VISUALIZATION

Visualization is a way of representing. Data visualization is representing of data in a graphical or pictorial format for better understanding of data. Visualization gives a good idea of data and the trends in it.

In python, data visualization has multiple libraries like matplotlib, seaborn, plotly etc.

Let us dive deeper into pandas and matplotlib.

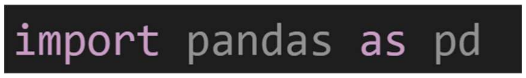
Pandas

1. Pandas was created in **2008** by **Wes McKinney** at AQR Capital.
2. It was built to handle **financial time series data** efficiently.
3. The name **Pandas** comes from **"panel data"** and **Python data analysis**.
4. Released as an **open-source library** in 2009–2010.
5. Introduced core structures: **Series** (1D) and **DataFrame** (2D).
6. Gained popularity for **data cleaning, manipulation, and analysis**.
7. Became part of the **PyData ecosystem** (with NumPy, Matplotlib, scikit-learn).
8. Supported by **NumFOCUS**, ensuring community-driven growth.
9. Performance improved with **Cython/C extensions** for speed.
10. Today, Pandas is a **core library for data science, ML, and analytics** worldwide.

By default, Pandas supports 10 main plot types (through DataFrame.plot(kind=...) and Series.plot(...)).
These are:

1. **line** → df.plot.line()
2. **bar** → df.plot.bar()
3. **barh** → df.plot.barh()
4. **hist** → df.plot.hist()
5. **box** → df.plot.box()
6. **kde / density** → df.plot.kde()
7. **area** → df.plot.area()
8. **scatter** → df.plot.scatter(x, y)
9. **hexbin** → df.plot.hexbin(x, y)
10. **pie** → df.plot.pie()

Importing pandas library:



```python
import pandas as pd
```

LINE PLOT

Definition

A **line plot** is a graph that connects individual data points with straight lines to show **trends or changes over continuous data**, such as **time series** or sequential observations.

Data

data = {

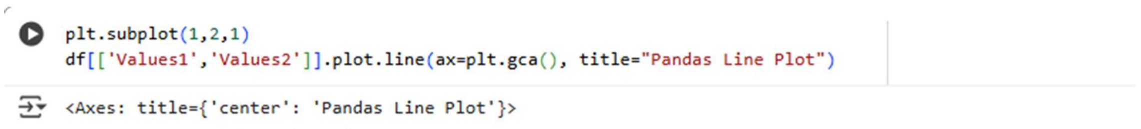
    'Category': ['A', 'B', 'C', 'D'],

    'Values1': [10, 20, 15, 25],

    'Values2': [12, 18, 22, 28]

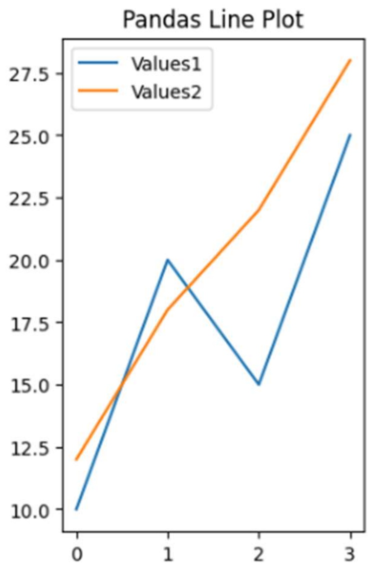
}

df = pd.DataFrame(data)



```python
plt.subplot(1,2,1)
df[['Values1','Values2']].plot.line(ax=plt.gca(), title="Pandas Line Plot")
```

```
<Axes: title={'center': 'Pandas Line Plot'}>
```



Bar Chart

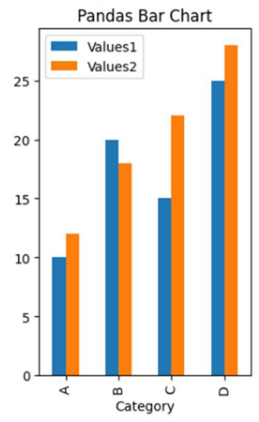
Definition

A **bar chart** is a graph that uses **rectangular bars** to represent data.

