MATPLOTLIB AND PLOTLY FOR DATA VISUALIZATION

TABLE CONTENT

1. Introduction
2. Library Overviews
    - Matplotlib
    - Plotly
3. Graph Types
    - Matplotlib Graphs
        - Line Plot
        - Scatter Plot
        - Bar Chart
        - Histogram
        - Pie Chart
    - Plotly Graphs
        - Line Plot
        - Scatter Plot
        - Bar Chart
        - Histogram
        - Pie Chart
4. Comparison
5. Conclusion

INTRODUCTION

Data visualization is essential in providing assistance for data scientists, analysts, and engineers to understand big datasets. This guide compares two widely-used Python visualization libraries: **Matplotlib** and **Plotly**, drawing on the official documentation for these libraries to provide accurate examples and insights.

LIBRARY OVERVIEWS

MATPLOTLIB

Matplotlib, as outlined in the official documentation, is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is the bedrock of Python plotting and is well-suited for creating publication-quality graphs with detailed customization.

Key Features:

- Rich API to create a lot of visualizations
- Deeply integrated with libraries such as Pandas, NumPy, and SciPy.
- Saving figures in high-resolution (PNG, SVG, PDF)
- Good for static plots (reports and academic papers)

PLOTLY

Plotly is a powerful library for creating interactive and web-based visualizations. The library allows users to create dynamic plots that can be integrated into dashboards and web applications. Plotly supports various chart types and interactive features out-of-the-box, making it a valuable tool for exploratory data analysis and dashboards.

Key Features:

- Web-based interactive visualizations.
- Built-in support for plot interactivity (zooming, panning, hovering).
- Ability to handle large datasets with smooth rendering.
- Strong integration with Dash for building analytical applications.

GRAPH TYPES

MATPLOTLIB GRAPHS

LINE PLOT

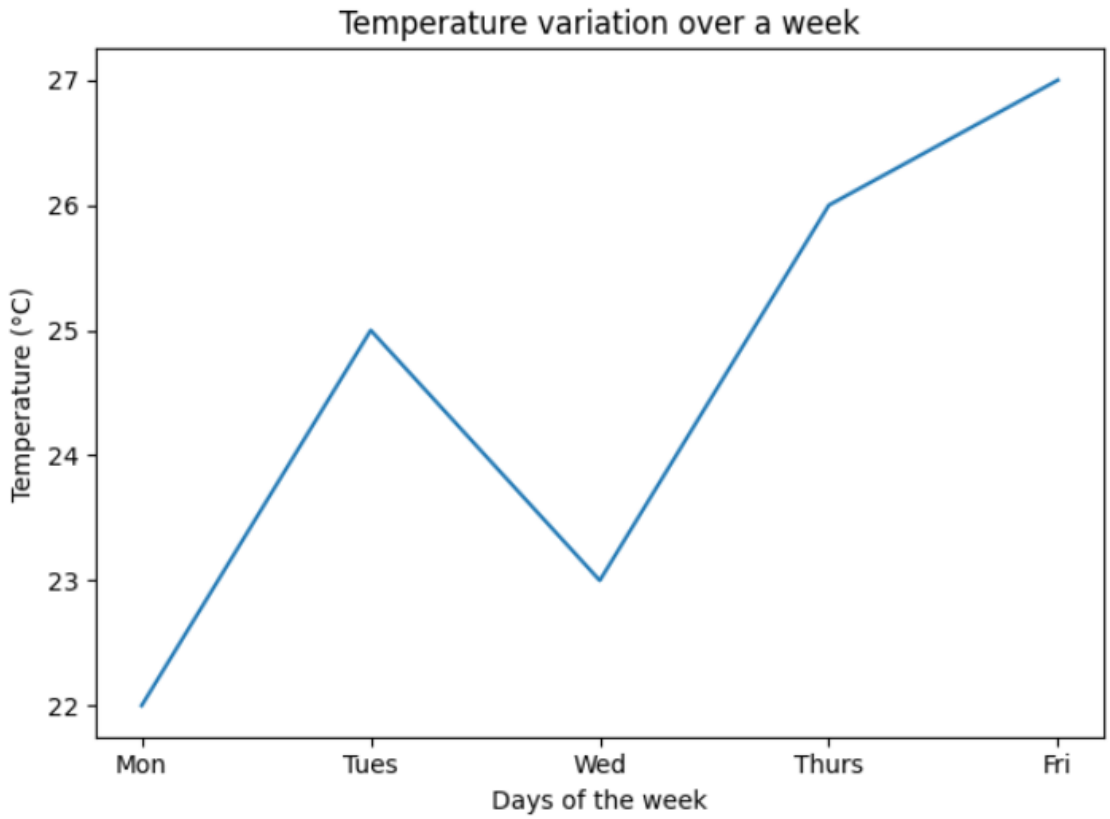
Description: A line plot is ideal for displaying continuous data over time, connecting data points with straight lines. This plot type is widely used for trend analysis.

