# Visualization Library Documentation: Matplotlib & Seaborn

## Objective

Create a comprehensive documentation guide for two Python visualization libraries—Matplotlib and Seaborn—focusing on the variety of graphs each can generate with practical examples (code + output images).

# 1. Matplotlib

## Overview

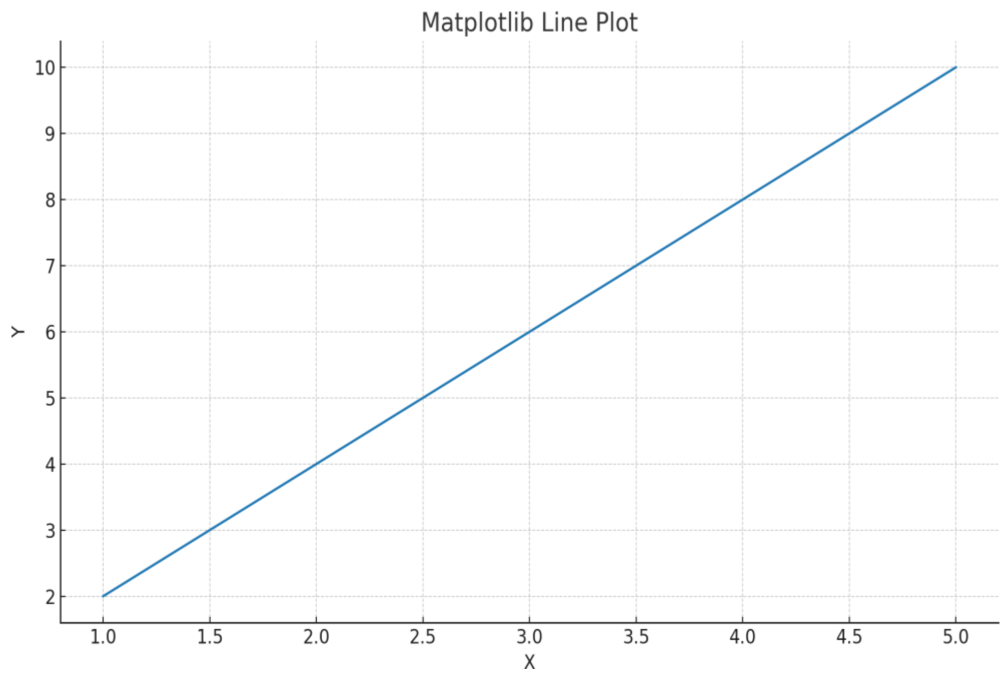
Matplotlib is the foundational plotting library in Python for creating static, publication-quality figures. It offers fine-grained control over every element of a plot (axes, ticks, spines, annotations). Typical use cases: academic research papers, custom dashboards, and any scenario requiring pixel-level customization.

### *Line Plot*

A line plot, also known as a line graph or line chart, is a type of graph that displays data points connected by line segments to show changes in value over a continuous progression. It's a common way to visualize trends and relationships between variables, especially when one variable represents a continuous sequence like time.

Use to show trends over ordered data (e.g., time series).

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot(x, y)
plt.title("LinePlot")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