- The **length or height** of each bar is **proportional to the value** it represents.
- Bar charts are typically used for **comparing categorical data**.
- Can be **vertical** or **horizontal**.

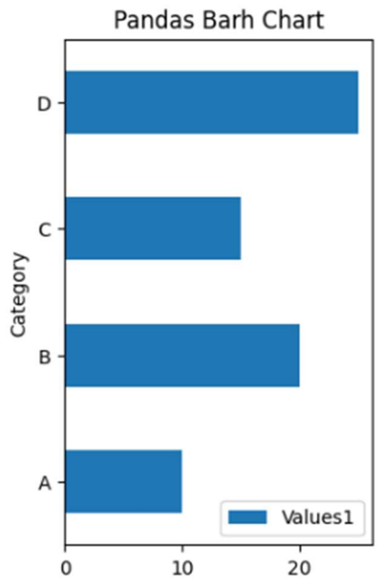
```
plt.subplot(1,2,1)
df.plot.bar(x='Category', y=['Values1','Values2'], ax=plt.gca(), title="Pandas Bar Chart")
```

```
<Axes: title={'center': 'Pandas Bar Chart'}, xlabel='Category'>
```



```
plt.subplot(1,2,1)
df.plot.barh(x='Category', y='Values1', ax=plt.gca(), title="Pandas Barh Chart")
```

```
<Axes: title={'center': 'Pandas Barh Chart'}, ylabel='Category'>
```



Histogram

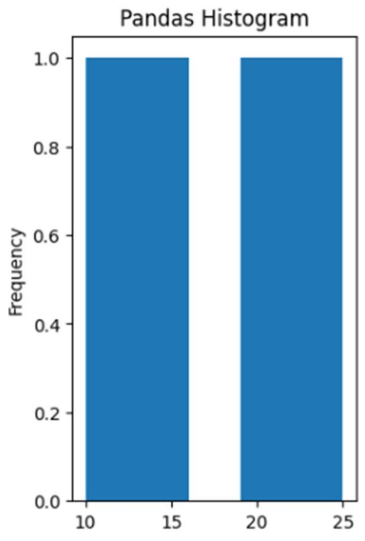
Definition

A **histogram** is a graphical representation of the **distribution of numerical data**.

- Data is divided into **bins (intervals)**, and the **height of each bar** shows how many values fall into that bin.
- Useful for **understanding the frequency, spread, and shape of the data**.
- Different from a bar chart: bars in a histogram **touch each other**, representing continuous data.

```
plt.subplot(1,2,1)
df['Values1'].plot.hist(ax=plt.gca(), bins=5, title="Pandas Histogram")
```

```
<Axes: title={'center': 'Pandas Histogram'}, ylabel='Frequency'>
```



Box plot

A **box plot** is a graphical representation of the **distribution of numerical data** using **five summary statistics**:

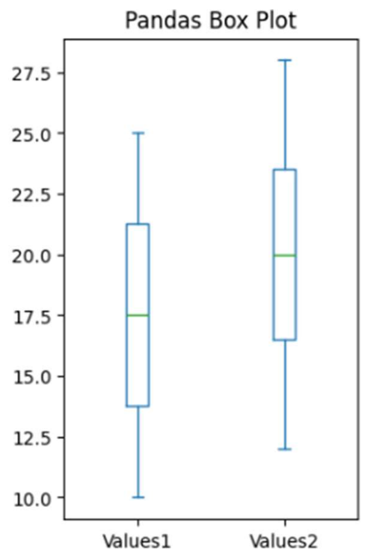
1. **Minimum** (lowest value)
2. **First Quartile (Q1)** – 25th percentile
3. **Median (Q2)** – 50th percentile
4. **Third Quartile (Q3)** – 75th percentile
5. **Maximum** (highest value)

