Advanced Plotting with Matplotlib & Seaborn

Data Visualization Course – Lecture 3

Dr. Muhammad Sajjad

R.A: Imran Nawar

October 2024

Overview

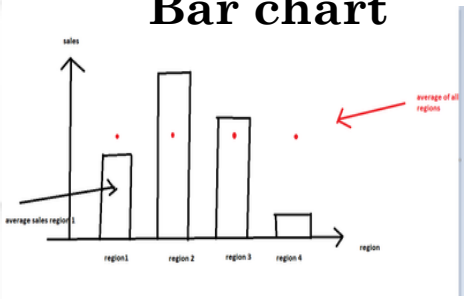
- Basic Plotting Techniques (Recap)
- Importance of Advanced Plotting Techniques
- **Data Preprocessing for Visualization**
  - Data Cleaning
  - Handling Missing values
  - Handling Outliers
  - Data Transformation
- **Advanced Plotting Techniques**
  - Subplots, Heatmaps, 3D Plots, Network Graphs, Choropleth Maps, Contour Plots
- Creating Subplots in Matplotlib
- Colormaps and Colorbars in Matplotlib
- Creating Heatmaps in Seaborn
- **Customizing and Enhancing Visualizations**
  - Adding labels, titles, and legends
  - Customizing font styles and sizes
  - Customizing legends
- Combining Matplotlib and Seaborn for Complex Visualizations.
- **Coding:** Practical coding examples for the topics

Basic Plotting Techniques (Recap)

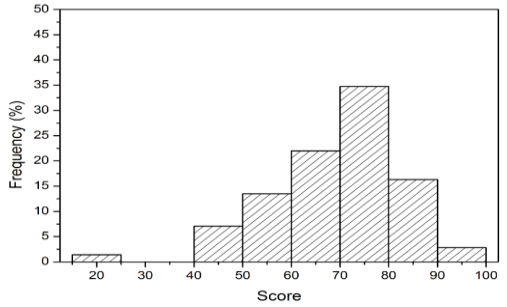
Basic Plotting Techniques in Data Visualization

1. **Line Plot:** Visualize trends over time or continuous data
2. **Bar Plot:** Comparing quantities across categories
3. **Histogram:** Display the frequency distribution of a variable.
4. **Scatter Plot:** Show relationships between two variables.
5. **Pie Chart:** Show proportions within a whole.

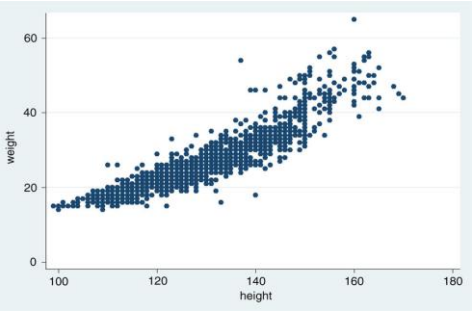
Bar chart



Histogram



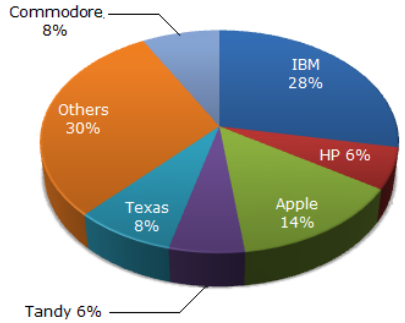
Scatter Plot



Python Libraries for Data Visualization:

1. **Matplotlib**
   - Low-level, highly customizable plotting library.
   - Foundation for many other libraries.
2. **Pandas**
   - Convenient for quick visualizations directly from DataFrames.
   - Ideal for simple exploratory plots.
3. **Seaborn**
   - Visualization library based on Matplotlib.
   - It makes the complicated plot simpler to create.
4. **Plotly**
   - Creates interactive, web based visualizations
   - We'll study it in later lectures.