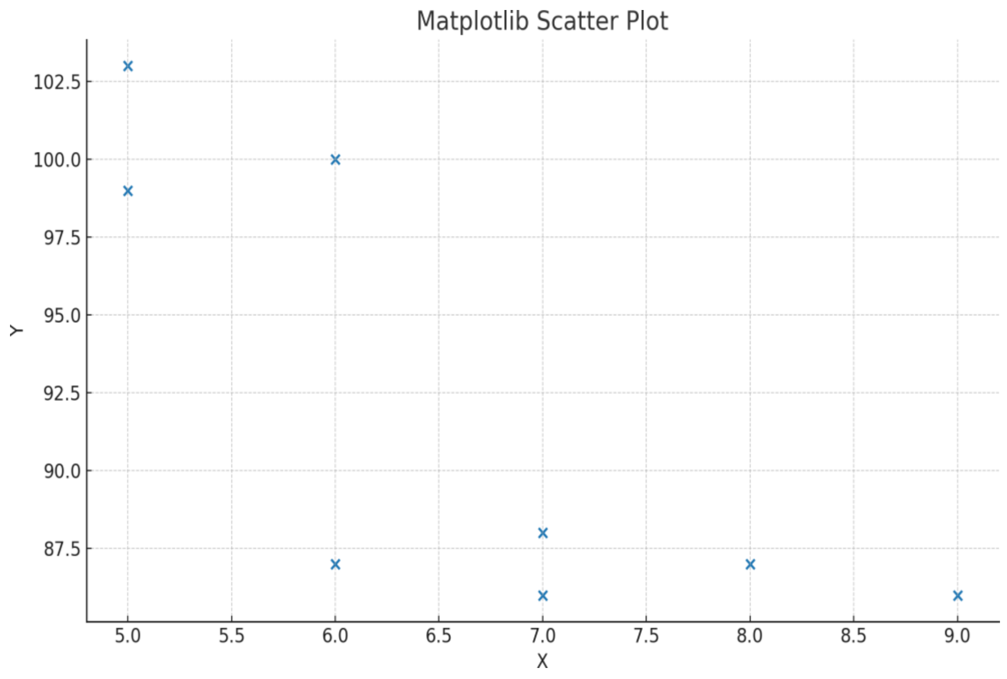


***Scatter Plot***

A scatter plot is a type of graph that displays the relationship between two numerical variables using dots. Each dot on the plot represents a single data point, with its position determined by the values of the two variables. By examining the pattern of dots, you can gain insights into the relationship, or correlation, between the variables.

Use to visualize relationships/correlation between two numeric variables.

```
import matplotlib.pyplot as plt
x = [5, 7, 8, 7, 6, 9, 5, 6]
y = [99, 86, 87, 88, 100, 86, 103, 87]
plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

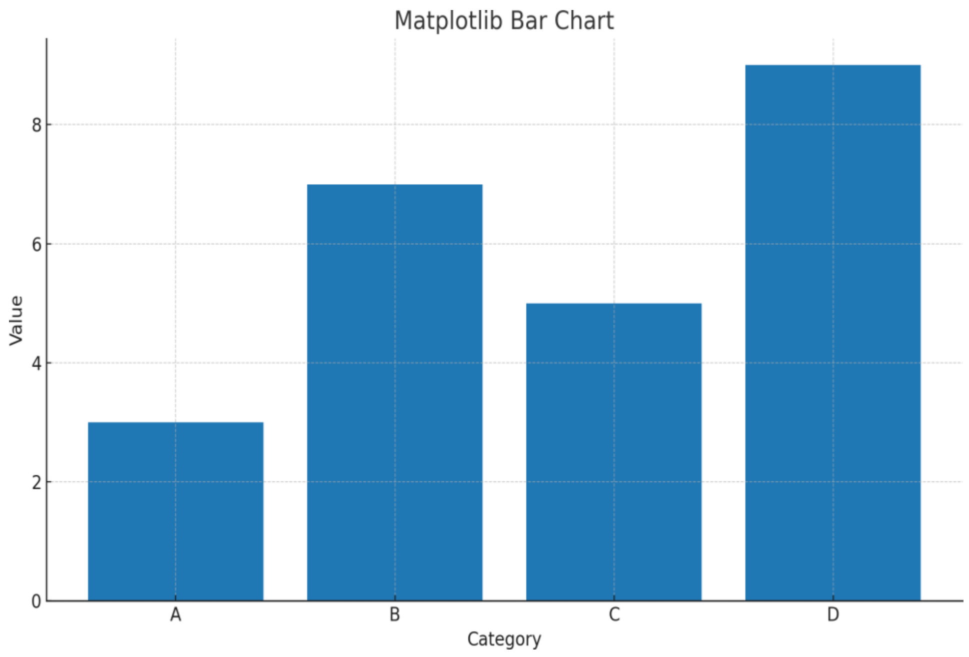


***Bar Chart***

A diagram that uses narrow bands of different heights to show different amounts so that they can be compared.

Use to compare aggregated values across categories.