In Pandas

- Use **.plot.box()** on a DataFrame or Series.

```python
plt.subplot(1,2,1)
df[['Values1','Values2']].plot.box(ax=plt.gca(), title="Pandas Box Plot")
```

```
<Axes: title={'center': 'Pandas Box Plot'}>
```



Area Plot

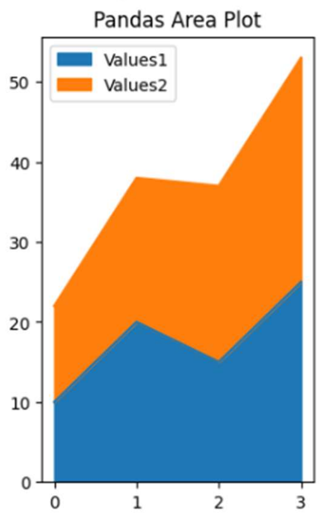
Definition

An **area plot** is similar to a **line plot**, but the area **under the line is filled** with color.

- It is used to **show cumulative trends over time or categories**.
- Helps visualize **part-to-whole relationships** when multiple series are stacked.

```python
plt.subplot(1,2,1)
df[['Values1','Values2']].plot.area(ax=plt.gca(), title="Pandas Area Plot")
```

```
<Axes: title={'center': 'Pandas Area Plot'}>
```

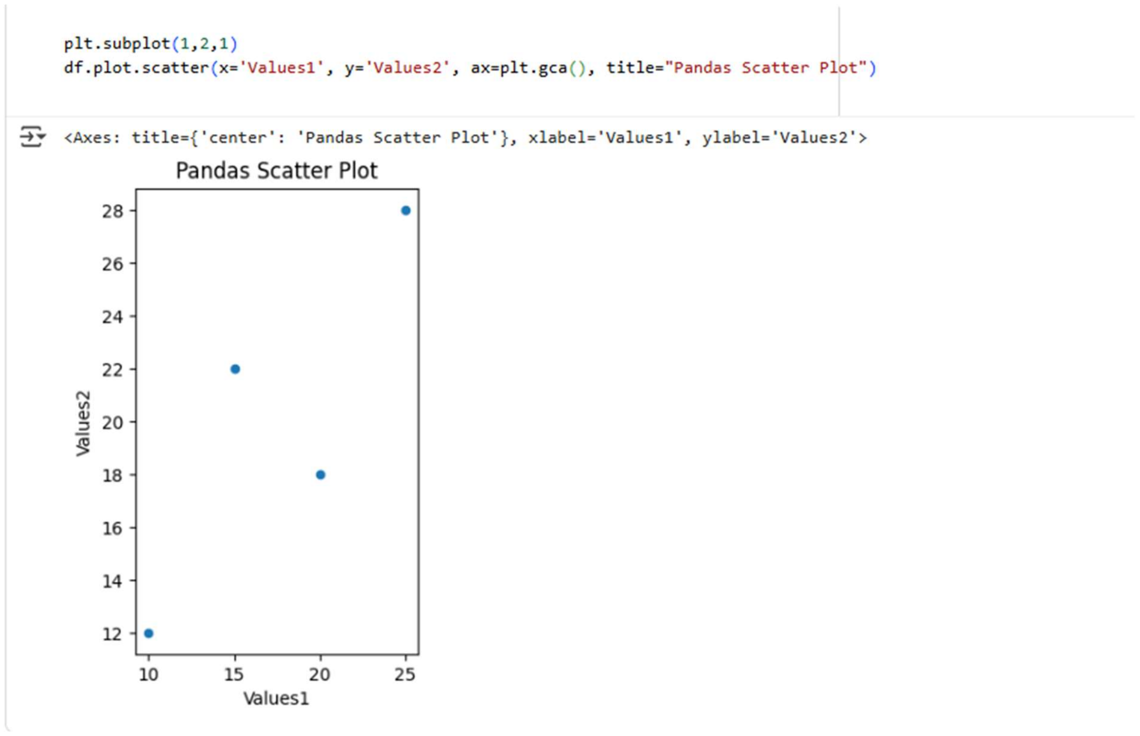


Scatter Plot

Definition

A **scatter plot** is a graph that displays **individual data points** on a 2D plane using **x and y coordinates**.