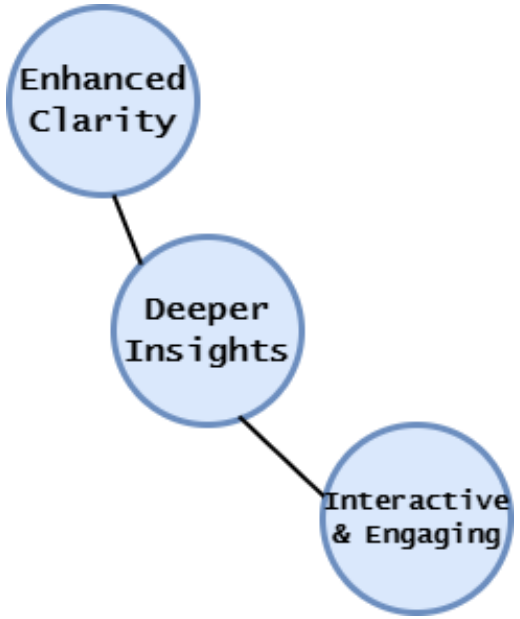
Pie Chart

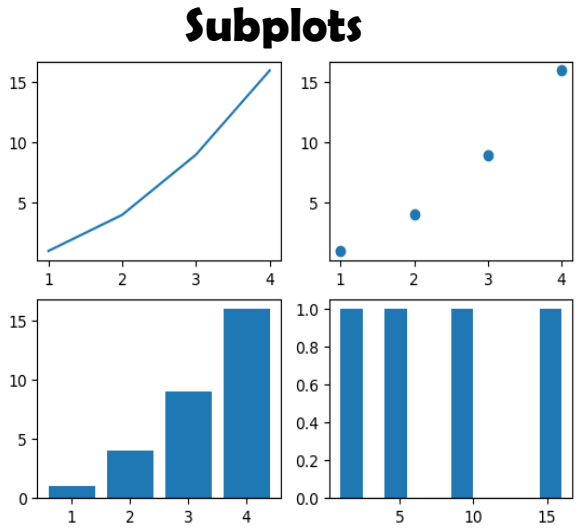


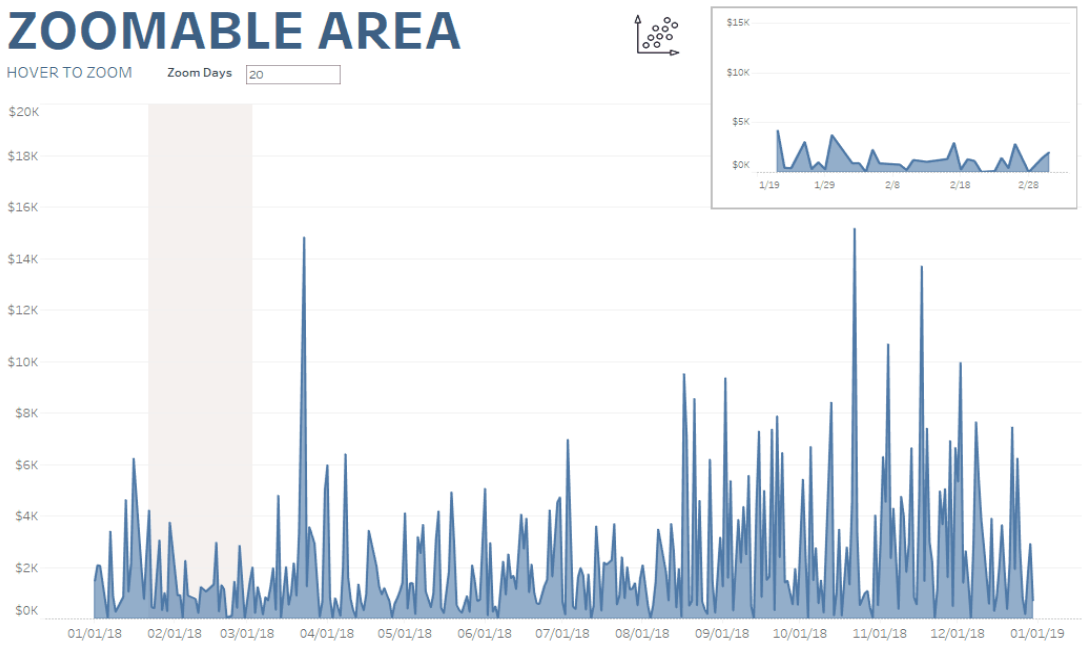
The pie chart shows the distribution of New York market share by value of different computer companies in 2005.

