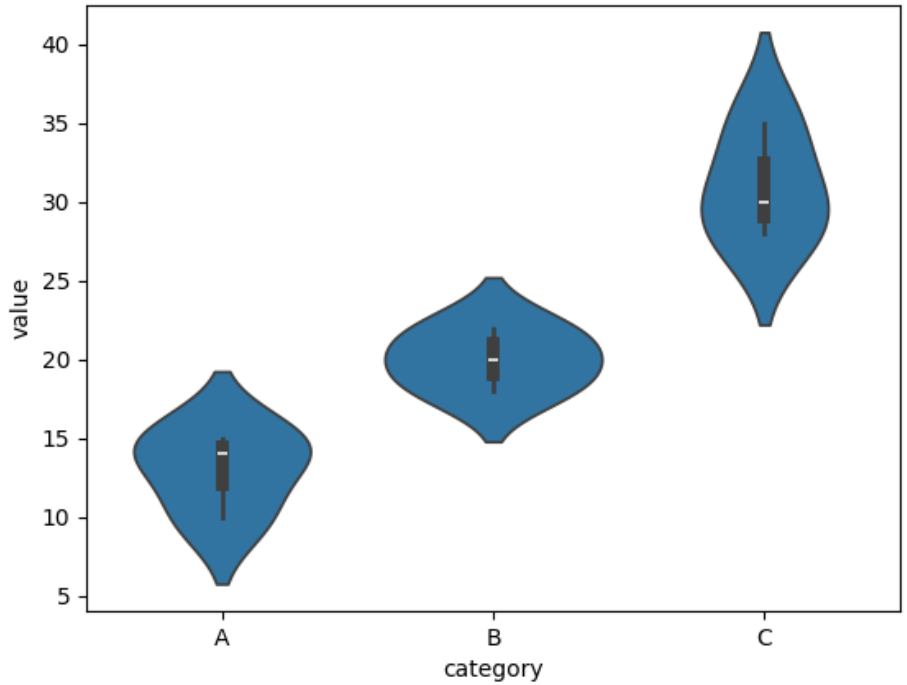
Visualizing Distributions within Categories:-

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

# Creating a sample dataset
data = {
    'category': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
    'value': [10, 15, 14, 20, 18, 22, 30, 28, 35]
}

df = pd.DataFrame(data)

# Creating a violin plot
sns.violinplot(x='category', y='value', data=df)
plt.show()
```



Python Visualization

Coding with Seaborn - Comparison Plots

Introduction to Comparison Plots:

- **Purpose:** Compare relationships between multiple variables.
- **Types:** Pair plots, joint plots, etc.

Advantages of Pair Plots:

- Quickly visualize relationships and distributions for multiple variables.
- Identify patterns and correlations between variables.

Python Visualization

Deep Dive into Comparison Plots:

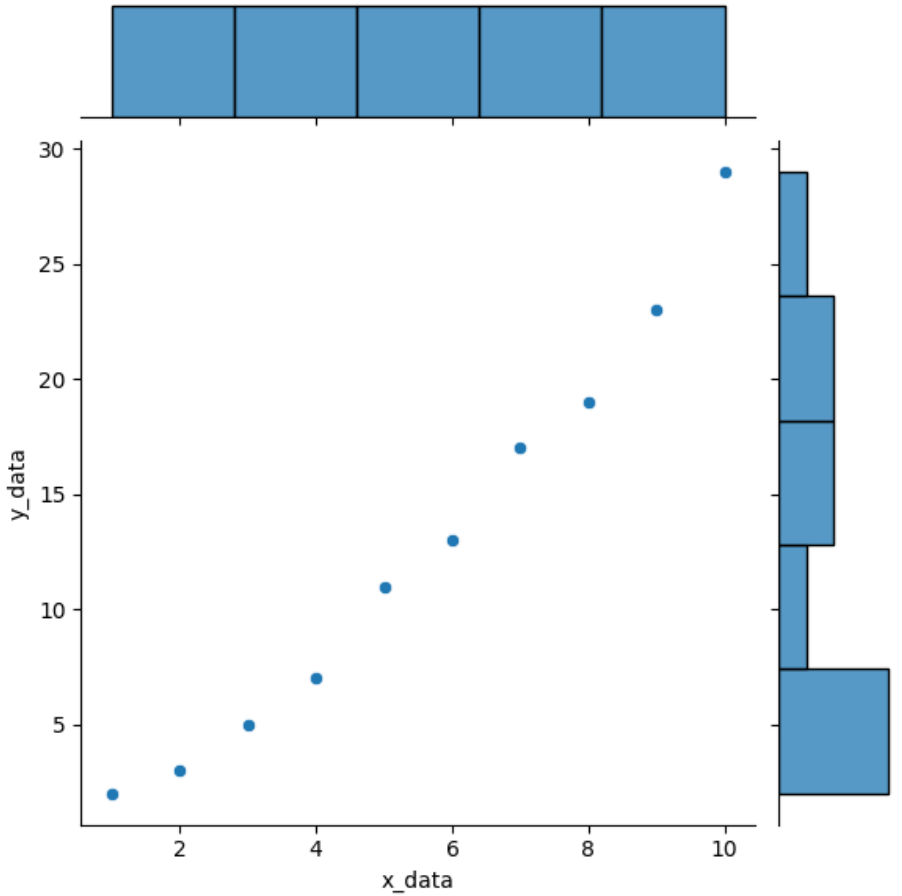
#Joint Plot Example:

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

# Creating a sample dataset
data = {
    'x_data': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'y_data': [2, 3, 5, 7, 11, 13, 17, 19, 23, 29] }

df = pd.DataFrame(data)

# Creating a joint plot
sns.jointplot(x='x_data', y='y_data', data=df)
plt.show()
```



Python Visualization

Seaborn Grid Plots

Introduction to Grid Plots:

- **Purpose:** Create multi-plot grids for detailed visualization.
- **Types:** FacetGrid, PairGrid, etc.
- **FacetGrid Example:**

```
g=sns.FacetGrid(data=your_dataframe,col='column_name’)
g.map(plt.hist, 'data_column’)
plt.show()
```

Advantages of Grid Plots:

- Allows for the comparison of multiple subsets of data.
- Facilitates the visualization of data patterns across different subsets.

Python Visualization

Seaborn Matrix Plots

Understanding Matrix Plots:-

- **Purpose:** Visualize data in a matrix format, such as correlation matrices.
- **Types:** Heatmaps, clustermaps, etc.
- **Heatmap Example:**

```
sns.heatmap(data=your_matrix_data, annot=True)
plt.show()
```

Advantages of Matrix Plots:

- Ideal for visualizing complex relationships in tabular data.
- Easy to identify patterns, correlations, and outliers.