- It is used to **show the relationship or correlation** between two numerical variables.
- Helps identify **patterns, trends, clusters, and outliers**.



```
plt.subplot(1,2,1)
df.plot.scatter(x='Values1', y='Values2', ax=plt.gca(), title="Pandas Scatter Plot")
```

```
<Axes: title={'center': 'Pandas Scatter Plot'}, xlabel='Values1', ylabel='Values2'>
```

Advantages of Pandas

1. **Easy Data Handling**
   - Handles **tabular data** (rows & columns) efficiently using **DataFrames** and **Series**.
2. **Fast and Efficient**
   - Built on **NumPy**, optimized for performance with large datasets.
3. **Flexible Data Input/Output**
   - Can **read/write data** from CSV, Excel, SQL, JSON, HTML, and more.
4. **Data Cleaning & Preparation**
   - Easily **filter, sort, merge, group, and reshape** data.
5. **Time Series Support**
   - Powerful tools to handle **dates, times, periods, and frequency-based operations**.
6. **Built-in Data Analysis**
   - Provides **statistics, aggregation, and descriptive analysis** functions.

7. **Visualization Integration**
   - Works directly with **Matplotlib** for plotting graphs like line, bar, histogram, etc.
8. **Community Support & Open Source**
   - Large **community**, regular updates, and free to use.

Matplotlib

Matplotlib was created by **John D. Hunter** in **2003**.

It was developed as a **2D plotting library for Python**, similar to MATLAB.

Released as **open-source software** under the Python Software Foundation License.

Aimed to provide **high-quality static, interactive, and animated plots**.

Integrated with **NumPy** for numerical and scientific computing.

Introduced the **pylab interface** to mimic MATLAB-style plotting.

Gained popularity in the **PyData ecosystem** alongside Pandas and SciPy.

Modern updates added **object-oriented interface, 3D plotting, and interactive backends**.

Inspired other libraries like **Seaborn, Plotly, and Bokeh**.

Today, Matplotlib remains a **core library for Python data visualization**.

Importing matplotlib:

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

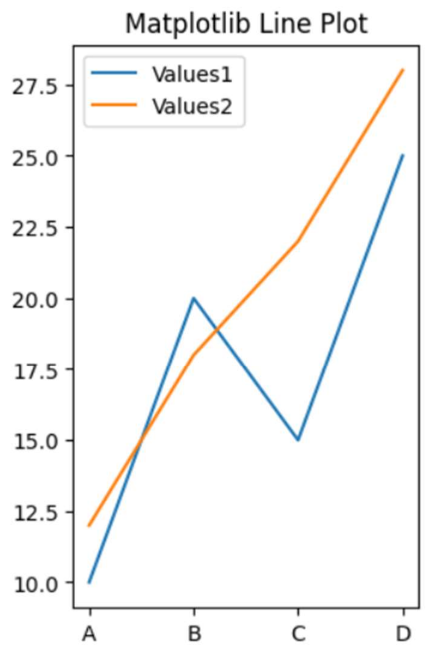
Line Plot

Definition

A **line plot** is a graph that displays **data points connected by straight lines** to show **trends, patterns, or changes over a continuous variable**, such as time or sequence.

