Data visualization

Data visualization is the process of representing data graphically to help understand patterns, trends, and insights. Python offers several powerful libraries for creating a wide range of visualizations.

Popular Data Visualization Libraries in Python

- Matplotlib: The foundational library for static, animated, and interactive plots. Supports line plots, scatter plots, bar charts, histograms, pie charts, heatmaps, and more.
- Seaborn: Built on Matplotlib, provides a high-level interface for attractive and informative statistical graphics such as categorical plots, distribution plots, and correlation plots.
- Plotly: Interactive plotting library for web-based visualization, supporting complex dashboards and dynamic charts.
- Bokeh: For creating interactive visualizations that can be used in web browsers.
- Pandas Visualization: Pandas integrates simple plotting based on Matplotlib directly on DataFrames and Series for quick visualization.

Common Visualization Types

- Line Plot: Showing trends over time or ordered categories.
- Scatter Plot: Visualizing relationships between two variables.
- Bar Chart: Comparing quantities across categories.
- Histogram: Distribution of numeric data.
- Box Plot: Summary of data distribution highlighting medians, quartiles, and outliers.
- Heatmap: Displaying data matrices with color coding.
- Pie Chart: Showing proportions in a whole.
- Violin Plot, Swarm Plot: Advanced categorical visualizations (Seaborn).

Example: Simple Line Plot using Matplotlib

python

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

x = np.arange(1, 11)
y = np.random.randint(1, 10, size=10)

plt.plot(x, y, marker='o', linestyle='-', color='blue')
plt.title('Sample Line Plot')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.grid(True)
plt.show()
```

Example: Categorical Bar Plot using Seaborn

python

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

tips = sns.load_dataset("tips")
sns.barplot(x="day", y="total_bill", data=tips)
plt.show()
```

Plotting data with Matplotlib

Plotting data with Matplotlib, a foundational Python library for creating static, animated, and interactive visualizations, involves a few key steps using its pyplot interface.

Basic Steps for Plotting with Matplotlib

1. Import the library:

python

```python
import matplotlib.pyplot as plt
```

2. Prepare your data: This can be lists, NumPy arrays, or Pandas Series/DataFrames.
3. Create a plot: Use functions like plt.plot() for line plots, plt.scatter() for scatter plots, etc.
4. Customize the plot: Add titles, labels, legends, grid, colors.
5. Display the plot: Use plt.show() to render the figure.

Example: Simple Line Plot

python

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

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

plt.plot(x, y, label='sin(x)', color='blue', linestyle='-')
plt.title('Sine Wave')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.legend()
plt.grid(True)
plt.show()
```

Other Common Plot Types

Scatter Plot:

python

```python
plt.scatter(x, y, color='red')
```

Bar Chart:

python

```python
categories = ['A', 'B', 'C']
values = [10, 15, 7]
plt.bar(categories, values)
```

Histogram:

python

```python
data = np.random.randn(1000)
plt.hist(data, bins=30, color='green')
```

Customization Options

- Change colors, line styles, markers.
- Add multiple plots to the same figure.
- Adjust axis limits: plt.xlim(), plt.ylim().

- Add annotations and text.
- Save figures with plt.savefig('filename.png').