Detailed Examples and Code Walkthroughs

Python Visualization

Bar Plot:-

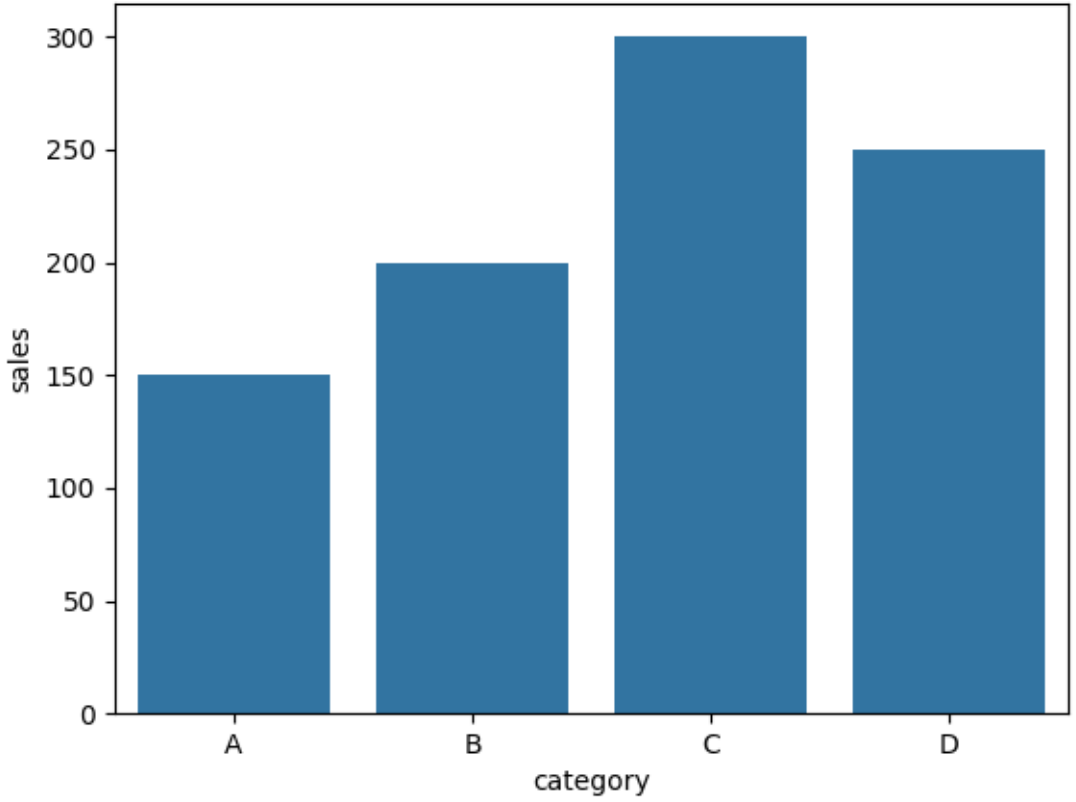
Data Example:

Context: Sales data for different product categories.

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

data = {
    'category': ['A', 'B', 'C', 'D'],
    'sales': [150, 200, 300, 250]
}
df = pd.DataFrame(data)

sns.barplot(x='category', y='sales', data=df)
plt.show()
```



Python Visualization

Count Plot:-

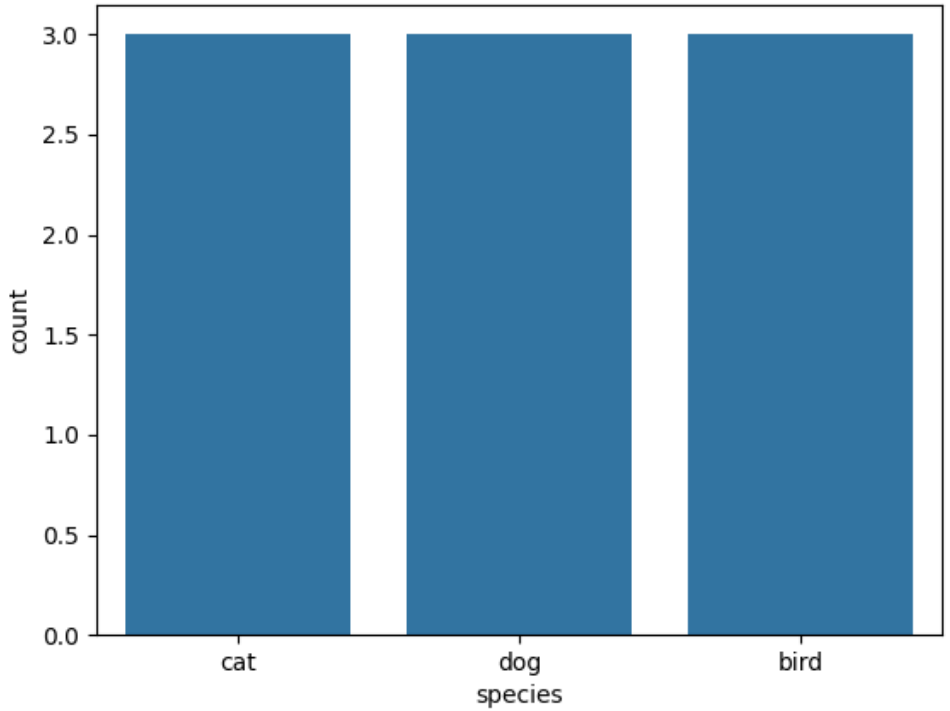
Data Example:

Context: Count of different species observed.

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

data = {
    'species': ['cat', 'dog', 'bird', 'cat', 'dog', 'cat', 'bird',
'bird', 'dog']
}
df = pd.DataFrame(data)

sns.countplot(x='species', data=df)
plt.show()
```