Importance of Advanced Plotting Techniques

- **Enhanced Clarity:**
  - Present complex data in a way that's easier to understand by organizing it efficiently.
    - **Example**: Using subplots to compare multiple datasets side by side.
- **Deeper Insights:**
  - Visualize multiple dimensions and uncover hidden patterns in the data.
    - **Example:** Heatmaps to reveal correlations in large datasets.
- **Interactive & Engaging:**
  - Facilitate exploration and interaction with data.
    - **Example**: Creating zoomable or clickable plots for detailed analysis







Data Preprocessing for Visualization

Data Cleaning:

- Process of identifying and correcting errors in the data.

➢ Importance for Effective Visualization

- Ensures accurate and meaningful visualizations
- Reduces the risk of misinterpretation and false conclusions
- Improves the reliability of insights derived from the data.

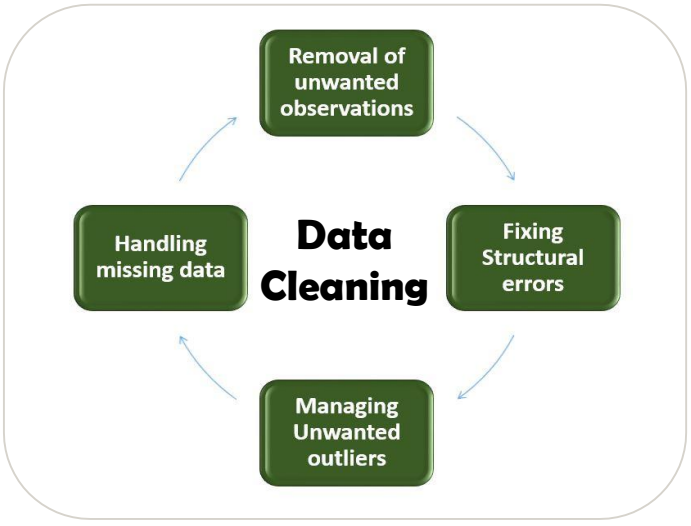
➢ Common Data Cleaning Techniques:

- **Remove Duplicates**
  - Use pandas `drop_duplicates() function.`
  - `drop_duplicates()` method removes duplicate rows.
- **Remove Irrelevant Data**
  - Identify and remove data that doesn't contribute to your analysis.
  - Improves processing speed and clarity of results.
- **Fix Structural Error**
  - **Handling inconsistent formatting:**
    - Apply string methods like `lower(), strip(),` or custom functions.
  - **Standardizing units:**
    - Convert all measurements to a consistent unit system.
    - **Example:** `df['height_cm'] = df['height_inches'] * 2.54`
- **Handling Missing Values** (covered in detail in next slide)

A sample messy dataset

color	director_name	duration	gross	movie_title	anguage	country	budget	title_year	imdb_score
Color	Martin Scorsese	240	116866727	The Wolf of Wall StreetÂ	English	USA	100000000	2013	8.2
Color	Shane Black	195	408992272	Iron Man 3Â	English	USA	200000000	2013	7.2
color	Quentin Tarantino	187	54116191	The Hateful EightÂ	English	USA	44000000	2015	7.9
Color	Kenneth Lonergan	186	46495	MargaretÂ	English	usa	14000000	2011	6.5
Color	Peter Jackson	186	258355354	The Hobbit: The Desolation of SmaugÂ	English	USA	225000000	2013	7.9
	N/A	183	330249062	Batman v Superman: Dawn of JusticeÂ	English	USA	250000000	202	6.9
Color	Peter Jackson	-50	303001229	The Hobbit: An Unexpected JourneyÂ	English	USA	180000000	2012	7.9
Color	Edward Hall	180		RestlessÂ	English	UK		2012	7.2
Color	Joss Whedon	173	623279547	The AvengersÂ	English	USA	220000000	2012	8.1
Color	Joss Whedon	173	623279547	The AvengersÂ	English	USA	220000000	2012	8.1
	Tom Tykwer	172	27098580	Cloud AtlasÂ	English	Germany	102000000	2012	-7.5
Color	Null	158	102515793	The Girl with the Dragon TattooÂ	English	USA	90000000	2011	7.8
Color	Christopher Spencer	170	59696176	Son of GodÂ	English	USA	22000000	2014	5.6
Color	Peter Jackson	164	255108370	The Hobbit: The Battle of the Five ArmiesÂ	English	New Zealand	250000000	2014	7.5
Color	Tom Hooper	158	148775460	Les MisÃ©rablesÂ	English	USA	61000000	2012	7.6
Color	Tom Hooper	158	148775460	Les MisÃ©rablesÂ	English	USA	61000000	2012	7.6