Use Case: Tracking the daily temperature over a month.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py

```python
import matplotlib.pyplot as plt

# Sample data
days = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri']
temperature = [22, 25, 23, 26, 27]

# Create line plot
plt.plot(days, temperature)

# Add labels and title
plt.xlabel('Day of the Week')
plt.ylabel('Temperature (°C)')
plt.title('Temperature Variation Over a Week')

# Show the plot
plt.show()

```



SCATTER PLOT

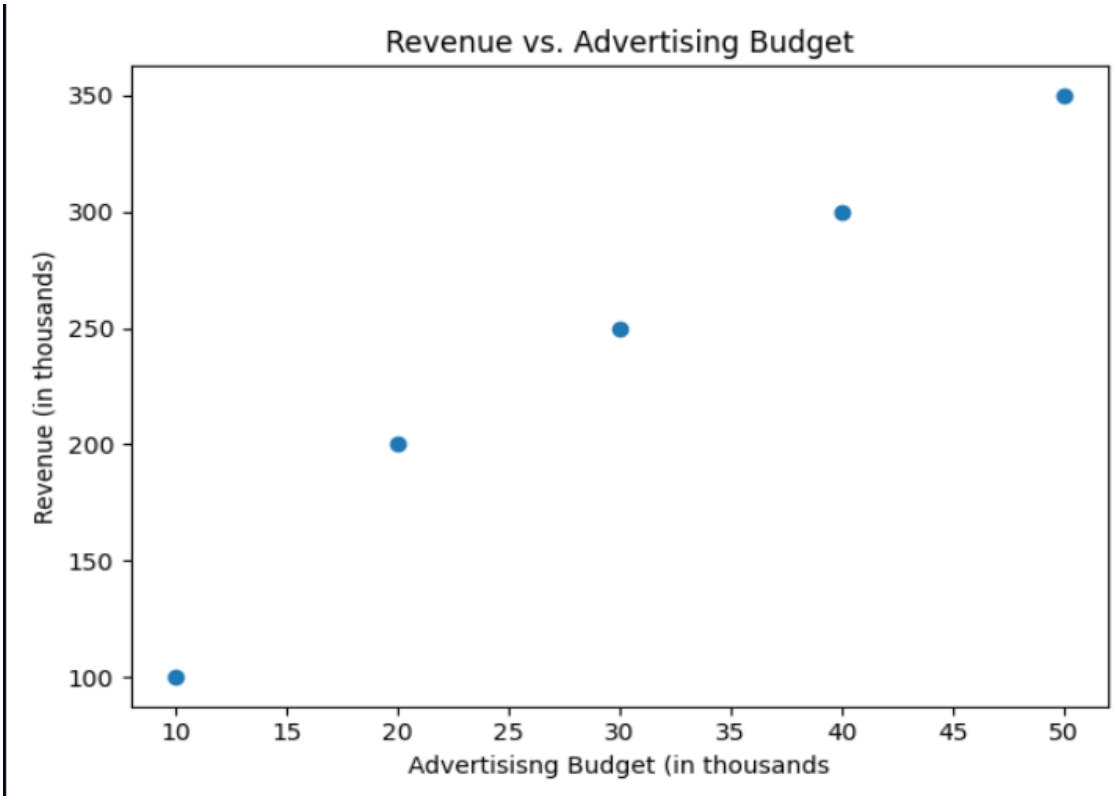
Description: Scatter plots are used to represent individual data points on a 2D plane. It is a great way to visualize relationships between variables.