- It is commonly used to **visualize time series data**, trends, or comparisons between multiple datasets.
- Matplotlib allows **customization of line style, color, markers, and labels**.

```
plt.subplot(1,2,2)
plt.plot(df['Category'], df['Values1'], label='Values1')
plt.plot(df['Category'], df['Values2'], label='Values2')
plt.title("Matplotlib Line Plot")
plt.legend()
plt.show()
```



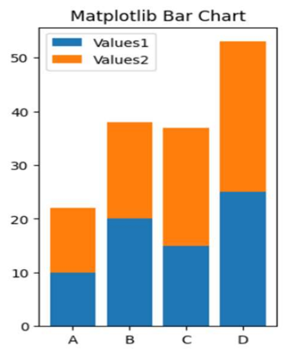
Bar chart

Definition

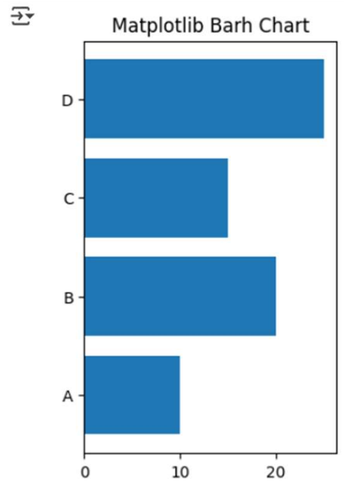
A **bar chart** is a graph that uses **rectangular bars** to represent data, where the **length or height of each bar is proportional to the value** it represents.

- Used to **compare categories** or **discrete data points**.
- Bars can be **vertical or horizontal**, and multiple series can be **grouped or stacked**.

```
plt.subplot(1,2,2)
plt.bar(df['Category'], df['Values1'], label='Values1')
plt.bar(df['Category'], df['Values2'], bottom=df['Values1'], label='Values2')  # stacked
plt.title("Matplotlib Bar Chart")
plt.legend()
plt.show()
```



```
plt.subplot(1,2,2)
plt.barh(df['Category'], df['Values1'])
plt.title("Matplotlib Barh Chart")
plt.show()
```



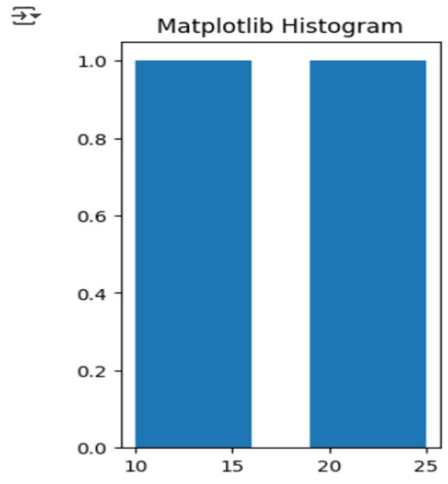
Histogram

Definition

A **histogram** is a graphical representation of the **distribution of numerical data**.

- Data is divided into **bins (intervals)**, and the **height of each bar** shows how many values fall into that bin.
- Useful for **understanding frequency, spread, and shape** of a dataset.
- Bars in a histogram **touch each other**, unlike a standard bar chart, to represent **continuous data**.

```
plt.subplot(1,2,2)
plt.hist(df['Values1'], bins=5)
plt.title("Matplotlib Histogram")
plt.show()
```



Box plot

Definition

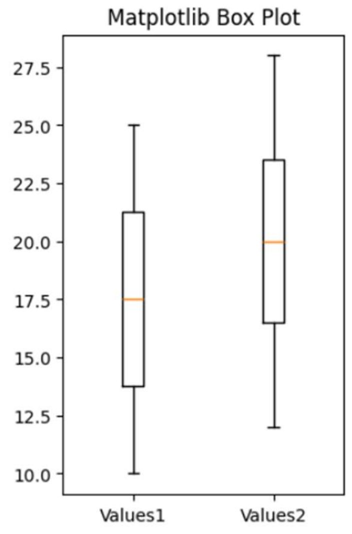
A **box plot** (or **whisker plot**) is a graphical representation of the **distribution of numerical data** using **five summary statistics**:

1. **Minimum** (lowest value)
2. **First Quartile (Q1)** – 25th percentile
3. **Median (Q2)** – 50th percentile
4. **Third Quartile (Q3)** – 75th percentile
5. **Maximum** (highest value)