Python Visualization

Box Plot:-

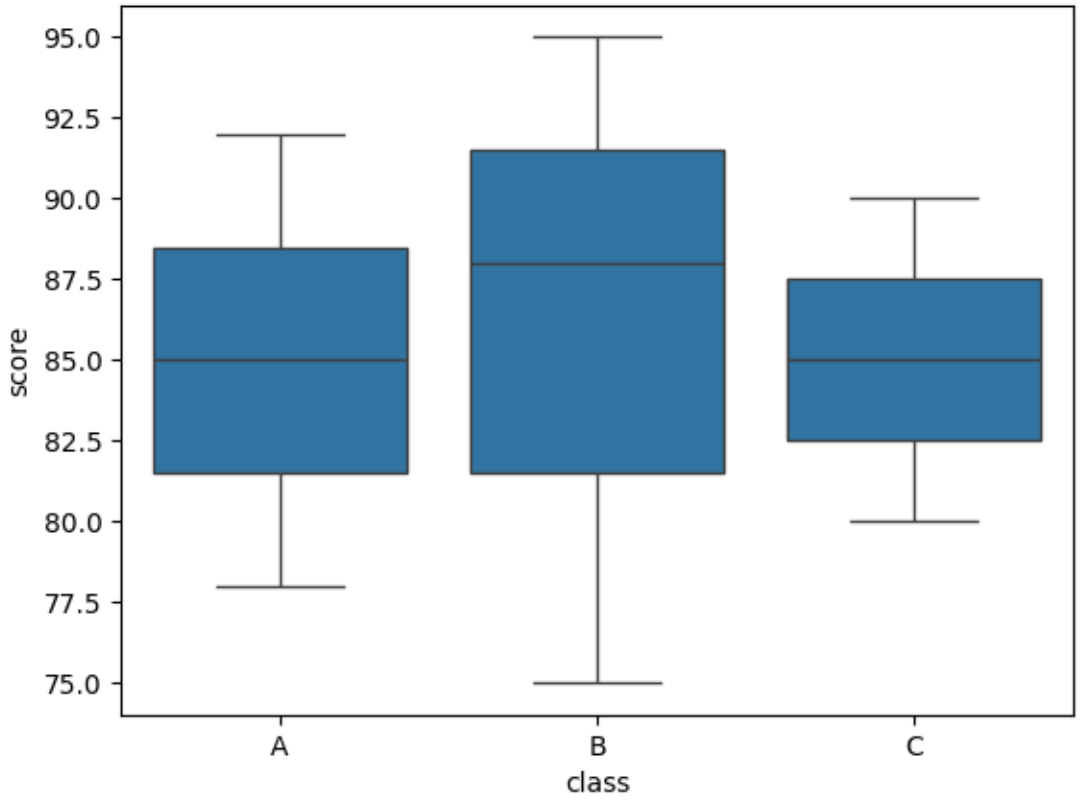
Data Example:

Context: Count of different species observed.

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

data = {
   'class': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
   'score': [85, 78, 92, 88, 75, 95, 90, 85, 80]
}
df = pd.DataFrame(data)

sns.boxplot(x='class', y='score', data=df)
plt.show()
```



Python Visualization

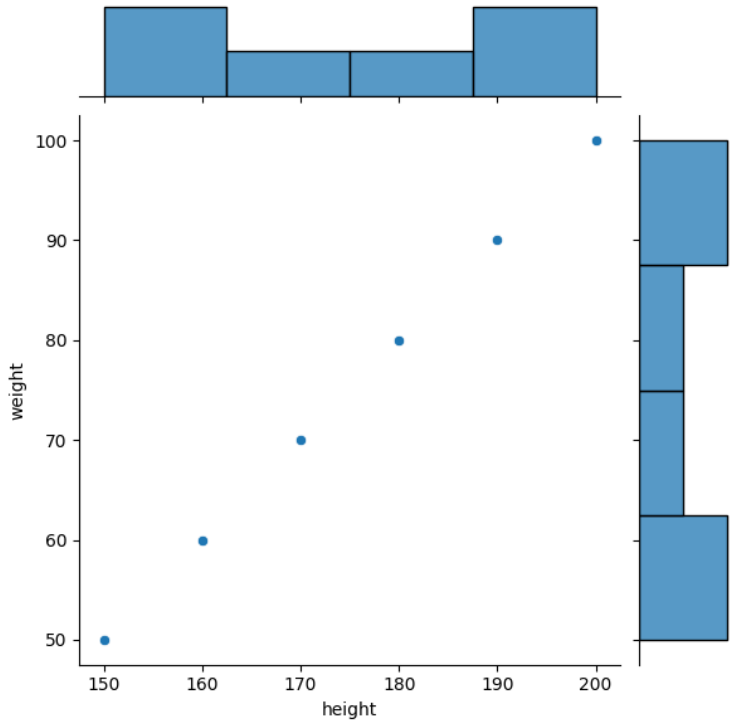
Pair Plot:-

Context: Relationship between different numerical features in the Iris dataset.

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

data = {
    'height': [150, 160, 170, 180, 190, 200],
    'weight': [50, 60, 70, 80, 90, 100]
}
df = pd.DataFrame(data)

sns.jointplot(x='height', y='weight', data=df)
plt.show()
```



Python Visualization

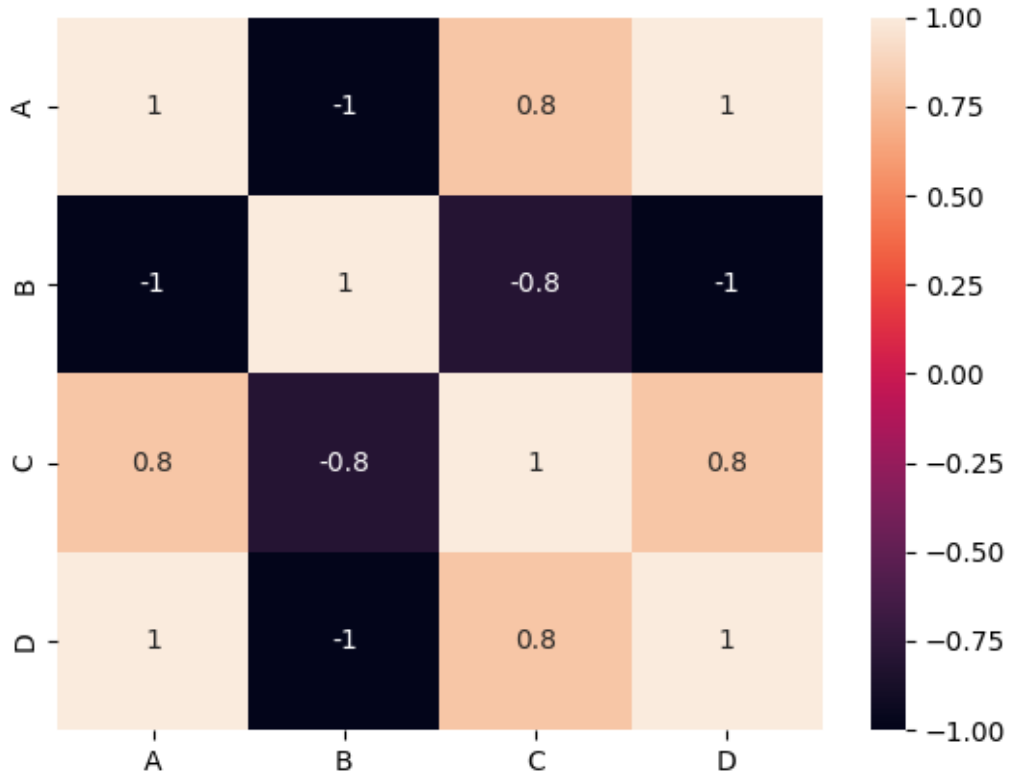
Heatmap

Context: Correlation matrix for a sample dataset.