Year	City	Amount
1990	New York City	$1,123,456.00
1995-96		2.2 mil
2000s	NYC	No data
2020	New_York	50000000



Data Preprocessing for Visualization Handling Missing Values

Strategies for dealing with missing data:

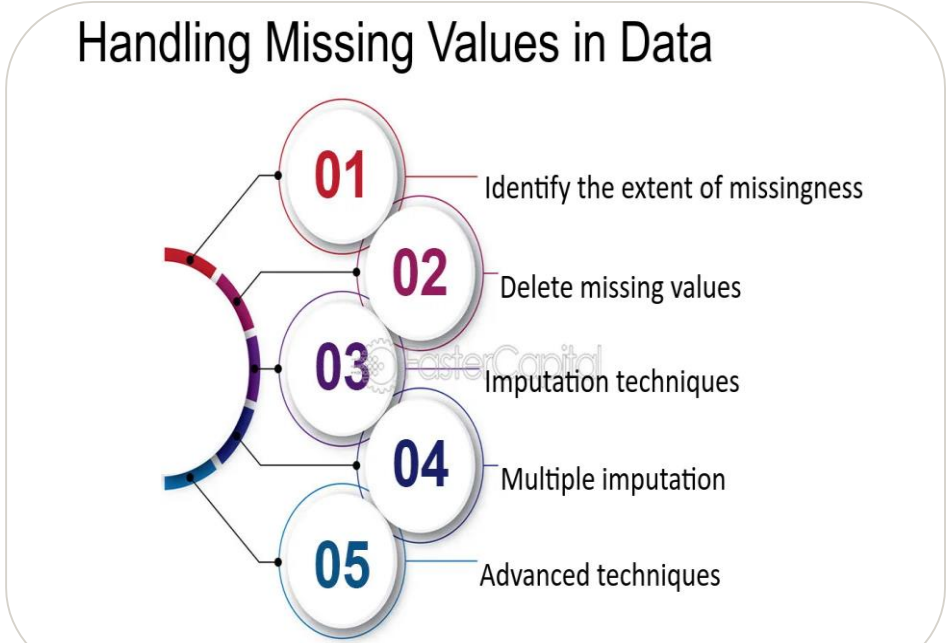
- **Deletion Methods**
  - **Listwise Deletion:**
    - Removes entire rows that contain any missing values.
  - **Pairwise Deletion:**
    - Removes only the specific missing data points while retaining other available information in a row.
- **Imputation Techniques**
  - **Mean/median imputation:**
    - Replaces missing values with the mean or median of the available data.
  - **K-Nearest Neighbors (KNN) imputation:**
    - Fills missing values by finding the k-nearest neighbors and averaging their values. This technique considers the relationship between data points.
- **Advanced Techniques**
  - **Machine Learning models:**
    - Use models such as regression, decision trees, or deep learning to predict and impute missing values based on patterns in the data.

```
In [1]: import pandas as pd

        dataset = pd.read_csv("C:/Users/Admin/Desktop/Blog/Missing values/data.csv")
        dataset
```

Out[1]:

	Height	Weight	Country	Place	Number of days	Some column
0	12.0	35.0	India	Bengaluru	1.0	NaN
1	NaN	36.0	US	New York	2.0	NaN
2	13.0	32.0	UK	London	NaN	NaN
3	15.0	NaN	France	Paris	4.0	NaN
4	16.0	39.0	US	California	5.0	12.0
5	NaN	NaN	NaN	Mumbai	NaN	NaN
6	NaN	NaN	NaN	NaN	6.0	NaN



