Data Visualization — Matplotlib & Pandas

Comprehensive Beginner Guide with Examples, Code & Outputs

NAME: Pooja M

DATE: October 28 2025

DOMAIN: Data Science

DELIVERABLE: Internship Pdf or Markdown

1. Introduction to Data Visualization

Data visualization is the graphical representation of information and data. It helps translate large, complex datasets into visual formats — charts, graphs, and maps — that are easier to understand and interpret. Effective visualization reveals patterns, trends, outliers, and relationships that might be missed in raw tables. Visualizations are essential in exploratory data analysis (EDA), reporting, and communicating findings to stakeholders.

Why visualization matters

- Quick pattern recognition and trend detection.
- Simplifies communication of results to non-technical audiences.
- Aids in hypothesis generation during EDA.
- Supports data-driven decision making

How this document is organized

This document covers two libraries — Matplotlib and Pandas (plotting).
For each library you will find: overview, unique features, use-cases, five chart types with code and outputs, and comparison. References and an appendix with full code are included at the end.

2. Matplotlib

Matplotlib is a mature, full-featured Python library for creating static, animated, and interactive visualizations. It provides fine-grained control over plots — axes, tick marks, labels, legends, figure size, fonts, and more — making it suitable for publication-quality figures.

Unique features

- Low-level control (fine-grained customization).
- Large ecosystem (works with seaborn, pandas, cartopy).
- Multiple output formats (PNG, PDF, SVG).
- Ability to create complex multi-axis and subplot figures.

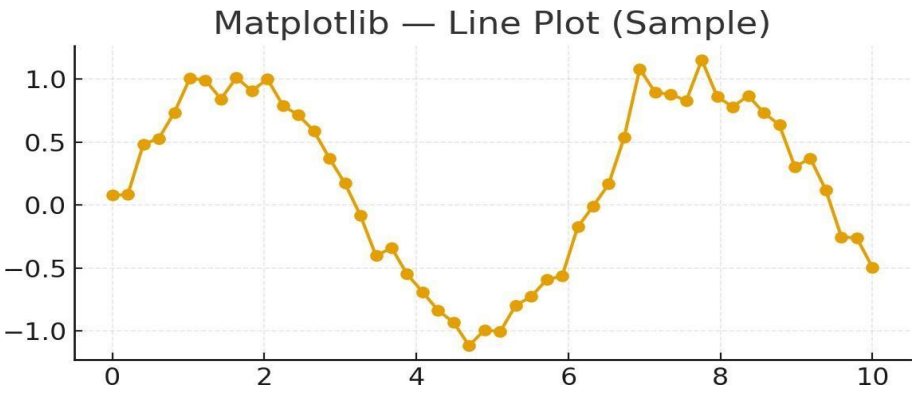
Typical use-cases

- Publication-quality figures for academic papers.
- Detailed exploratory analysis requiring custom styling.
- Building static images for reports and dashboards.

Line Plot

Displays continuous data and trends over a numeric or time axis. When to use: Trend analysis, time series visualization.
Code example:

```
import matplotlib.pyplot as
plt plt.plot(x, y)
plt.title('Line Plot')
plt.show()
```

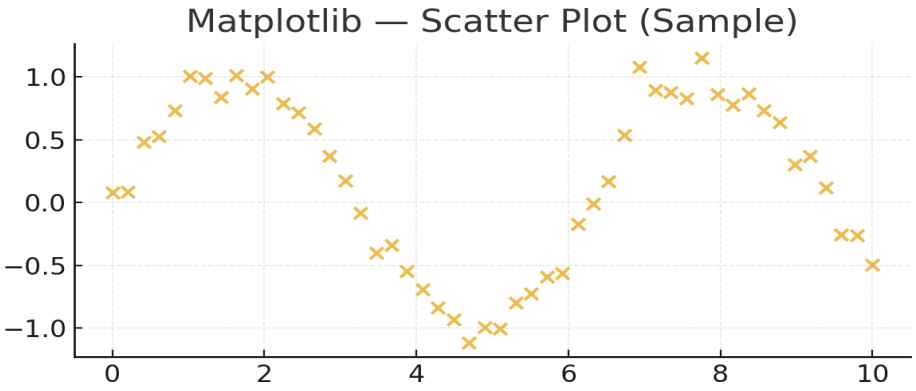


Interpretation: See the plot above for visual cues.