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

data = {
    'A': [1, 2, 3, 4, 5],
    'B': [5, 4, 3, 2, 1],
    'C': [1, 3, 2, 5, 4],
    'D': [2, 3, 4, 5, 6]
}
df = pd.DataFrame(data)
corr = df.corr()

sns.heatmap(corr, annot=True)
plt.show()
```



Python Visualization

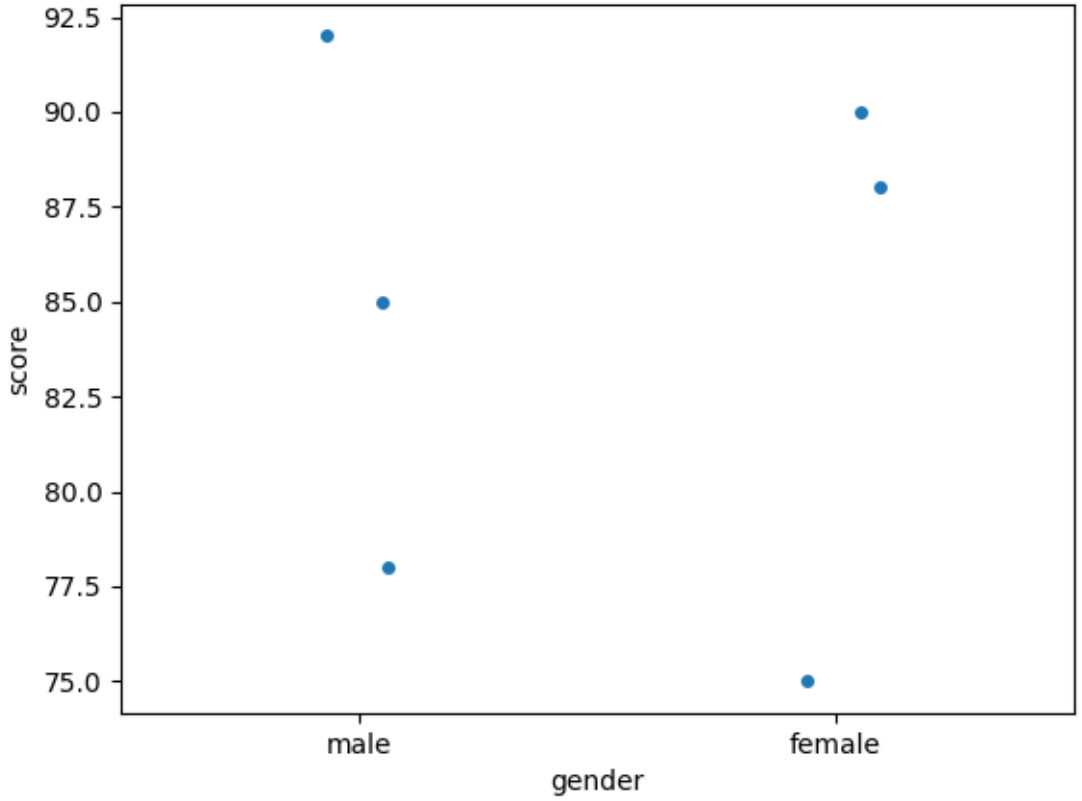
Clustermap

Context: Hierarchical clustering of a sample dataset.

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

data = {
   'A': [1, 2, 3, 4, 5],
   'B': [5, 4, 3, 2, 1],
   'C': [1, 3, 2, 5, 4],
   'D': [2, 3, 4, 5, 6]
}
df = pd.DataFrame(data)

sns.clustermap(df, annot=True)
plt.show()
```



Python Visualization

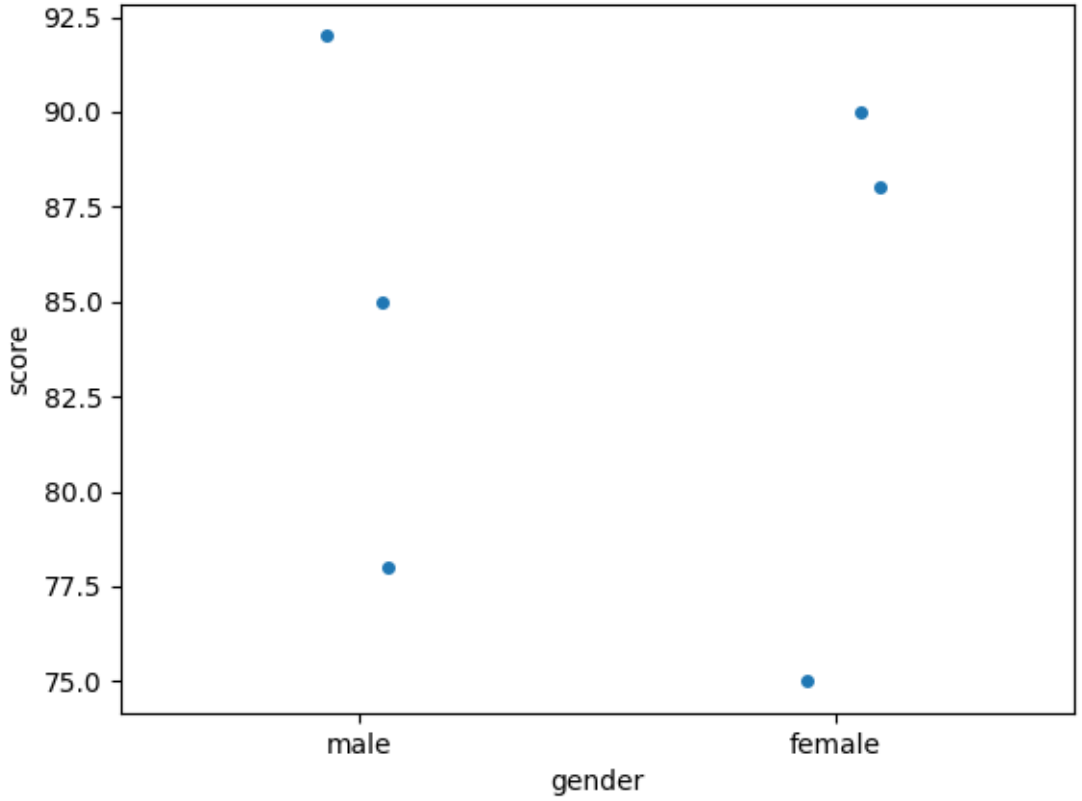
Strip Plot

Context: Hierarchical clustering of a sample dataset.

