Visualization Guide — Matplotlib & Seaborn

Level: Beginner

Objective: A compact, practical reference showing what Matplotlib and Seaborn can do, with short explanations, use-cases, and copy-paste-ready code examples.

Table of contents

---

1) Prerequisites

Make sure you have Python 3.8+ and these packages installed:

```
pip install matplotlib seaborn numpy pandas
```

---

2) Library overviews

Matplotlib

What it is: The foundational Python plotting library. Low-level but very flexible — the base on which many other libraries (including Seaborn) are built.

Strengths: Fine-grained control, highly customizable, wide ecosystem, good for publishing-quality static figures.

Typical use-cases: Scientific plotting, custom figures, fine control of axes/annotations, static reports.

Seaborn

What it is: A statistical visualization library built on top of Matplotlib. It exposes convenient high-level functions for common plot types and statistical graphics, with attractive default styles.

Strengths: Quick to create informative statistical graphics, great defaults and themes, easy to add statistical summaries (confidence intervals, aggregations).

Typical use-cases: Exploratory Data Analysis (EDA), quick statistical plots, attractive visualizations for reports and presentations.

---

3) Matplotlib — Graph types, descriptions, and examples

All Matplotlib examples assume the usual imports:

```
import numpy as np

import matplotlib.pyplot as plt
```

Line plot — plt.plot

Description: Connects points in order. Great for time series and continuous functions.

Use case: Plotting sensor readings, stock prices, model loss/accuracy over epochs.

Example:

```
x = np.linspace(0, 10, 200)
```

```
y = np.sin(x)

plt.figure(figsize=(8,4))

plt.plot(x, y, label='sin(x)', linewidth=2)

plt.title('Line plot — sine wave')

plt.xlabel('x')

plt.ylabel('y')

plt.legend()

plt.grid(True)

plt.show()
```

Scatter plot — plt.scatter

Description: Plots unconnected points. Good for showing relationships and distributions across two variables.

Use case: Visualizing two-feature correlations, cluster spread.

Example:

```
np.random.seed(0)

x = np.random.rand(100)

y = np.random.rand(100)

sizes = 100 * np.random.rand(100)

colors = np.random.rand(100)

plt.scatter(x, y, s=sizes, c=colors, alpha=0.6)

plt.title('Scatter plot — random points')

plt.xlabel('x')

plt.ylabel('y')

plt.colorbar(label='color scale')

plt.show()
```

Bar chart — plt.bar / plt.barh

Description: Categorical comparisons using vertical (or horizontal) bars.

Use case: Showing counts or aggregated values (e.g., sales per region).

Example:

```
labels = ['A', 'B', 'C', 'D']
values = [23, 45, 56, 12]
plt.figure(figsize=(6,4))
plt.bar(labels, values)
plt.title('Bar chart — categorical values')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```

Histogram — plt.hist

Description: Distribution of a single numeric variable.

Use case: Showing frequency distribution (e.g., exam scores).

Example:

```
data = np.random.normal(loc=0, scale=1, size=1000)
plt.hist(data, bins=30, edgecolor='k', alpha=0.7)
plt.title('Histogram — normal distribution')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

Pie chart — plt.pie

Description: Part-to-whole proportions. Use sparingly; bars are often clearer.

Use case: Share of categories in a small set.

Example:

```
sizes = [15, 30, 45, 10]
labels = ['Group A', 'Group B', 'Group C', 'Group D']
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('Pie chart — category shares')
plt.axis('equal')  # keeps pie circular
plt.show()
```

Boxplot — plt.boxplot

Description: Distribution + median + quartiles + outliers.

Use case: Comparing distributions across groups.

Example:

```
data = [np.random.normal(loc, 1, 200) for loc in [0, 2, 4]]
plt.boxplot(data, labels=['Group 1', 'Group 2', 'Group 3'])
plt.title('Boxplot — compare distributions')
plt.show()
```

Heatmap / imshow — plt.imshow / plt.matshow

Description: Visualizes matrices (e.g., correlation matrices).

Use case: Correlation matrices, image arrays, confusion matrices.

Example:

```
matrix = np.random.rand(10, 10)
plt.imshow(matrix, aspect='auto')
plt.colorbar()
plt.title('Heatmap — random matrix')
plt.show()
```

Subplots — plt.subplots

Description: Multiple plots in a single figure.

Use case: Comparing multiple views or related plots in one figure.

Example:

```
fig, axes = plt.subplots(2, 2, figsize=(10, 6))
axes[0,0].plot(x, np.sin(x)); axes[0,0].set_title('sin')
axes[0,1].plot(x, np.cos(x)); axes[0,1].set_title('cos')
axes[1,0].hist(np.random.randn(500), bins=20); axes[1,0].set_title('hist')
axes[1,1].scatter(np.random.rand(50), np.random.rand(50)); axes[1,1].set_title('scatter')
plt.tight_layout()
plt.show()
```

Saving figures & quick customization tips

- Save: plt.savefig('figure.png', dpi=300, bbox_inches='tight').
- Figure size: plt.figure(figsize=(w,h)).
- Fonts, ticks, spines: use ax.set_xlabel(...), ax.tick_params(...), sns.despine() (if using Seaborn), or Matplotlib rcParams.