Scatter Plot

Shows relationship and correlation between two numeric variables. When to use: Correlation, cluster inspection.
Code example:

```
plt.scatter(X, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

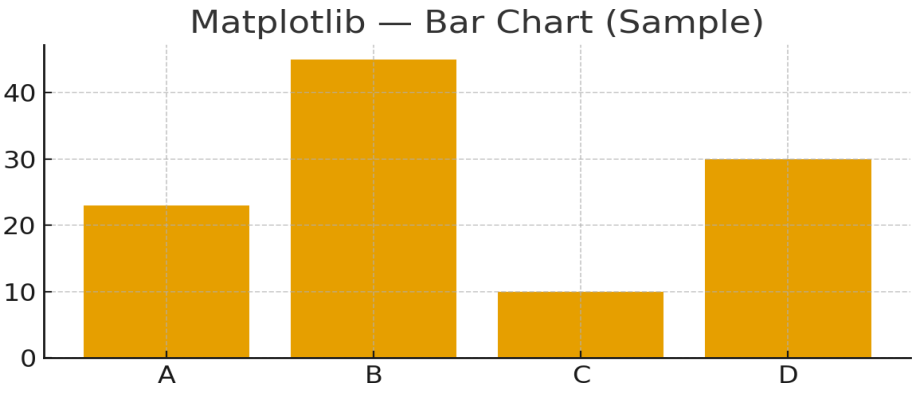


Interpretation: See the plot above for visual cues.

Bar Chart

Compares categorical values using rectangular bars.
When to use: Comparing categories or
aggregated metrics. Code example:

```
categories =
['A','B','C'] values =
[5,7,3]
plt.bar(categories,
values) plt.show()
```

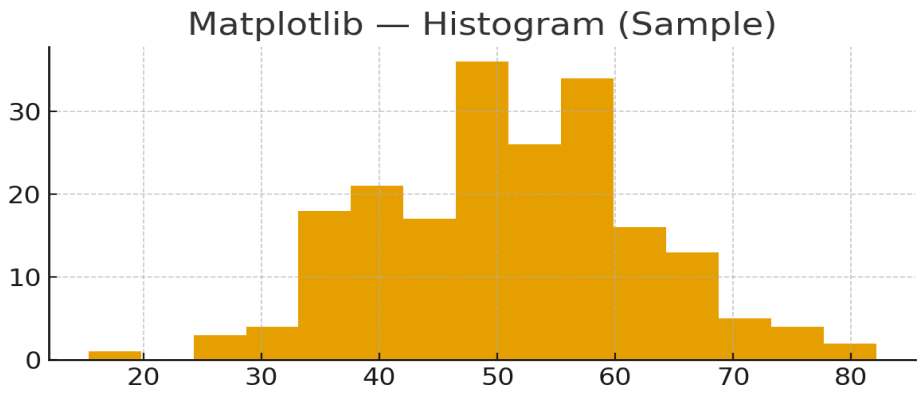


Interpretation: See the plot above for visual cues.

Histogram

Shows distribution (frequency) of a numerical variable.
When to use: Understanding distribution,
skewness, modality. Code example:

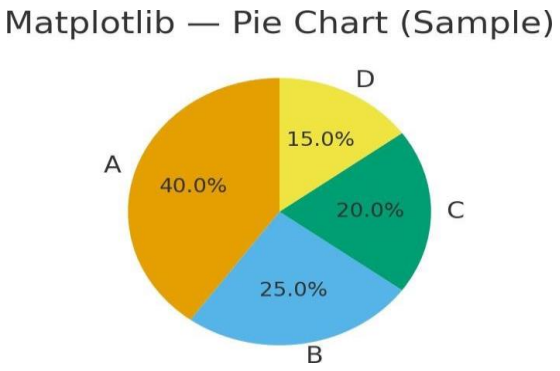
```
plt.hist(data, bins=10)
plt.show()
```



Pie Chart

Displays parts-of-a-whole as slices of a circle.
When to use: Simple composition/proportion views (use sparingly). Code example:

```
plt.pie([30,20,50], labels=['A','B','C'],
autopct='%1.1f%%') plt.show()
```



3. Pandas (plotting)

Pandas is a powerful library for data manipulation. Its plotting API (df.plot / series.plot) is a convenient wrapper around Matplotlib that allows quick visualizations directly from Data Frames and Series.

Unique features

- One-line plotting from DataFrames and Series.
- Automatic index and label handling.
- Seamless integration with pandas transformations (groupby, resample).
- Great for fast EDA.

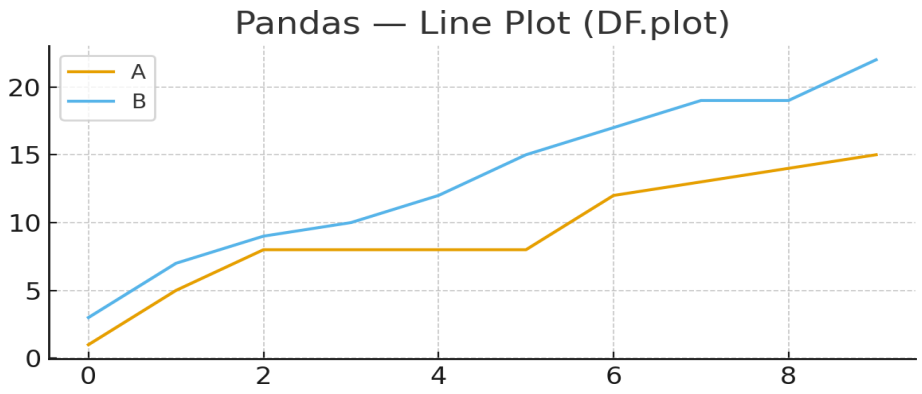
Typical use-cases

- Quick exploratory plots during EDA.
- Prototyping visualizations directly from cleaned DataFrames.
- Visual checks after aggregations or groupby operations.

Line Plot

df.plot() creates quick line plots from DataFrame/Series. When to use: Quick trend checks across columns.
Code example:

```
df.plot(kind='line')
plt.show()
```

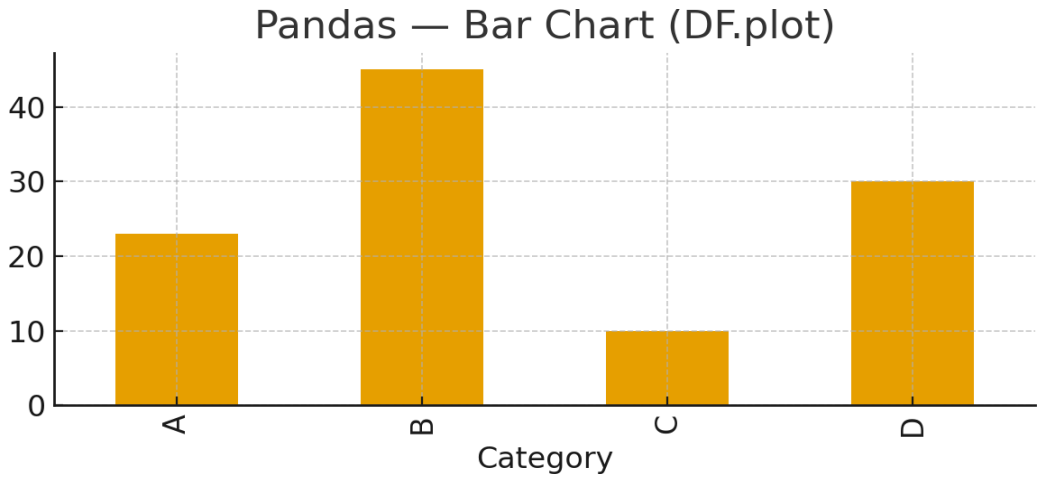


Interpretation: See the plot above for quick insights.

Bar Chart

df.plot.bar() for categorical comparisons.
When to use: Category comparisons after groupby/aggregation. Code example:

```
df.plot(kind='bar', x='Category', y='Values')
plt.show()
```



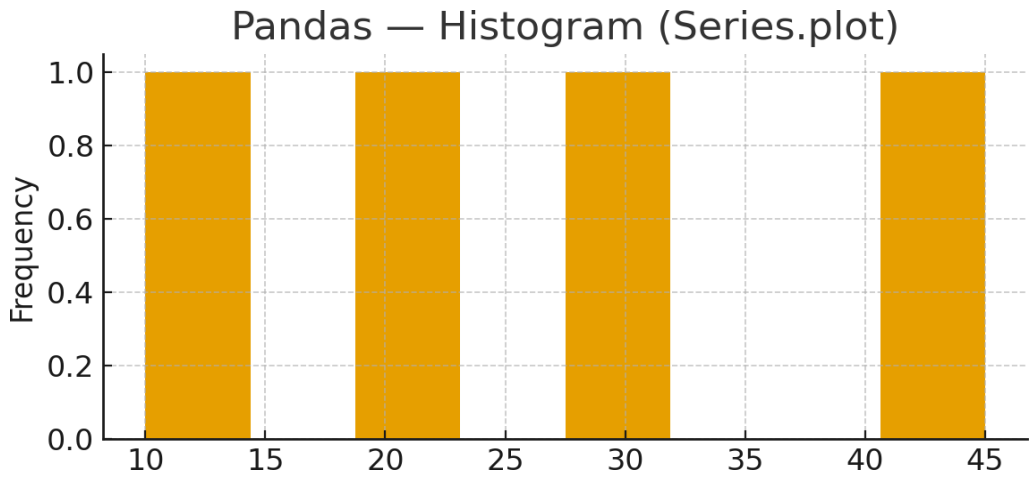
Interpretation: See the plot above for quick insights.

Histogram

series.plot.hist() shows distribution of a Series. When to use: Distribution checks for a column.

Code example:

```
df['Values'].plot(kind='hist', bins=10) plt.show()
```

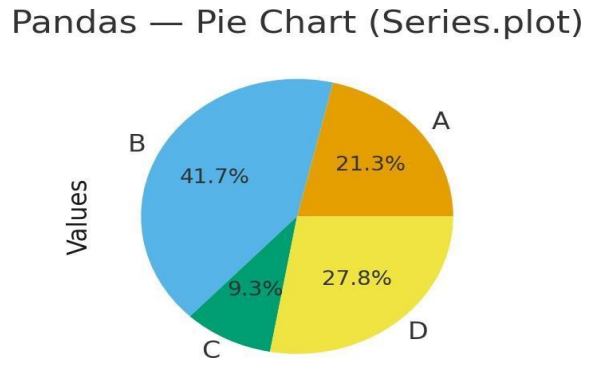


Interpretation: See the plot above for quick insights.

Pie Chart

series.plot.pie() for composition visualization.
When to use: Small number of categories for
proportion view. Code example:

```
df.set_index('Category')['Values'].plot.pie(autopct=
'%1.1f%%') plt.show()
```



Interpretation: See the plot above for quick insights.

4. Comparison and Recommendations

Feature	Matplotlib	Pandas (plot)
Ease of Use	Requires learning core API; more ver	Id si geh — one-liners & quick plots
Customization	Very high — control over every elem	nLtimited — uses Matplotlib underneath
Interactivity	Supports basic interactivity; reserve f	Lstatic-ustatior Matplotlib-backedlimages
Performance (large data)	Good with optimized backends & do	nDs ea pmepnlidnsg on Matplotlib; better for sma ll-medium da
Best suited for	Publication-quality & complex figures	Rapid EDA and prototyping

Matplotlib Strengths

- Extremely customizable
- Large ecosystem
- Good for publication figures

Matplotlib Weaknesses

- Steeper learning curve
- Verbose for simple tasks

Pandas Strengths

- Quick plotting from DataFrames
- Great for fast EDA

Pandas Weaknesses

- Less customization without Matplotlib tweaks
- Not for final polished figures without extra work

5. References

Matplotlib Official Documentation: https://matplotlib.org/stable/
Pandas Visualization Guide:
https://pandas.pydata.org/docs/user_guide/visualization.html
StackOverflow and Matplotlib community examples for customization
tips.

Conclusion

Matplotlib and Pandas complement each other — Pandas for speed and convenience during EDA, and Matplotlib for detailed, publication-ready figures. For internship-level tasks, using Pandas for quick checks and Matplotlib for final visuals is a practical workflow.

Appendix — Full code examples

```
# Full example: Matplotlib & Pandas snippets (see README for
runnable notebook) import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Matplotlib line
x = np.linspace(0,10,50)
y = np.sin(x) + 0.1*np.random.randn(50)
plt.plot(x,y,marker='o'); plt.title('Line');
plt.show()

# Pandas bar
df =
pd.DataFrame({'Category':['A','B','C'],'Values':[10
,20,15]}) df.plot(kind='bar', x='Category',
y='Values'); plt.show()
```