Use Case: Examining the relationship between advertising budgets and revenue.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py

```python
import matplotlib.pyplot as plt

# Sample data
ad_budget = [10, 20, 30, 40, 50]
revenue = [100, 200, 250, 300, 350]

# Create scatter plot
plt.scatter(ad_budget, revenue)

# Add labels and title
plt.xlabel('Advertising Budget (in thousands)')
plt.ylabel('Revenue (in thousands)')
plt.title('Revenue vs Advertising Budget')

# Show the plot
plt.show()

```



BAR CHART

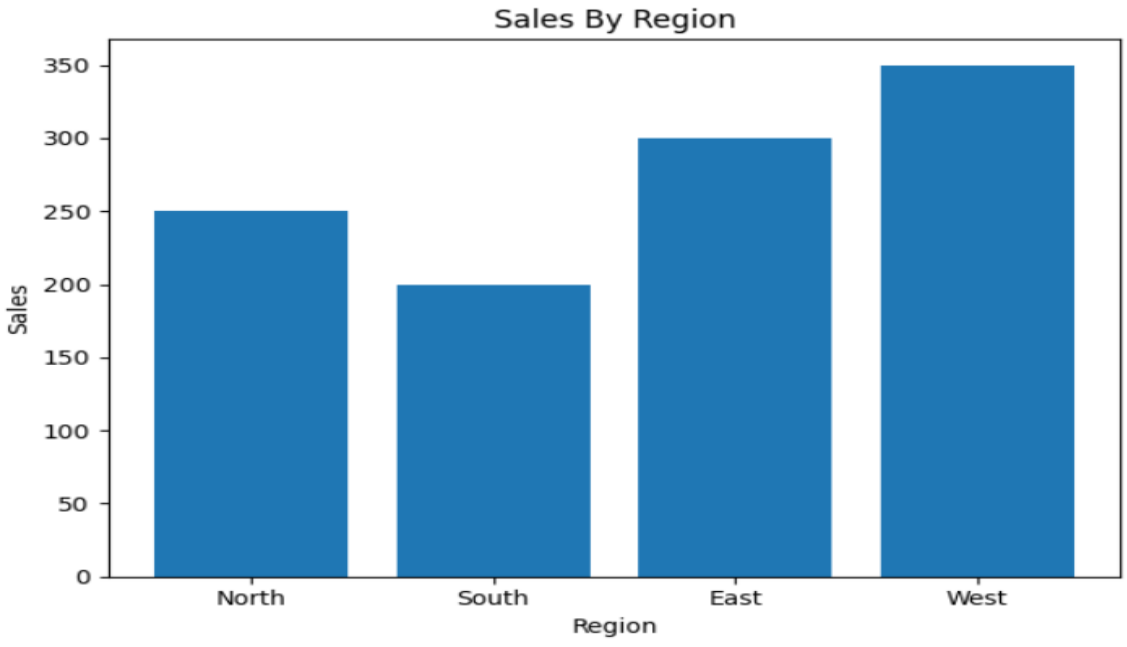
Description: Bar charts are used to compare different categories using rectangular bars. The height of each bar corresponds to the category's value.

Use Case: Comparing sales across different regions.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py

```python
import matplotlib.pyplot as plt

# Sample data
regions = ['North', 'South', 'East', 'West']
sales = [250, 200, 300, 150]

# Create bar chart
plt.bar(regions, sales)

# Add labels and title
plt.xlabel('Region')
plt.ylabel('Sales')
plt.title('Sales by Region')

# Show the plot
plt.show()

```



HISTOGRAM

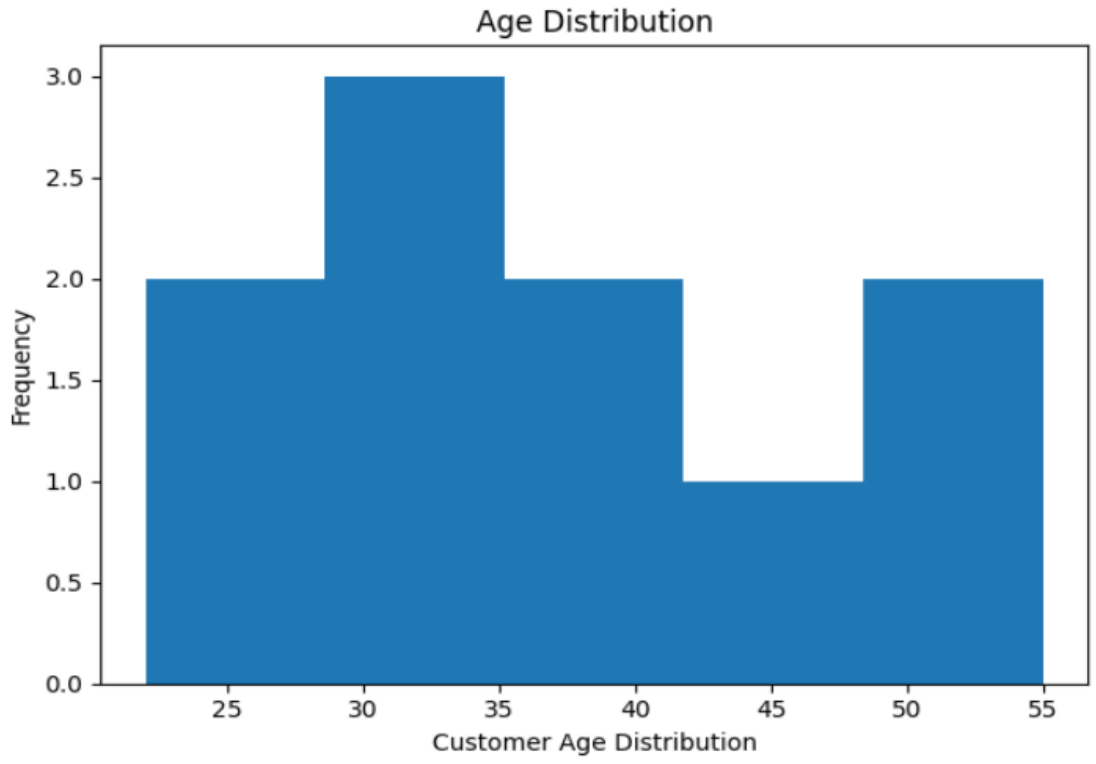
Description: Histograms show the frequency distribution of a continuous variable by splitting data into intervals (bins).

