

# Data Science

**By:**  
**Dr. Shikha Deep**

# PLOTTINGS

## Contents

- ❖ Basic Plots
- ❖ Visualizing distributions
- ❖ Detecting Outliers
- ❖ Heatmaps

# Plotting

- Plotting means creating visual representations of data to make it easier to observe patterns, trends, and insights.

## **Why Plotting is Important:**

- Makes data easy to understand
- Helps in decision making
- Reveals **hidden patterns, trends, and outliers**
- Makes communication of results more visual and impactful

# Plotting in Python: Libraries

To plot graphs in Python, we mainly use:

## 1. Matplotlib

- Core plotting library in Python.
- Easy for basic plots.

## 2. Pandas

- Easier syntax for simple plots using `.plot()`.

## 3. Seaborn

- Built on Matplotlib, used for statistical and attractive visualizations.
- Includes advanced plots like heatmaps, violin plots, etc.

# How to Install Plotting Libraries

```
pip install matplotlib seaborn pandas
```

## How to Import

```
import pandas as pd
```

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

```
import seaborn as sns
```

Note: Use `%matplotlib inline` if working in Jupyter Notebook to show plots inline.

## Example :

```
import pandas as pd
```

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

```
data = {  
    'Month': ['Jan', 'Feb', 'Mar', 'Apr', 'May'],  
    'Sales': [250, 400, 300, 500, 450],  
    'Profit': [80, 120, 100, 160, 130]  
}
```

```
df = pd.DataFrame(data)
```

```
print (data)
```

```
Print (df)
```

# 1. Line Plot

- A graph that connects data points using a straight line.
- To **show trends over time** (like monthly sales or temperature changes).

## Code 1: Line Plot

```
df.plot(x='Month', y='Sales', kind='line', marker='o')  
plt.title("Monthly Sales Line Plot")  
plt.xlabel("Month")  
plt.ylabel("Sales")  
plt.grid(True)  
plt.show()
```

Note: `marker='o'` or `marker='*'` or `marker='s'`

The marker is used to highlight individual data points on a line plot.



## 2. Bar Graph

- A chart with rectangular bars representing values of different categories.
- To compare values across different groups/categories.

## Code 2 : Bar Graph

```
df.plot(x='Month', y='Sales', kind='bar', color='skyblue')  
plt.title("Sales Bar Chart")  
plt.xlabel("Month")  
plt.ylabel("Sales")  
plt.show()
```

### 3. Pie Chart

- A circular chart divided into slices to illustrate parts of a whole.
- To show proportions or percentage share (like % of total sales by month).

## Code 3 : Pie Chart

```
plt.pie(df['Sales'], labels=df['Month'], autopct='%1.1f%%')  
plt.title("Sales Share by Month")  
plt.show()
```

### NOTE:

(1) **autopct = “auto percentage”**

It is used to display the percentage value on each slice of the pie chart.

(2)

Parameter	Meaning
%1.1f%%	1 decimal place floating number + percent sign
%1.2f%%	2 decimal places

## 4. Scatter Plot

- A graph with points plotted on X and Y axes to show relationships.
- To visualize the correlation or relationship between two numeric variables.

## Code 4 : Scatter Plot

```
df.plot(kind='scatter', x='Sales', y='Profit', color='red')  
plt.title("Sales vs Profit")  
plt.xlabel("Sales")  
plt.ylabel("Profit")  
plt.show()
```

## 5. Histogram

- A histogram is a type of bar chart that represents the frequency distribution of numerical data by grouping values into intervals or bins.

### Useful for:

To understand the distribution (normal, skewed, uniform, etc.)

To find data spread, central values, or gaps

To detect peaks or skewness

## Code 5 : Histogram

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

data = {'Marks': [23, 45, 55, 60, 67, 67, 70, 72, 75, 78, 80, 85, 85, 90, 95, 100]}
df = pd.DataFrame(data)

plt.figure(figsize=(8,5))
plt.hist(df['Marks'], bins=5, color='skyblue', edgecolor='black')

plt.title('Histogram of Marks')
plt.xlabel('Marks Range')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