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

data = {
   'A': [1, 2, 3, 4, 5],
   'B': [5, 4, 3, 2, 1],
   'C': [1, 3, 2, 5, 4],
   'D': [2, 3, 4, 5, 6]
}
df = pd.DataFrame(data)

sns.clustermap(df, annot=True)
plt.show()
```



Python Visualization

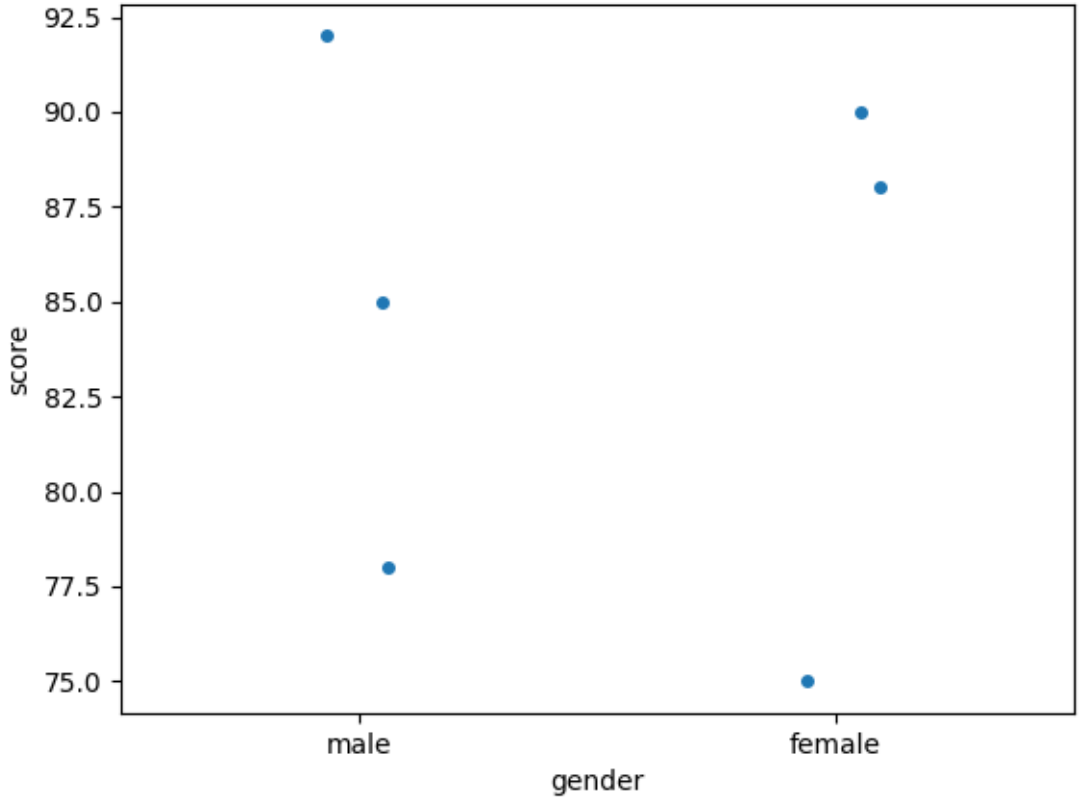
Strip Plot

Context: Hierarchical clustering of a sample dataset.

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

data = {
   'A': [1, 2, 3, 4, 5],
   'B': [5, 4, 3, 2, 1],
   'C': [1, 3, 2, 5, 4],
   'D': [2, 3, 4, 5, 6]
}
df = pd.DataFrame(data)

sns.clustermap(df, annot=True)
plt.show()
```



Python Visualization

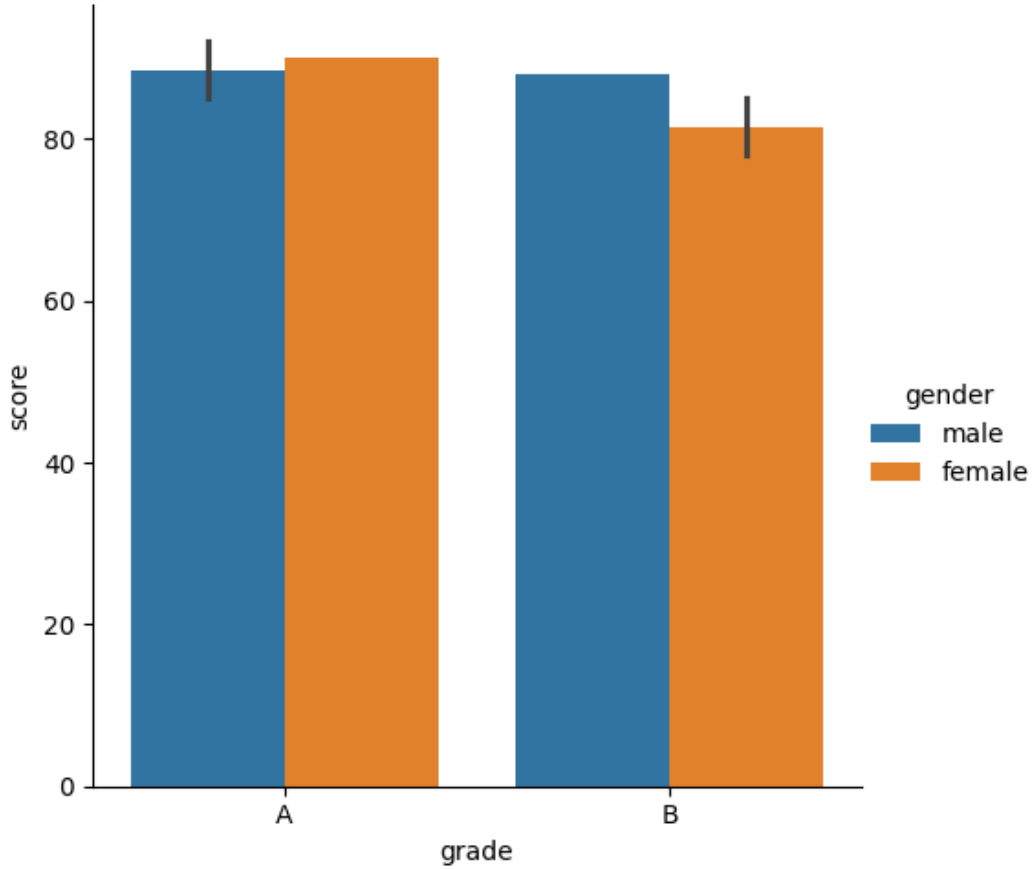
Factor Plot

Context: Average test scores by grade and gender.