- It also shows **outliers** as individual points.
- Useful for **comparing distributions across multiple datasets**.

```
plt.subplot(1,2,2)
plt.boxplot([df['Values1'], df['Values2']], labels=['Values1','Values2'])
plt.title("Matplotlib Box Plot")
plt.show()
```

```
/tmp/ipython-input-1659133459.py:2: MatplotlibDeprecationWarning: The 'labels' parameter of boxplot() has been re
  plt.boxplot([df['Values1'], df['Values2']], labels=['Values1','Values2'])
```



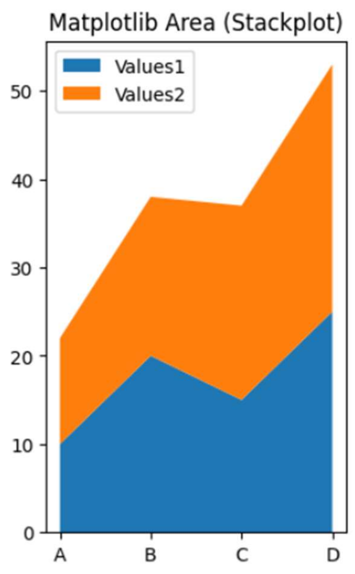
Area plot

Definition

An **area plot** is similar to a **line plot**, but the area **under the line is filled** with color.

- Used to **show cumulative trends** over time or categories.
- Useful for **visualizing part-to-whole relationships** when multiple series are stacked.

```
plt.subplot(1,2,2)
plt.stackplot(df['Category'], df['Values1'], df['Values2'], labels=['Values1','Values2'])
plt.legend()
plt.title("Matplotlib Area (Stackplot)")
plt.show()
```