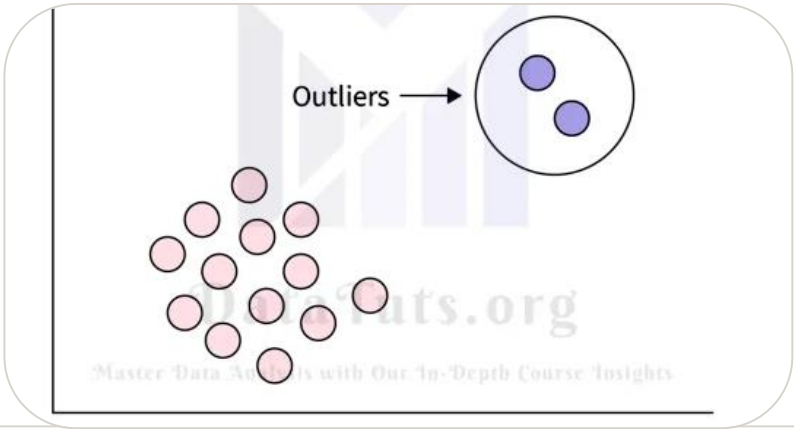
Data Preprocessing for Visualization Handling Outliers

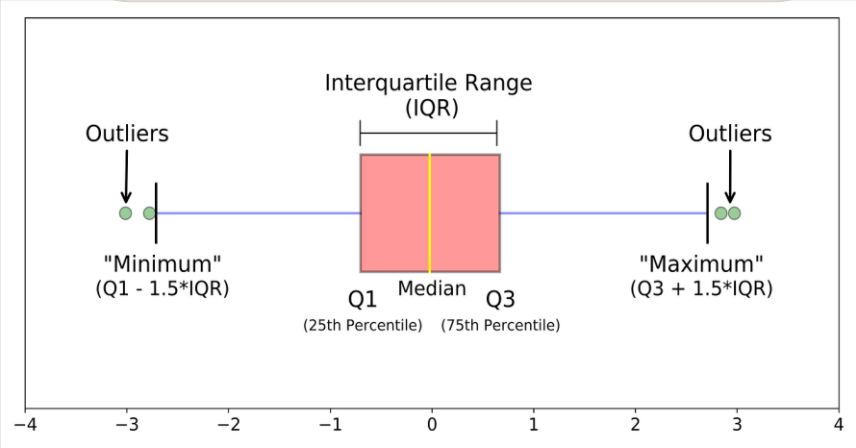
Identifying Outliers:

- **Statistical methods**: Use the Interquartile Range (IQR) to find outliers, Data points beyond 1.5 times the IQR are potential outliers.
- **Visual methods:** Tools like box plots or scatter plots help visually spot outliers, as these points will appear far from most data.

Treating Outliers:

- **Removal**: In some cases, outliers may need to be completely removed from the dataset.
- **Transformation**: Use *log transformation* to reduce the impact of outliers by compressing the data range.
- **Capping (winsorization):** Replace extreme outliers with more reasonable values, such as the 95th percentile, to reduce their impact while preserving data integrity.





Impact of Missing Data and Outliers on Visualizations:

Skewed Distributions and Misleading Trends:

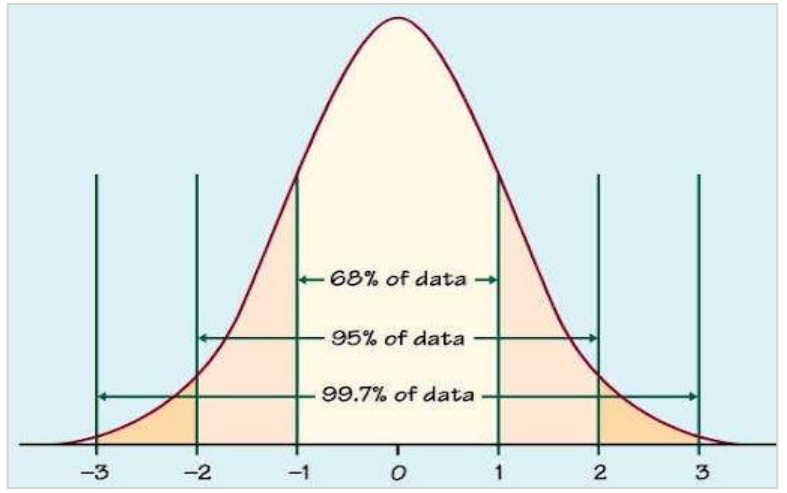
- Outliers and missing data can skew the results, creating charts that misrepresent the true nature of the data.