```
import matplotlib.pyplot as plt
categories = ["A", "B", "C", "D"]
values = [3, 7, 5, 9]
plt.bar(categories, values)
plt.title("Bar Chart")
plt.xlabel("Category")
plt.ylabel("Value")
plt.show()
```

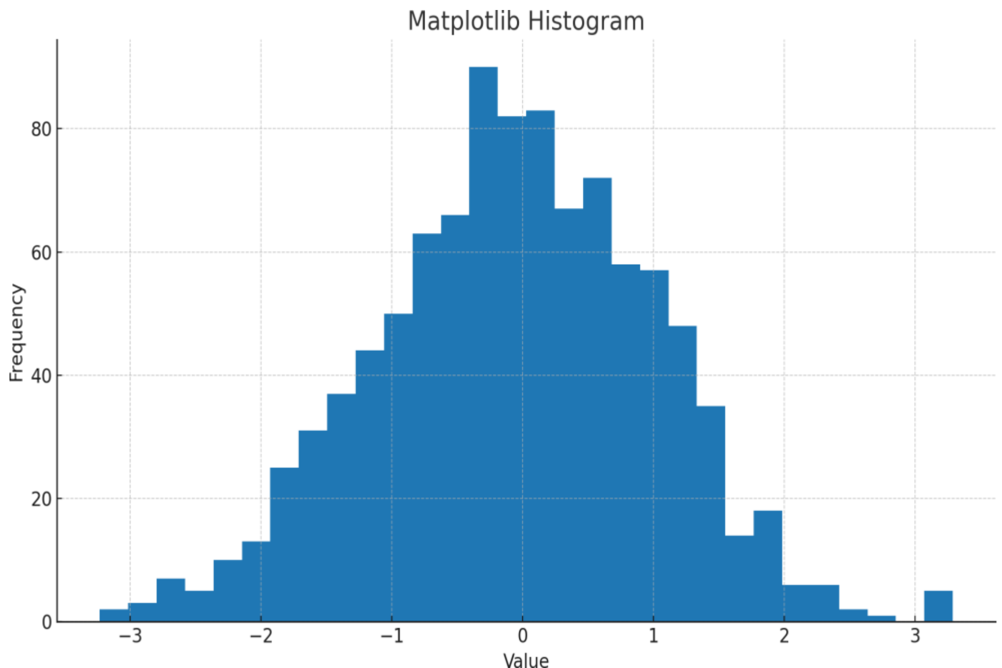


***Histogram***

A histogram is a graphical representation of the distribution of numerical data, showing the frequency of data points within specified ranges or bins. It's essentially a type of bar chart where the x-axis represents the range of values (bins) and the y-axis represents the frequency or count of data points within each bin. Histograms are useful for understanding the distribution, central tendency, and variability of a dataset.

Use to show the distribution of a variable and spot skewness/outliers.

```
import matplotlib.pyplot as plt
import numpy as np
data = np.random.randn(1000)
plt.hist(data, bins=30)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show(
```

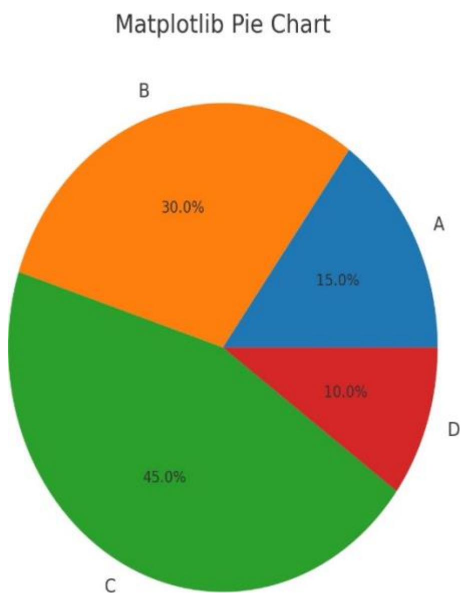


***Pie Chart***

A pie chart is a circular graph divided into sectors, where each sector's size represents a proportion of the whole. It's used to visualize the relative size of different categories within a dataset.

Use to show part-to-whole composition when categories are few and distinct.