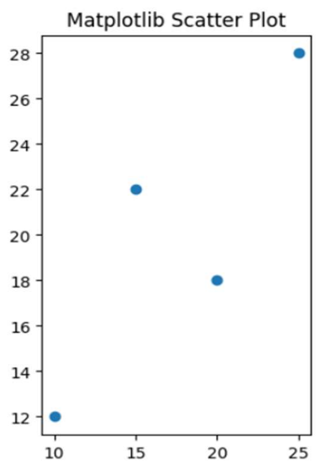
Scatter plot

Definition

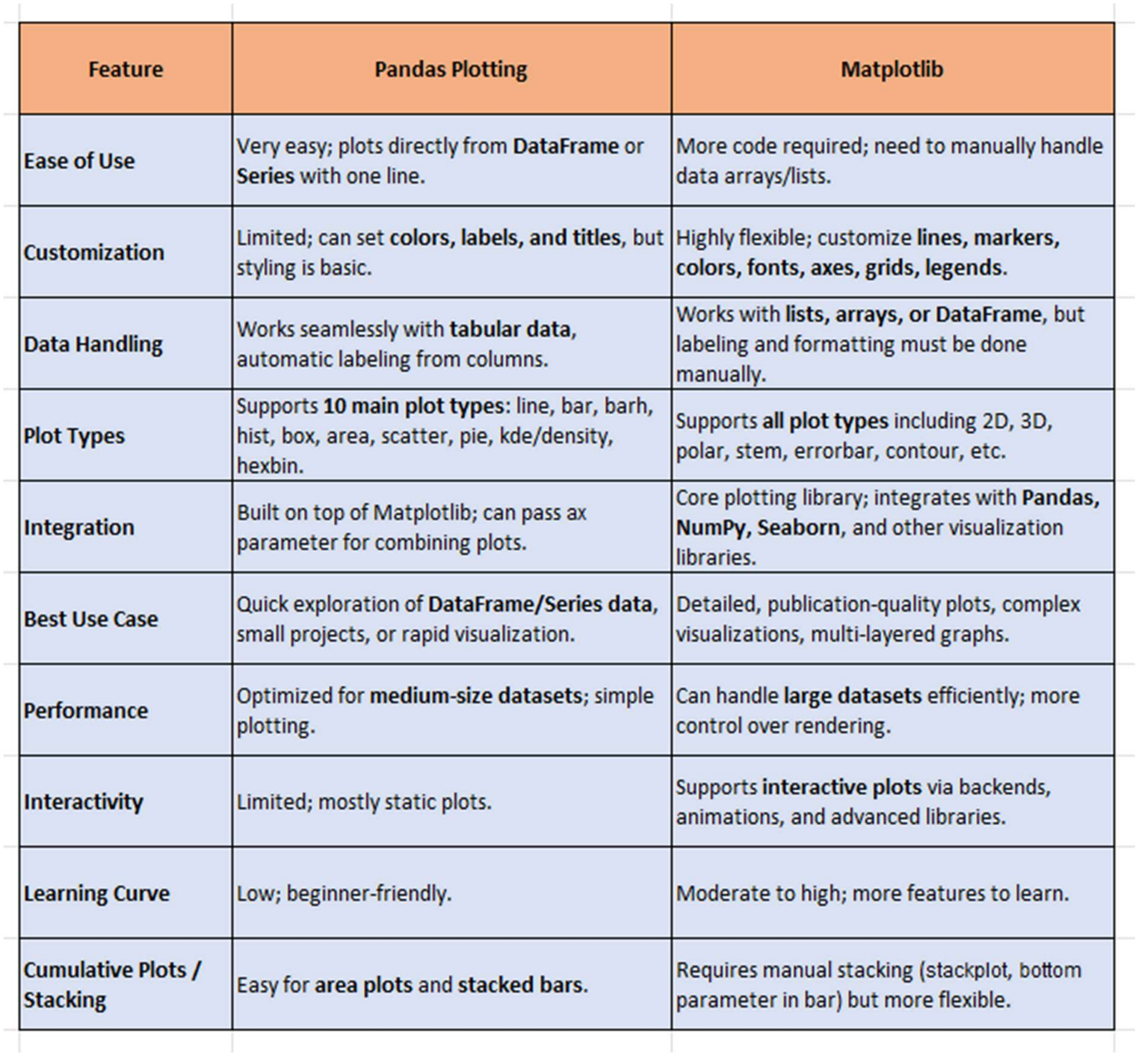
A **scatter plot** is a graph that displays **individual data points** on a 2D plane using **x and y coordinates**.

- Used to **visualize the relationship or correlation** between two numerical variables.
- Helps identify **patterns, trends, clusters, or outliers** in the data.

```
plt.subplot(1,2,2)
plt.scatter(df['Values1'], df['Values2'])
plt.title("Matplotlib Scatter Plot")
plt.show()
```



Comparison between pandas and matplotlib



Feature	Pandas Plotting	Matplotlib
Ease of Use	Very easy; plots directly from **DataFrame** or **Series** with one line.	More code required; need to manually handle data arrays/lists.
Customization	Limited; can set **colors, labels, and titles**, but styling is basic.	Highly flexible; customize **lines, markers, colors, fonts, axes, grids, legends**.
Data Handling	Works seamlessly with **tabular data**, automatic labeling from columns.	Works with **lists, arrays, or DataFrame**, but labeling and formatting must be done manually.
Plot Types	Supports **10 main plot types**: line, bar, barh, hist, box, area, scatter, pie, kde/density, hexbin.	Supports **all plot types** including 2D, 3D, polar, stem, errorbar, contour, etc.
Integration	Built on top of Matplotlib; can pass ax parameter for combining plots.	Core plotting library; integrates with **Pandas, NumPy, Seaborn**, and other visualization libraries.
Best Use Case	Quick exploration of **DataFrame/Series data**, small projects, or rapid visualization.	Detailed, publication-quality plots, complex visualizations, multi-layered graphs.
Performance	Optimized for **medium-size datasets**; simple plotting.	Can handle **large datasets** efficiently; more control over rendering.
Interactivity	Limited; mostly static plots.	Supports **interactive plots** via backends, animations, and advanced libraries.
Learning Curve	Low; beginner-friendly.	Moderate to high; more features to learn.
Cumulative Plots / Stacking	Easy for **area plots** and **stacked bars**.	Requires manual stacking (stackplot, bottom parameter in bar) but more flexible.