Use Case: Visualizing the distribution of customer ages in a store.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py

```python
import matplotlib.pyplot as plt

# Sample data
ages = [22, 25, 29, 32, 35, 37, 40, 45, 50, 55]

# Create histogram
plt.hist(ages, bins=5)

# Add labels and title
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Customer Age Distribution')

# Show the plot
plt.show()

```



PIE CHART

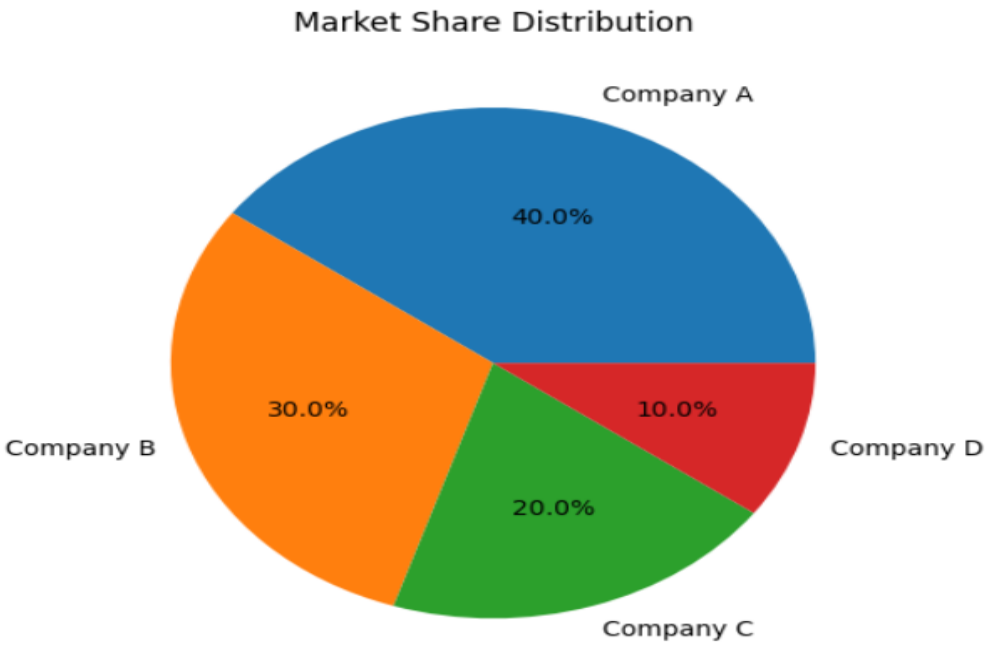
Description: A pie chart displays proportions of categories in a dataset as slices of a circle.