```
import matplotlib.pyplot as plt
sizes = [15, 30, 45, 10]
labels = ["A", "B", "C", "D"]
plt.pie(sizes, labels=labels, autopct="%1.1f%%")
plt.title("Pie Chart")
plt.show()
```



## 2. Seaborn

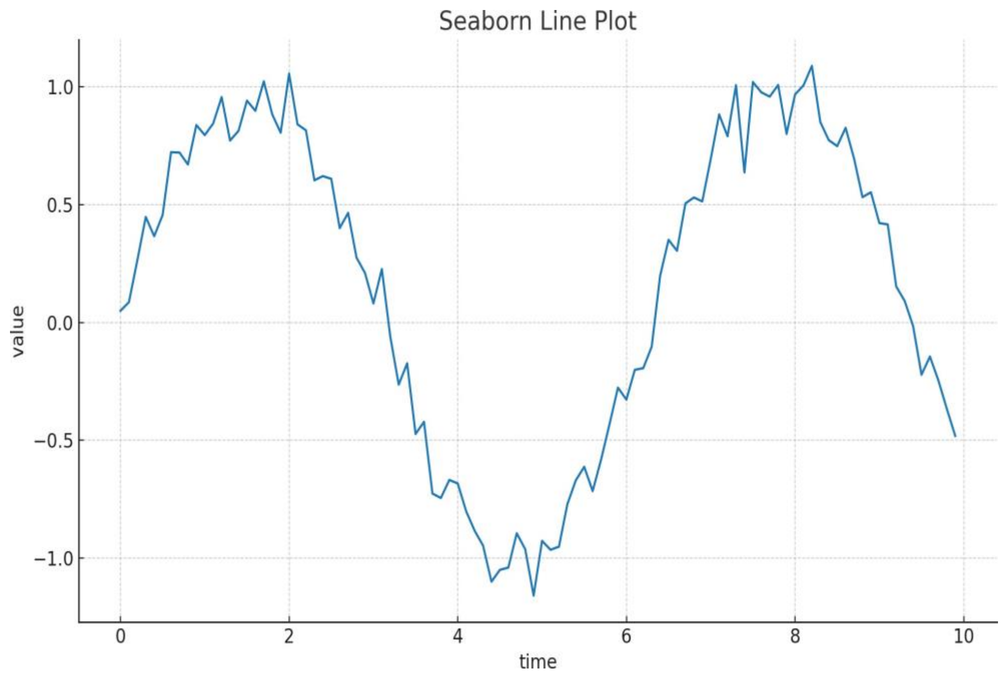
### Overview

Seaborn is a high-level visualization library built on top of Matplotlib. It offers beautiful defaults, tight integration with pandas DataFrames, and many statistical plots out of the box. Typical use cases: exploratory data analysis, quick statistical visualizations, and consistent styling across plots.

***Line Plot***
A pie chart is a circular graph divided into sectors, where each sector's size represents a proportion of the whole. It's used to visualize the relative size of different categories within a dataset.

Use to show smoothed trends or changes over a continuous variable.

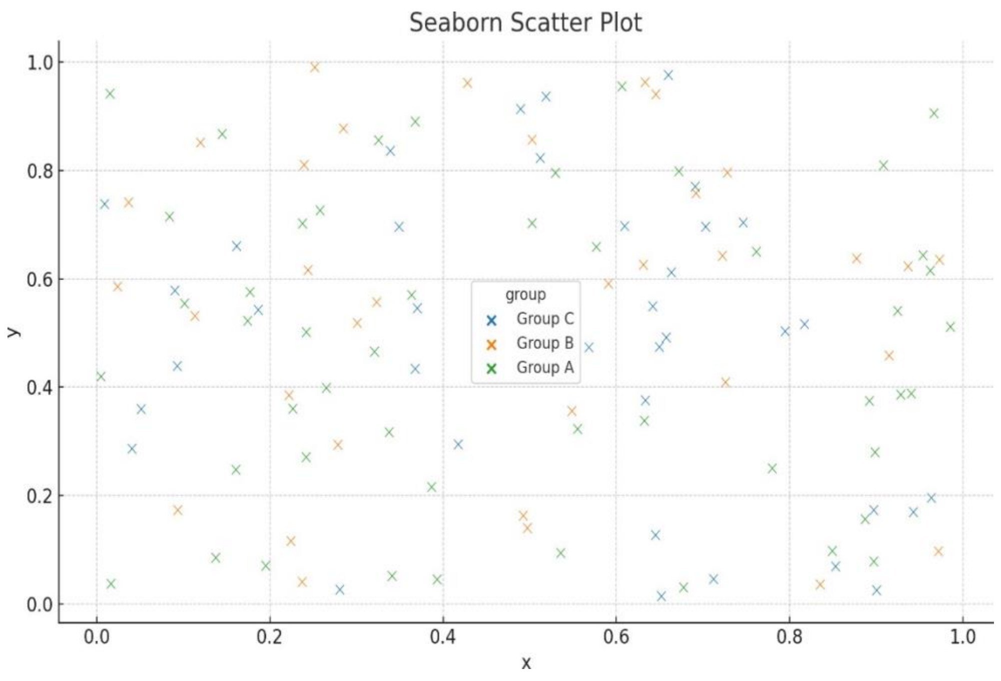
```
import seaborn as sns
import pandas as pd
import numpy as np
df = pd.DataFrame({"time": np.arange(0, 10, 0.1)})
df["value"] = np.sin(df["time"]) + 0.1 * np.random.randn(len(df))
sns.lineplot(x="time", y="value", data=df)
```



***Scatter Plot (with hue)***
A scatter plot is a type of graph that displays the relationship between two numerical variables using dots. Each dot on the plot represents a single data point, with its position determined by the values of the two variables. By examining the pattern of dots, you can gain insights into the relationship, or correlation, between the variables.

Use to examine relationships and clusters; color encodes a third categorical variable.

**import seaborn as sns import pandas as pd import numpy as np**
**df = pd.DataFrame({**
**"x": np.random.rand(120),**
**"y": np.random.rand(120),**
**"group": np.random.choice(["Group A", "Group B", "Group C"], 120)})**
**sns.scatterplot(x="x", y="y", hue="group", data=df)**

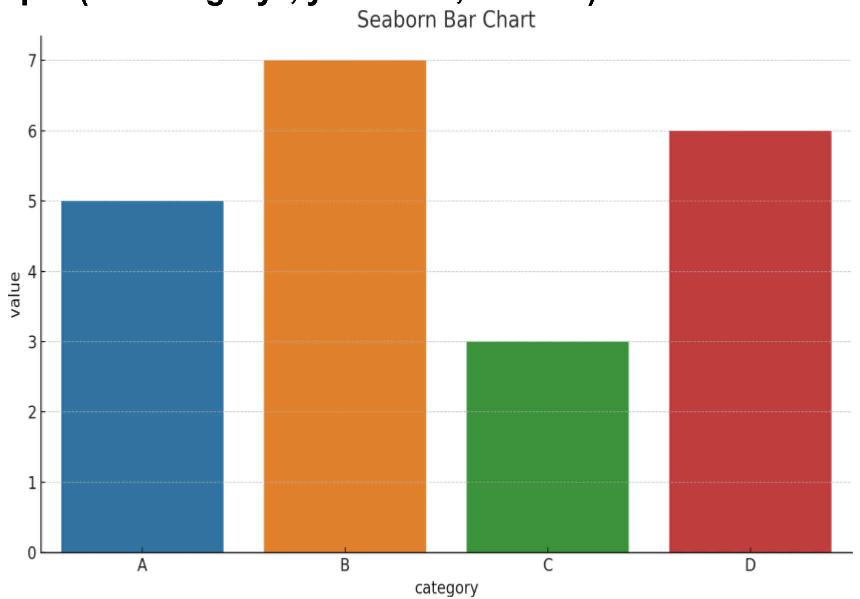


***Bar Chart***

A bar plot in Seaborn is created using the seaborn.barplot() function and is used to visualize the relationship between a categorical variable and a numerical variable. It displays point estimates (such as the mean) and confidence intervals as rectangular bars. The height of each bar represents an estimate of central tendency for the numerical variable, while error bars indicate the uncertainty around that estimate.

Use to compare categories; Seaborn can compute confidence intervals/estimates automatically.

**import seaborn as sns import pandas as pd**
**df = pd.DataFrame({"category": ["A", "B", "C", "D"],**
**"value": [5, 7, 3, 6]})**
**sns.barplot(x="category", y="value", data=df)**

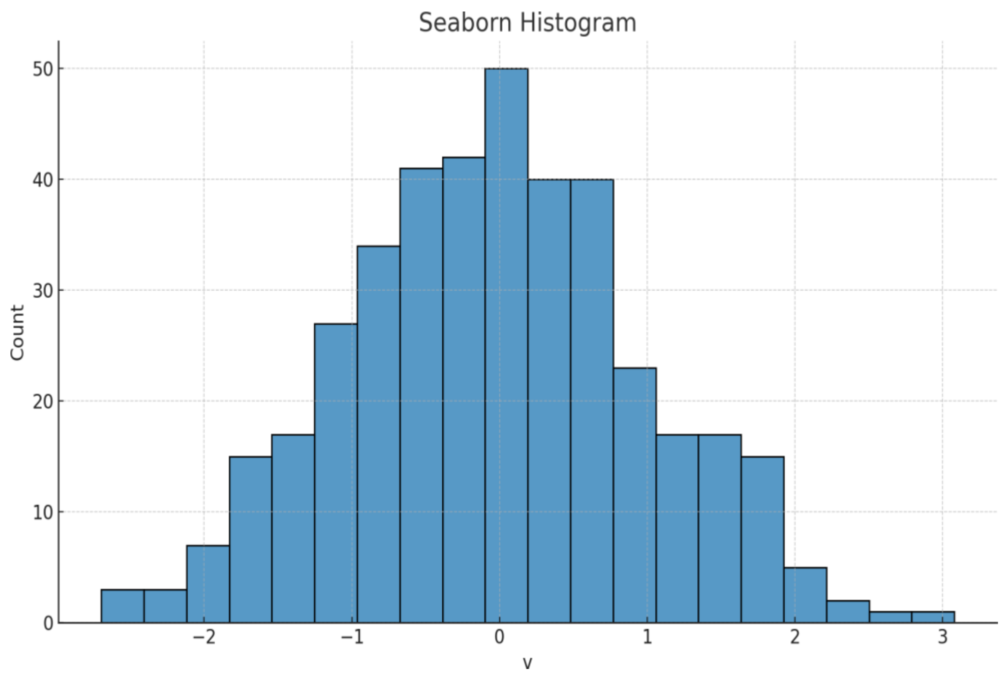


## *Histogram*

A histogram is a classic visualization tool that represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.

Use to show the distribution of a variable; optionally add KDE for smooth density.

