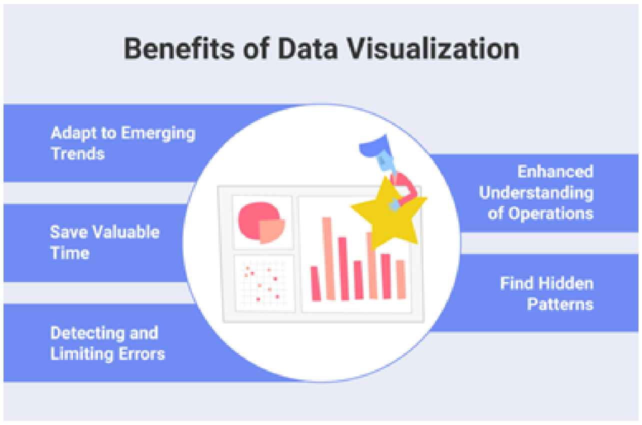
Visualisation in Python Cheat Sheet

Hamed Ahmadinia, Ph.D.
Hamed.Ahmadinia@metropolia.fi

Why Visualize Data?

Purpose:

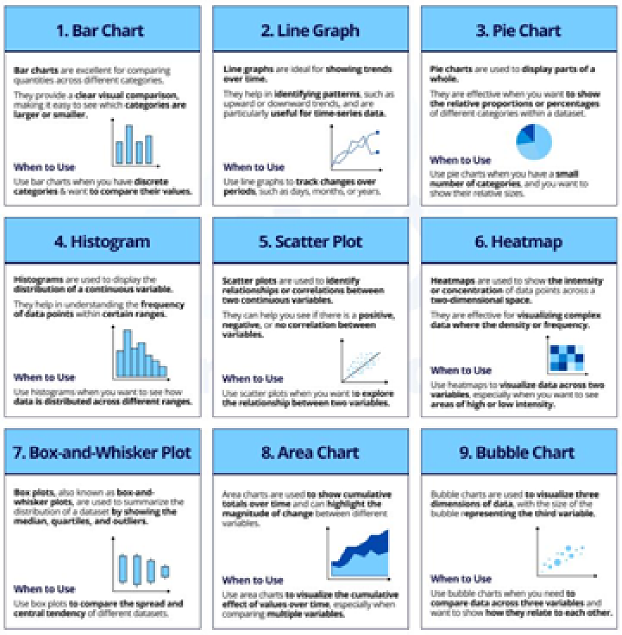
- Helps analyze and communicate data effectively.
- Identifies patterns, trends, and outliers.


Benefits of Data Visualization
Adapt to Emerging Trends
Save Valuable Time
Detecting and Limiting Errors
Enhanced Understanding of Operations
Find Hidden Patterns

Common Chart Types

Types of Visualizations:

- **Bar Chart**: Compare quantities across categories.
- **Line Plot**: Show trends over time.
- **Histogram**: Show data distribution.
- **Scatter Plot**: Visualize relationships between two variables.
- **Box Plot**: Highlight outliers and summary statistics.
- **Heatmap**: Display value intensity using colors.
- **Pie Chart**: Show proportions as slices of a whole.



1. Bar Chart

Bar charts are excellent for comparing quantities across different categories.

They provide a **clear visual comparison,** making it easy to see which **categories are larger or smaller.**

When to Use

Use bar charts when you have **discrete categories** & want **to compare their values.**

2. Line Graph

Line graphs are ideal for **showing trends over time.**

They help in **identifying patterns**, such as upward or downward trends, and are particularly **useful for time-series data.**

When to Use

Use line graphs to **track changes over periods**, such as days, months, or years.

3. Pie Chart

Pie charts are used to **display parts of a whole.**

They are effective when you want **to show the relative proportions or percentages** of different categories within a dataset.

When to Use

Use pie charts when you have a **small number of categories**, and you want to show their relative sizes.