Use Case: Displaying market share distribution among competitors.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py

```python
import matplotlib.pyplot as plt

# Sample data
companies = ['Company A', 'Company B', 'Company C', 'Company D']
market_share = [40, 30, 20, 10]

# Create pie chart
plt.pie(market_share, labels=companies, autopct='%1.1f%%', startangle=90)

# Add title
plt.title('Market Share Distribution')

# Show the plot
plt.show()

```



PLOTLY GRAPHS
Line Plot

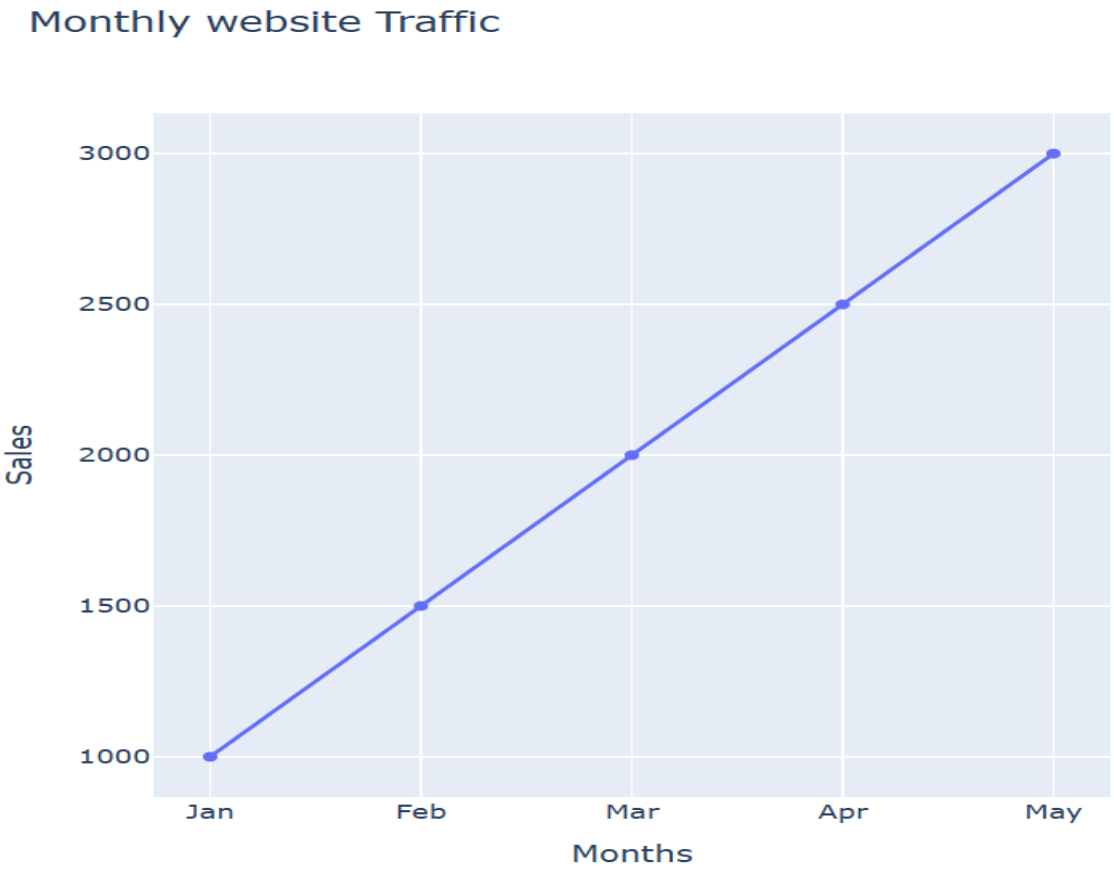
Description: A line plot in Plotly is fully interactive, allowing users to hover over data points and explore trends.

Use Case: Tracking monthly website traffic.

Code Snippet:.

C: > Users > Admin > Desktop > Code snippet.py

```python
import plotly.express as px

# Sample data
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']
visitors = [1000, 1500, 2000, 2500, 3000]

# Create interactive line plot
fig = px.line(x=months, y=visitors, title='Monthly Website Traffic')
fig.show()