```
import seaborn as sns import numpy as np
data = np.random.randn(400)
sns.histplot(data, bins=20, kde=False)
```

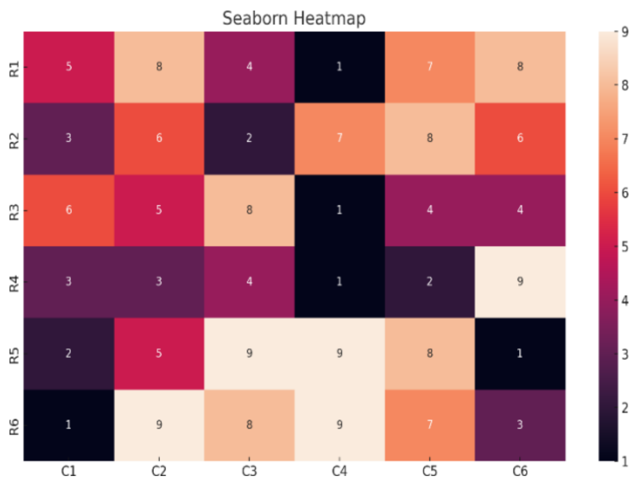


## *Heatmap*

A heatmap in Seaborn is a two-dimensional graphical representation of data where individual values within a matrix are represented by colors. It is a powerful visualization tool used to display the magnitude of a phenomenon, patterns, and correlations within a dataset.

Use to visualize matrix-like data or correlation matrices.