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

data = {
   'grade': ['A', 'B', 'A', 'B', 'A', 'B'],
   'gender': ['male', 'male', 'female', 'female', 'male',
'female'],
   'score': [85, 88, 90, 85, 92, 78]
}
df = pd.DataFrame(data)

sns.catplot(x='grade', y='score', hue='gender', data=df,
kind='bar')
plt.show()
```



Python Visualization

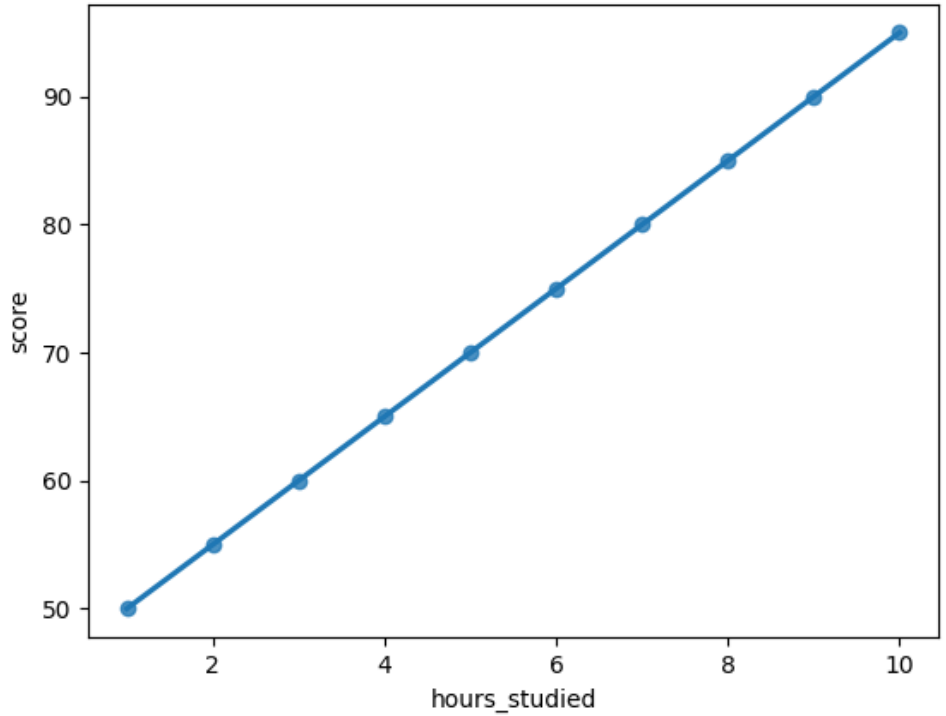
Reg Plot

Context: Relationship between hours studied and exam score.

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

data = {
   'hours_studied': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
   'score': [50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
}
df = pd.DataFrame(data)

sns.regplot(x='hours_studied', y='score', data=df)
plt.show()
```



Python Visualization

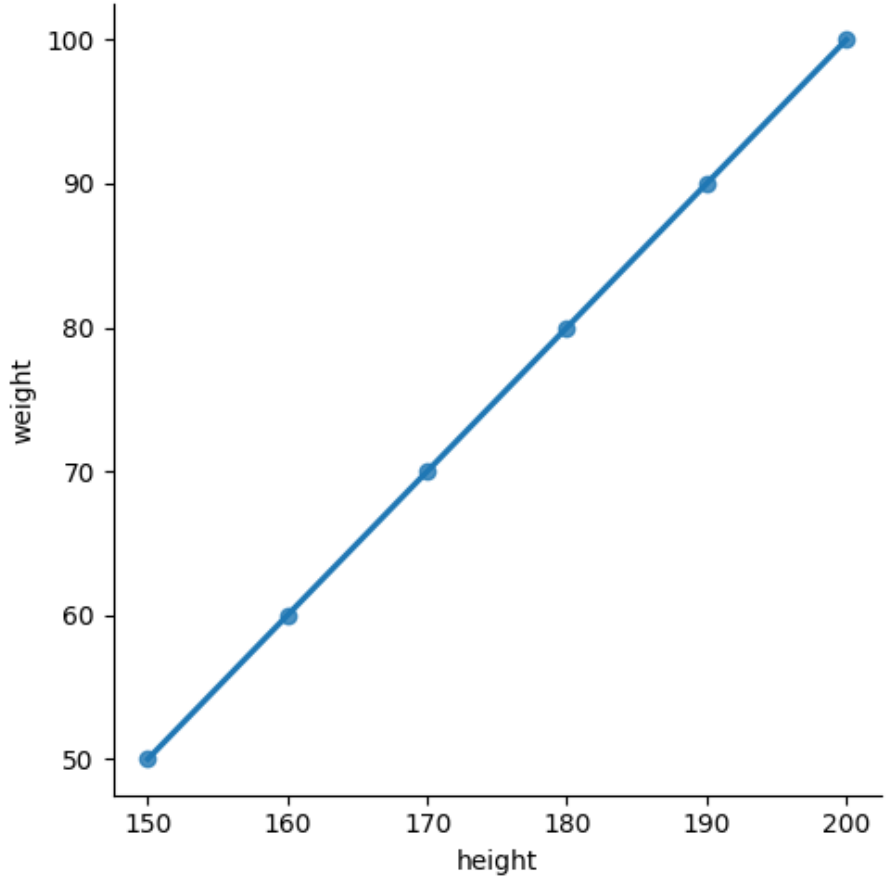
LM Plot

Context: Linear regression plot for weight vs. height.