4. Histogram

Histograms are used to display the **distribution of a continuous variable.**

They help in understanding the **frequency of data points** within **certain ranges.**

When to Use

Use histograms when you want to see how **data is distributed across different ranges.**

5. Scatter Plot

Scatter plots are used to **identify relationships or correlations between two continuous variables.**

They can help you see if there is a **positive, negative,** or **no correlation between variables.**

When to Use

Use scatter plots when you want **to explore the relationship between two variables.**

6. Heatmap

Heatmaps are used to show **the intensity or concentration** of data points across a **two-dimensional space.**

They are effective for **visualizing complex data where the density or frequency.**

When to Use

Use heatmaps to **visualize data across two variables**, especially when you want to see **areas of high or low intensity.**

7. Box-and-Whisker Plot

Box plots, also known as **box-and-whisker plots,** are used to summarize the distribution of a dataset **by showing the median, quartiles, and outliers.**

When to Use

Use box plots **to compare the spread and central tendency** of different datasets.

8. Area Chart

Area charts are used **to show cumulative totals over time** and can **highlight the magnitude of change** between different variables.

When to Use

Use area charts **to visualize the cumulative effect of values over time**, especially when comparing **multiple variables.**

9. Bubble Chart

Bubble charts are used **to visualize three dimensions of data**, with the size of the bubble **representing the third variable.**

When to Use

Use bubble charts when you need **to compare data across three variables** and want to show **how they relate to each other.**

Matplotlib Basics

Matplotlib Overview: Matplotlib is a powerful library for creating static, animated, and interactive visualizations in Python.

Key Functions:

- `plt.plot()` - Creates a line plot.
- `plt.scatter()` - Creates a scatter plot.
- `plt.bar()` - Generates a bar chart.
- `plt.hist()` - Displays a histogram.
- `plt.xlabel()` / `plt.ylabel()` - Adds axis labels.
- `plt.title()` - Adds a title to the plot.
- `plt.grid()` - Adds a grid for better readability.

Basic Line Plot Example:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 50]

plt.plot(x, y, marker='o', linestyle='-',
         color='b', label="Growth")
plt.xlabel("X-axis (Time)")
plt.ylabel("Y-axis (Value)")
plt.title("Basic Line Plot Example")
plt.legend()
plt.grid(True)
plt.show()
```

Creating Subplots

Subplots Overview: Subplots allow multiple plots in one figure for side-by-side comparisons.

Syntax:

- `fig, ax = plt.subplots(rows, cols)` - Creates a grid of subplots.
- `ax[index].plot()` - Plots on a specific subplot.

Example - Basic Subplots:

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2, figsize=(10,4))

# First subplot: Line plot
ax[0].plot([1, 2, 3, 4], [10, 20, 25, 30], 'r')
ax[0].set_title("Line Plot")

# Second subplot: Histogram
ax[1].hist([10, 20, 25, 30, 40, 50], bins=5, color='g')
ax[1].set_title("Histogram")

plt.show()
```

Seaborn for Statistical Plots

Seaborn Overview: Seaborn is a high-level library based on Matplotlib, making statistical visualization simpler.
Key Functions:

- `sns.pairplot()` - Shows relationships between numeric features.
- `sns.histplot()` - Displays distributions.
- `sns.boxplot()` - Visualizes data distribution and outliers.

Pair Plot Example:

```
import seaborn as sns
import pandas as pd

df = pd.read_csv("data.csv")
sns.pairplot(df, hue="category", diag_kind="kde")
plt.show()
```

Heatmaps - Correlation Matrix

Heatmaps Overview: Heatmaps visualize feature relationships using color gradients.
Example - Correlation Matrix:

```
import numpy as np
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```

Residual Plots - Model Fit

Residual Plots Overview: Residual plots help assess the goodness of fit of a regression model.
Key Concepts:

- **Residuals:** Difference between actual and predicted values.
- **Ideal Fit:** Residuals should be randomly scattered around zero.
- **Non-Linear Fit:** Patterns in residuals suggest a non-linear model is needed.

Example:

```
import statsmodels.api as sm
import seaborn as sns

X = df[['feature1', 'feature2']]
y = df['target']
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
residuals = model.resid

sns.residplot(x=model.fittedvalues, y=residuals,
        lowess=True, line_kws={'color': 'red'})
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```

Website: `ahmadinia.fi`