---

4) Seaborn — Graph types, descriptions, and examples

Seaborn works well with pandas.DataFrame objects and has many high-level plotting functions. We import:

```
import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

sns.set_theme(style='whitegrid')
```

Seaborn includes sample datasets that are handy for examples: tips, iris, titanic, flights, fmri.

Scatter / relational plots — sns.scatterplot, sns.relplot

Description: Scatter plots but with built-in hue, size, and style support for categorical/semi-continuous variables.

Use case: Visualizing relationships while showing categories.

Example:

```
tips = sns.load_dataset('tips')

plt.figure(figsize=(8,5))

sns.scatterplot(data=tips, x='total_bill', y='tip', hue='smoker', style='time')

plt.title('Scatter — tips dataset')

plt.show()
```

Line plot — sns.lineplot

Description: High-level line plotting with confidence intervals for repeated measures.

Use case: Time series or mean with CI across repeated measurements.

Example:

```
fmri = sns.load_dataset('fmri')

sns.lineplot(data=fmri, x='timepoint', y='signal', hue='event')
```

```
plt.title('Line plot — fmri example')

plt.show()
```

Barplot & Countplot — sns.barplot, sns.countplot

Description: Barplot shows aggregated values with error bars; countplot shows counts per category.

Use case: Category averages or counts (e.g., mean score per class).

Example:

```
sns.barplot(data=tips, x='day', y='total_bill', estimator=np.mean)

plt.title('Barplot — average bill per day')

plt.show()


sns.countplot(data=tips, x='day')

plt.title('Countplot — bills per day')

plt.show()
```

Histogram & KDE — sns.histplot, sns.kdeplot

Description: Distributions and density estimates.

Use case: Visualizing distribution shape and multimodality.

Example:

```
sns.histplot(tips['total_bill'], kde=True, bins=20)

plt.title('Histogram + KDE — total bill')

plt.show()
```

Boxplot & Violinplot — sns.boxplot, sns.violinplot

Description: Boxplots for quartiles; violinplots show density per group.

Use case: Compare distributions across categories with added density info.

Example:

```
sns.boxplot(data=tips, x='day', y='total_bill')

plt.title('Boxplot — total bill by day')

plt.show()
```

```
sns.violinplot(data=tips, x='day', y='total_bill')
plt.title('Violinplot — total bill by day')
plt.show()
```

Pairplot & Jointplot — sns.pairplot, sns.jointplot

Description: pairplot shows pairwise relationships across multiple numeric columns; jointplot gives a focused 2-variable scatter + marginals.

Use case: Quick multi-variable EDA (pairwise correlations, distributions).

Example:

```
iris = sns.load_dataset('iris')
sns.pairplot(iris, hue='species')
plt.show()

sns.jointplot(data=iris, x='sepal_length', y='sepal_width', kind='scatter')
plt.show()
```

---

lmplot (regression) & catplot

Description: lmplot fits linear models and plots regression lines + CI; catplot is a figure-level interface for categorical plots.

Example:

```
sns.lmplot(data=tips, x='total_bill', y='tip', hue='sex', height=5)
plt.show()

sns.catplot(data=tips, x='day', y='total_bill', kind='box', height=4)
plt.show()
```

Heatmap (from pivot) — sns.heatmap

Description: Pretty, annotated heatmaps from matrices or pivoted DataFrames.

Example:

```
pivot = tips.pivot_table(index='day', columns='time', values='total_bill', aggfunc='mean')
sns.heatmap(pivot, annot=True, fmt='.2f')
plt.title('Heatmap — avg bill per day/time')
```

plt.show()

Styling & palettes

Seaborn makes it easy to change overall look:

sns.set_theme(style='darkgrid')

sns.set_palette('pastel')

Common palettes: 'deep', 'muted', 'bright', 'pastel', 'dark', 'colorblind'.

---

5) Comparison: Matplotlib vs Seaborn

Aspect	**Matplotlib**	**Seaborn**
Ease of use (basic plots)	medium — needs more lines	easy — high-level functions
Aesthetic defaults	basic	polished, modern by default
Statistical plots	manual (need to compute aggregates)	built-in aggregation & CI support
Customization	excellent (low-level control)	good (but layered on Matplotlib)
Interactivity	limited (static by default; interactive backends exist)	same as Matplotlib (since it uses Matplotlib)
Performance on very large data	good but can be slow for millions of points	similar performance (built on Matplotlib)

When to choose Matplotlib: You need full control, publication-quality figures, or want to draw custom shapes/annotations.

When to choose Seaborn: Quick EDA, statistical plots, and attractive default styling with fewer lines of code.

Notes about interactivity and large data: If you need interactive dashboards or very large-scatter performance, consider Plotly, Bokeh, or Datashader (not covered here).

---

6) Resources & further reading

- Matplotlib quick start: https://matplotlib.org/stable/users/explain/quick_start.html#quick-start
- Seaborn tutorial: https://seaborn.pydata.org/tutorial/introduction.html