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

data = {
   'height': [150, 160, 170, 180, 190, 200],
   'weight': [50, 60, 70, 80, 90, 100]
}
df = pd.DataFrame(data)

sns.lmplot(x='height', y='weight', data=df)
plt.show()
```



Python Visualization

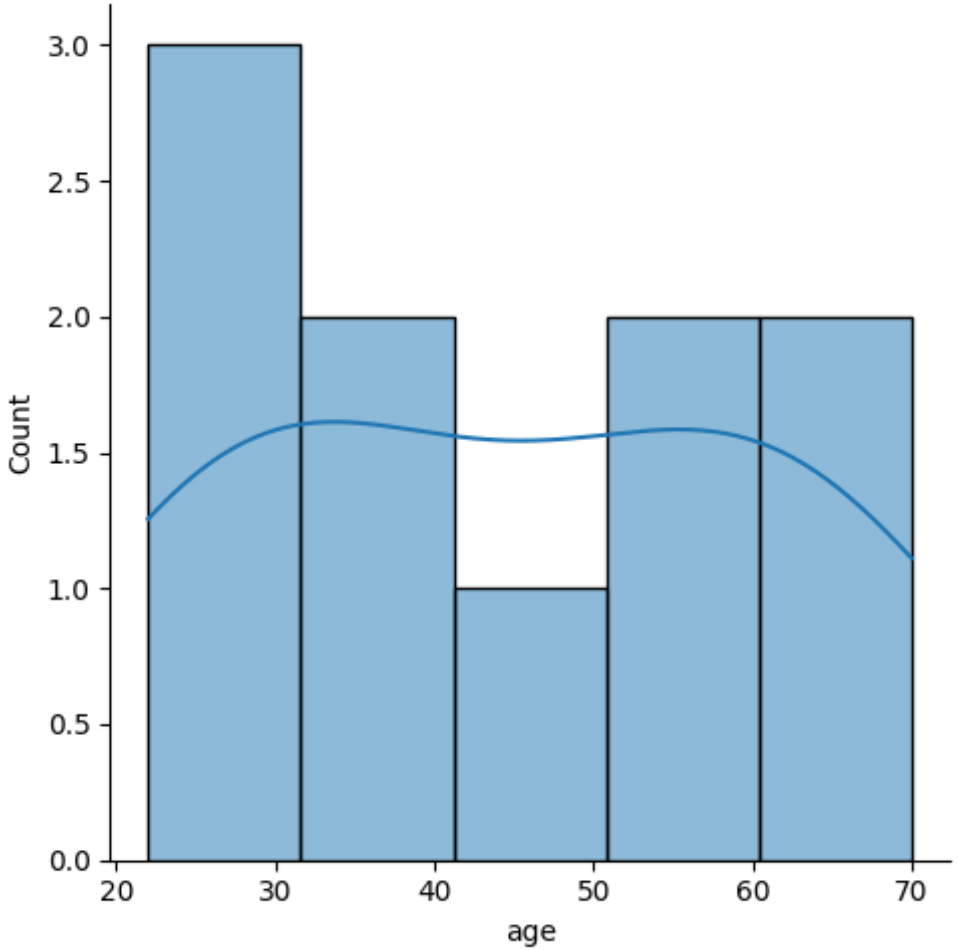
Dis Plot

Context: Distribution of ages in a sample dataset.

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

data = {
   'age': [22, 25, 29, 35, 40, 50, 55, 60, 65, 70]
}
df = pd.DataFrame(data)

sns.displot(df['age'], kde=True)
plt.show()
```



Python Visualization

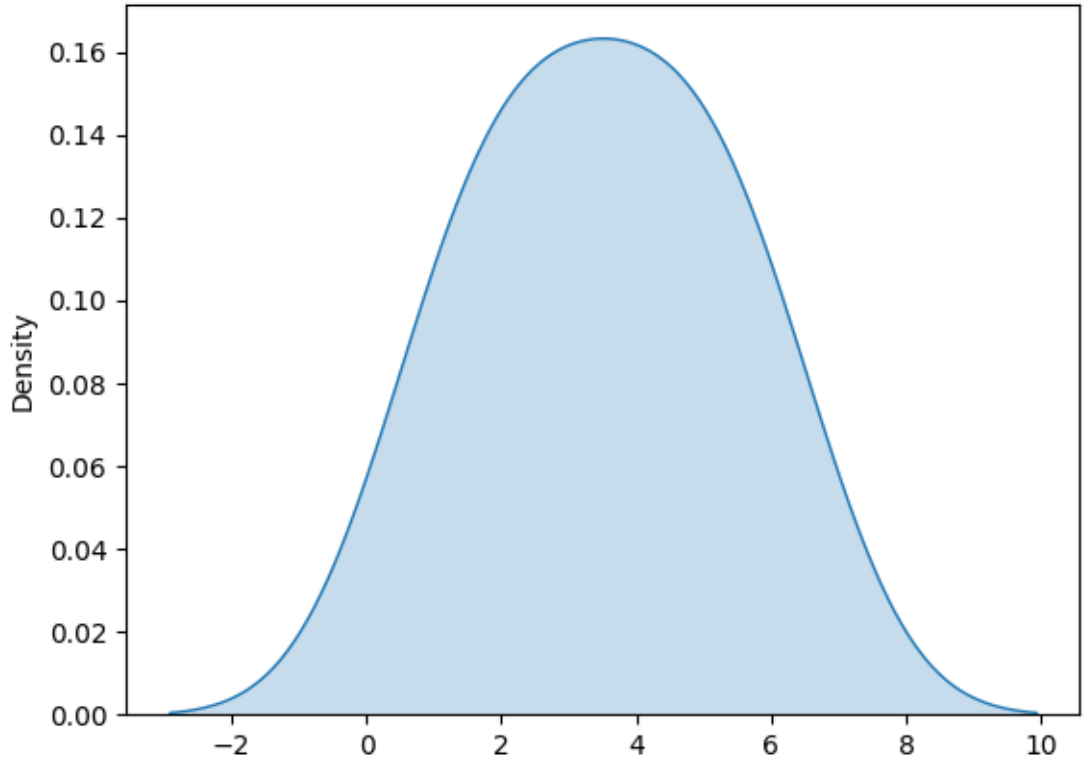
Kdeplot

Context: KDE plot of a sample data.