```



Scatter Plot

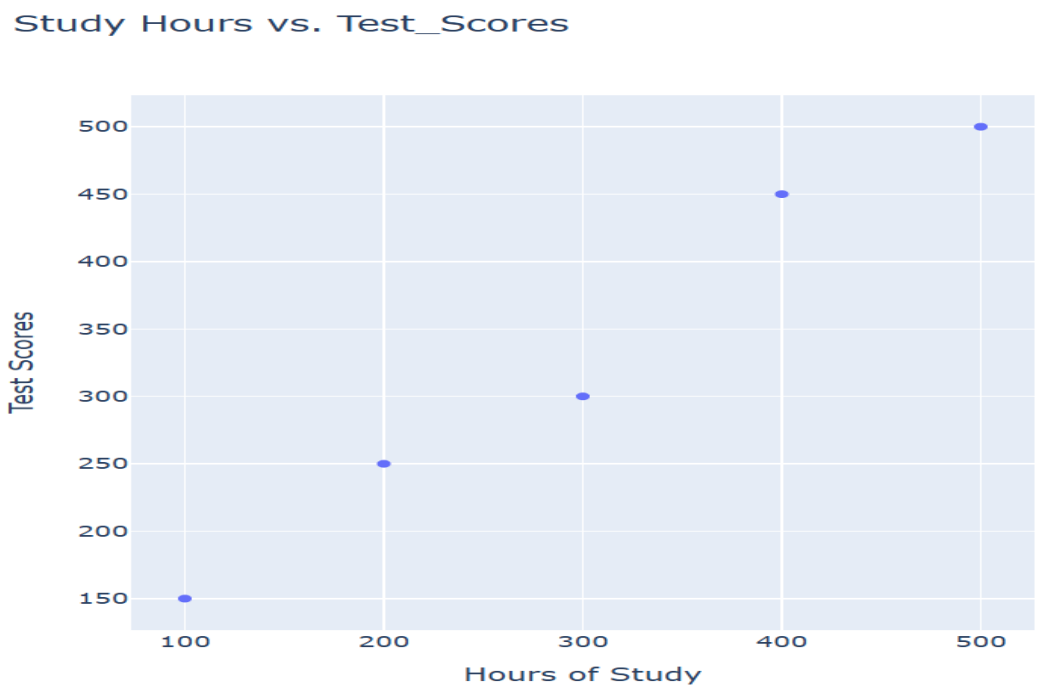
Description: Interactive scatter plots allow users to see the exact value of each data point when hovering.

Use Case: Analyzing the relationship between study hours and exam scores.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py

```python
import plotly.express as px

# Sample data
hours_studied = [1, 2, 3, 4, 5]
test_scores = [50, 60, 70, 80, 90]

# Create scatter plot
fig = px.scatter(x=hours_studied, y=test_scores, title='Study Hours vs Test Scores')
fig.show()