```
import seaborn as sns import pandas as pd import numpy as np
mat = pd.DataFrame(np.random.randint(1, 10, (6, 6)),
columns=[f"C{i}" for i in range(1, 7)], index=[f"R{i}" for i in range(1, 7)])
sns.heatmap(mat, annot=True, fmt="d")
```



|  | C1 | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|
| R1 | 5 | 8 | 4 | 1 | 7 | 8 |
| R2 | 3 | 6 | 2 | 7 | 8 | 6 |
| R3 | 6 | 5 | 8 | 1 | 4 | 4 |
| R4 | 3 | 3 | 4 | 1 | 2 | 9 |
| R5 | 2 | 5 | 9 | 9 | 8 | 1 |
| R6 | 1 | 9 | 8 | 9 | 7 | 3 |

## 3. Comparison: Matplotlib vs Seaborn (5 Key Differences)

### Strengths & Weaknesses

***Matplotlib***
Strengths: ultimate customization; publication-quality; vast ecosystem; precise layout control.
Weaknesses: more verbose; styling requires effort; interactivity limited without extra tools.

***Seaborn***
Strengths: beautiful defaults; statistical plots built-in; fast for EDA; pandas-friendly.
Weaknesses: less granular control than Matplotlib; advanced customization sometimes requires dropping to Matplotlib APIs.

### Five Key Differences

**Level of Control vs. Convenience:** Matplotlib gives granular control over every plot element; Seaborn prioritizes convenience and attractive defaults.
**Statistical Plots:** Seaborn includes high-level statistical plots (box/violin/swarm, categorical plots, heatmaps) with minimal code; Matplotlib requires more manual work or extensions for similar results.
**Integration with pandas:** Seaborn integrates tightly with pandas DataFrames and can use column names directly; Matplotlib typically needs arrays/Series passed explicitly.
**Styling & Themes:** Matplotlib needs more code to style consistently across figures; Seaborn provides themes/palettes that yield consistent, modern styling out-of-the-box.
**Learning Curve & Speed to Insight:** Matplotlib has a steeper learning curve but ultimate flexibility; Seaborn accelerates exploratory analysis with fewer lines of code.