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

data = {
    'value': [1, 2, 2, 3, 3, 3, 4, 4, 5, 6]
}
df = pd.DataFrame(data)

sns.kdeplot(df['value'], shade=True)
plt.show()
```



Python Visualization

Rug Plot

Context: Visualizing distribution of a sample data with rug plot.

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

data = {
    'value': [1, 2, 2, 3, 3, 3, 4, 4, 5, 6]
}
df = pd.DataFrame(data)

sns.rugplot(df['value'])
plt.show()
```

Python Visualization

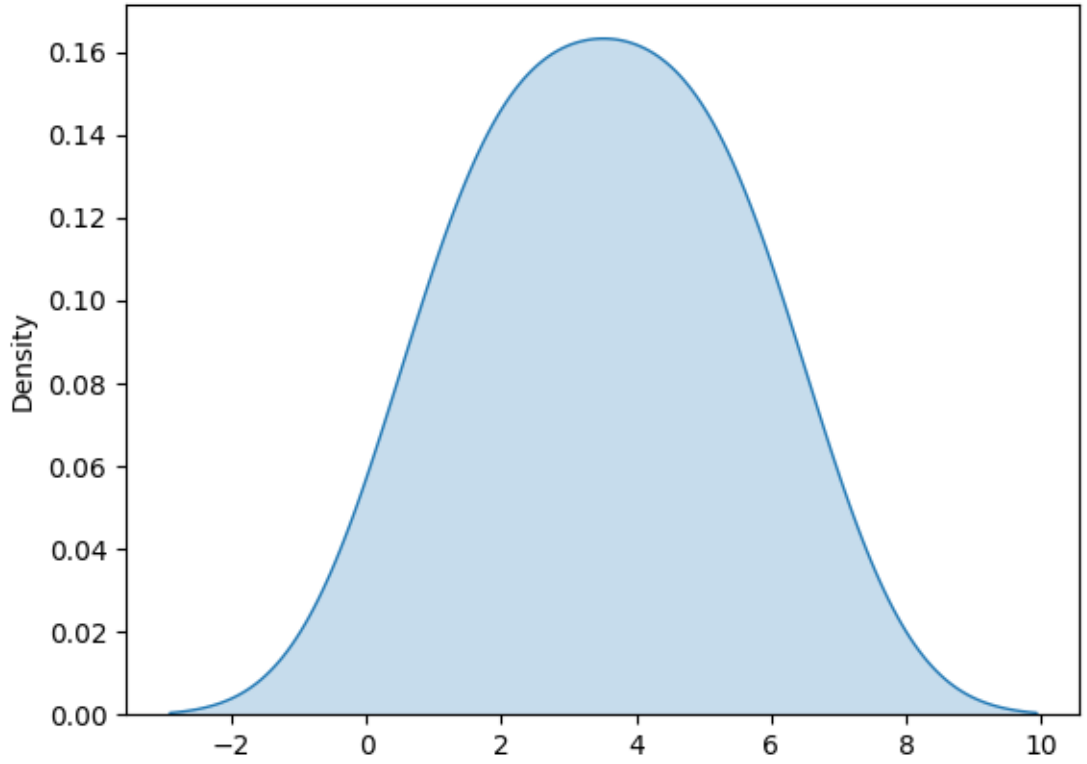
Kdeplot

Context: KDE plot of a sample data.

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

data = {
    'value': [1, 2, 2, 3, 3, 3, 4, 4, 5, 6]
}
df = pd.DataFrame(data)

sns.kdeplot(df['value'], shade=True)
plt.show()
```



Python Visualization

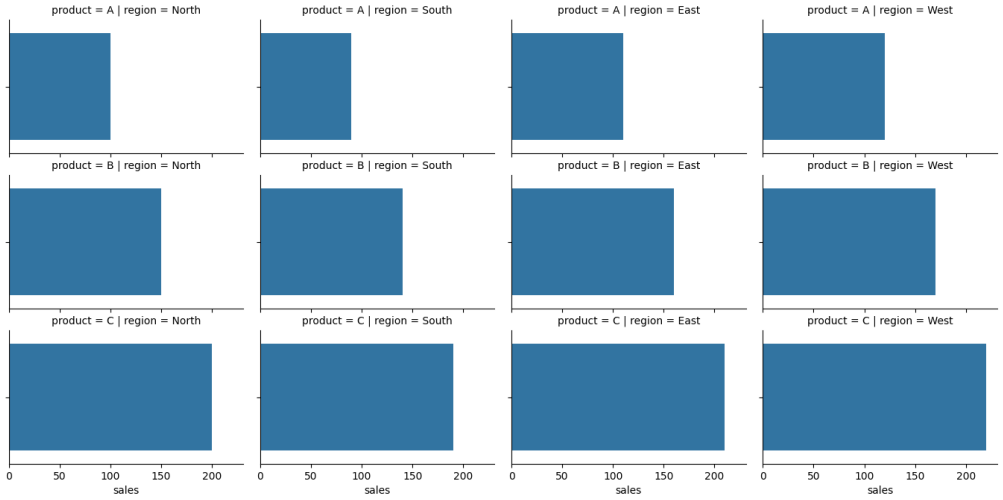
FacetGrid

Context: Facet grid of sample data for sales by region and product type.

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

data = {
   'region': ['North', 'North', 'North', 'South', 'South', 'South',
'East', 'East', 'East', 'West', 'West', 'West'],
   'product': ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C'],
   'sales': [100, 150, 200, 90, 140, 190, 110, 160, 210, 120,
170, 220]
}
df = pd.DataFrame(data)

g = sns.FacetGrid(df, col="region", row="product")
g.map(sns.barplot, "sales")
plt.show()
```



Python Visualization

Summary of Matplotlib

- **Key Takeaways from Matplotlib Sections:**
  - **Versatility:** Matplotlib can create a wide range of plots.
  - **Customization:** Highly customizable to meet specific needs.
  - **Integration:** Works seamlessly with other Python libraries like NumPy and Pandas.
- **Best Practices and Tips:**
  - Start with basic plots and gradually explore advanced features.
  - Use the extensive documentation and community resources for learning.
  - Practice creating different types of plots to become proficient.

Python Visualization

Summary of Matplotlib

- **Key Takeaways from Seaborn Sections:**
  - **Ease of Use:** Simplifies the creation of complex statistical plots.
  - **Attractive Plots:** Provides aesthetically pleasing default styles and color palettes.
  - **Integration with Pandas:** Works efficiently with Pandas data structures.
- **Best Practices and Tips:**
  - Leverage Seaborn's simplicity for quick and effective data visualization.
  - Combine Seaborn and Matplotlib for enhanced customization.
  - Explore Seaborn's rich documentation and examples to master different plot types.

Python Visualization

Best Practices in Data Visualization

- **Tips for Effective Data Visualization:**
  - **Clarity:** Ensure your plots are easy to understand.
  - **Accuracy:** Represent data accurately without misleading visual elements.
  - **Context:** Provide necessary context, labels, and legends.
  - **Aesthetics:** Use color and style thoughtfully to enhance readability.
- **Common Pitfalls and How to Avoid Them:**
  - **Overcomplicating Plots:** Keep it simple and focused.
  - **Inconsistent Scales:** Use consistent scales to avoid confusion.
  - **Neglecting Labels:** Always label axes and provide legends.

Thank You !!!