```



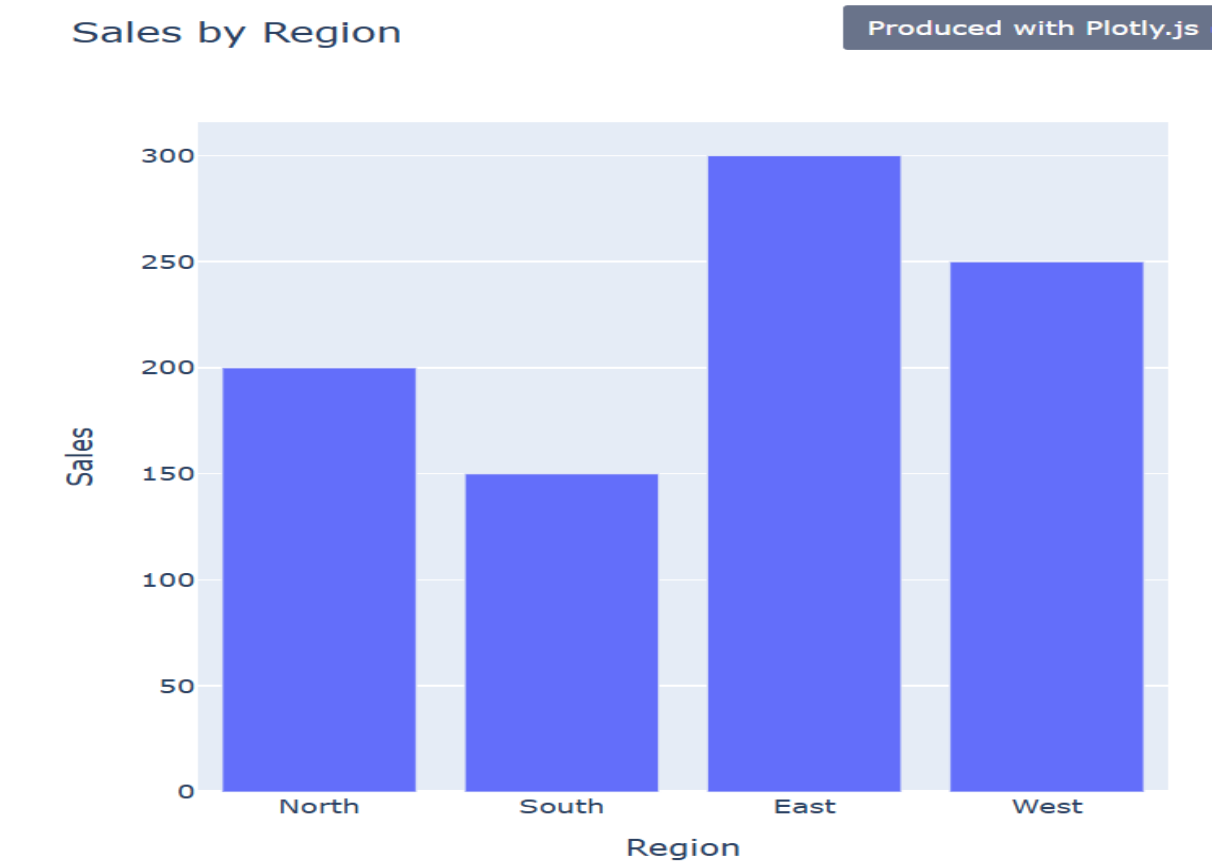
Bar Chart

Description: A bar chart displays categorical data with rectangular bars. Each bar's length is proportional to the data it represents, making it ideal for comparing different categories.

Use Case: Compare the total bill by day of the week in a restaurant.

Code Snippet:

```
C: > Users > Admin > Desktop > Code snippet.py > ...
1   import plotly.express as px
2
3   # Sample data from Plotly
4   df = px.data.tips()
5
6   # Create bar chart
7   fig = px.bar(df, x='day', y='total_bill', color='sex', title='Total Bill by Day and Gender')
8   fig.show()
9
```



Histogram (with Distplot)

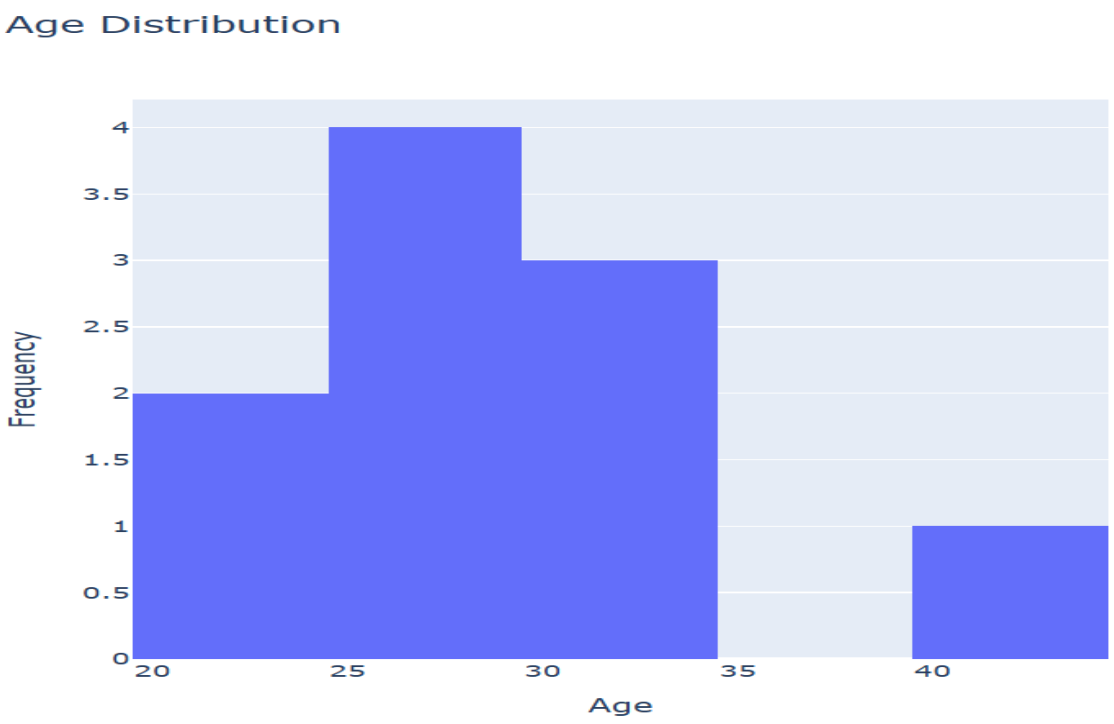
Description: A histogram is a graphical representation of the distribution of numerical data. It's useful for understanding the frequency distribution within a dataset. When combined with `distplot`, you can overlay the kernel density estimation (KDE) to smooth the distribution.

Use Case: Analyze the distribution of total bills in a restaurant with a histogram.

Code Snippet :

C: > Users > Admin > Desktop > Code snippet.py > ...

```python
import plotly.figure_factory as ff
import numpy as np

# Sample data
x = np.random.randn(1000)

# Create distplot (histogram + KDE)
fig = ff.create_distplot([x], group_labels=['Total Bills'], bin_size=0.2)
fig.update_layout(title='Total Bill Distribution')
fig.show()

```



Pie Chart

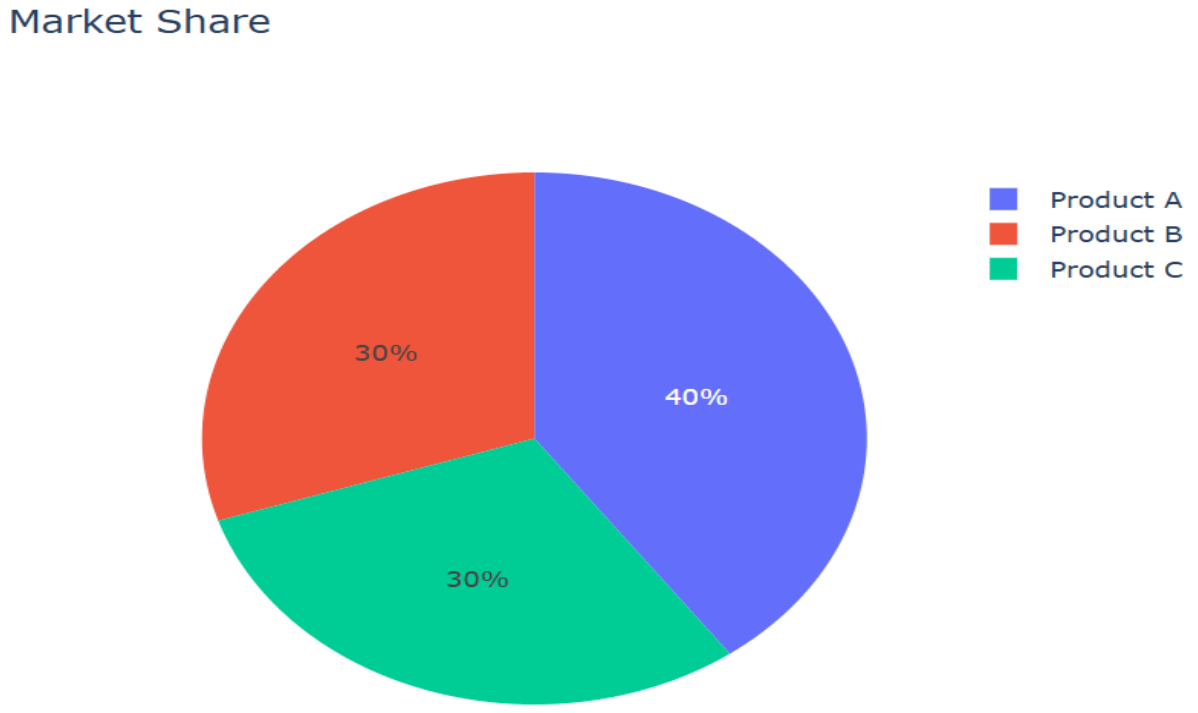
Description: A pie chart represents data as slices of a circle, where each slice's size is proportional to its percentage of the whole. It is ideal for showing part-to-whole relationships.

Use Case: Display the total bill share by day in a restaurant.

Code Snippet:

C: > Users > Admin > Desktop > Code snippet.py > ...

```python
import plotly.express as px

# Sample data from Plotly
df = px.data.tips()

# Create pie chart
fig = px.pie(df, values='total_bill', names='day', title='Total Bill Share by Day')
fig.show()

```



COMPARISON

ASPECT	MATPLOTLIB	PLOTLY
EASE OF USE	Requires more code for basic plots, but highly customizable.	Easy-to-use API for interactive plots with minimal code.
CUSTOMIZATION	Full control over plot elements, but requires detailed configuration.	Extensive interactivity with built-in features, but some limitations on styling.
INTERACTIVITY	Mainly static, though interactivity is possible via additional tools.	Highly interactive out-of-the-box, supporting zoom, pan, and hover.
PERFORMANCE	Efficient with large datasets for static plots.	Can handle large datasets but may slow down with complex interactivity.
BEST USE CASE	Ideal for creating static plots for reports or publications.	Excellent for interactive, web-based visualizations and dashboards.

CONCLUSION

In summary, **Matplotlib** is ideal for creating detailed, static plots, especially for reports and publications, while **Plotly** excels in interactive, web-based visualizations. Choose **Matplotlib** for precision and customization, and **Plotly** when interactivity and ease of sharing are key.