Inaccurate Measures:

- Measures like the mean or correlation can become distorted by extreme outliers.

Bias in Machine Learning Models:

- Outliers or improperly handled missing data can bias models, reducing accuracy and generalization performance.



Data Preprocessing for Visualization

Data Transformation:

- Converting raw data into a format more suitable for analysis and visualization.

Data Transformation Methods:

Normalization:

- Scaling features to a fixed range, typically between 0 and 1, so no feature dominates the analysis.
- Formula: $\frac{x-\min(x)}{\max(x)-\min(x)}$
- Helpful when you want all features to have equal weight, especially in machine learning models.

```
from sklearn.preprocessing import MinMaxScalar
```

Standardization:

- Scales features to have a mean of 0 and a standard deviation of 1.
- Formula: $(x - mean(x))/std(x)$

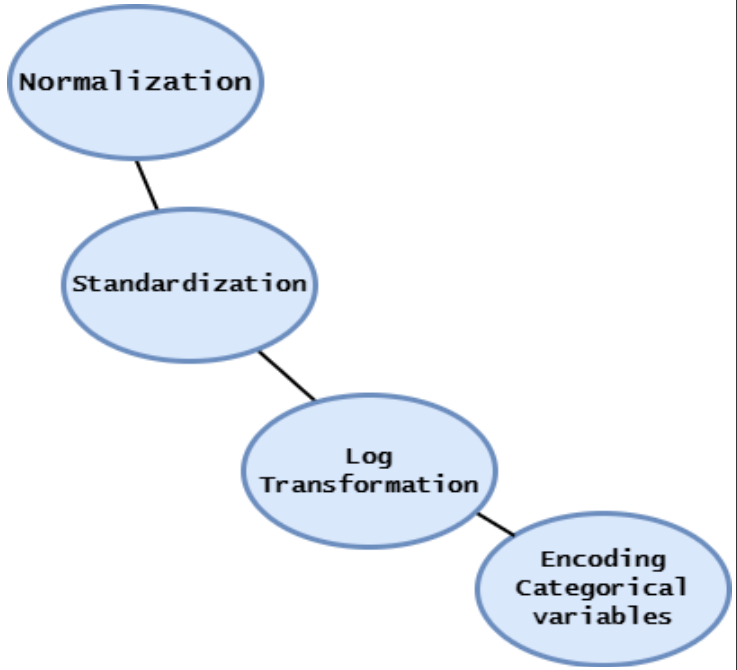
Log transformation:

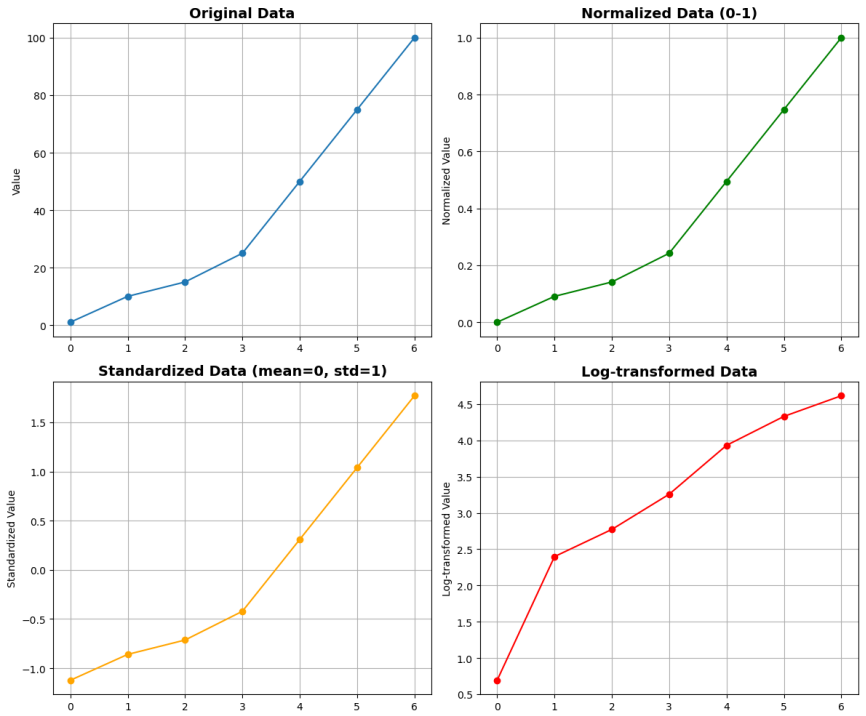
- Reduces the effect of extreme values by compressing the range of data, Useful for handling skewed distributions.
- Formula: $y = \log(x)$

Encoding categorical variables

- **One-hot encoding:** Creates binary columns for each category
- **Label encoding:** Assigns a unique integer to each category.

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
```





Advanced Plotting Techniques

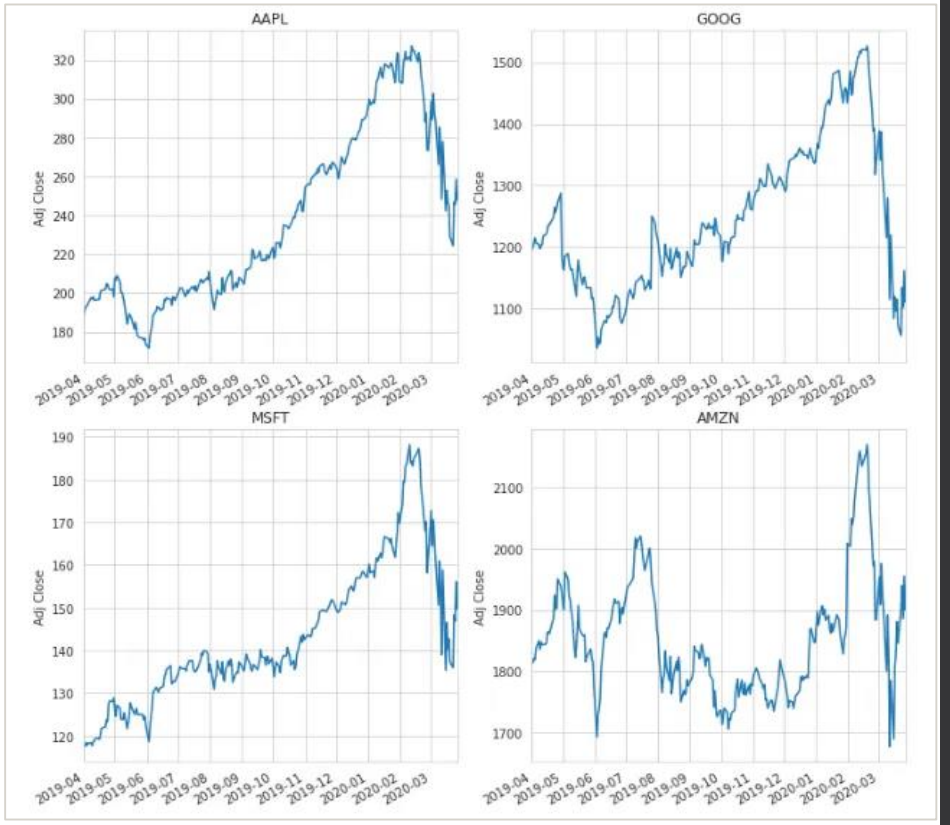
- Advanced plotting techniques offer sophisticated visualizations for exploring complex, multidimensional datasets.
- These methods reveal patterns, correlations, and insights that simpler charts may overlook.

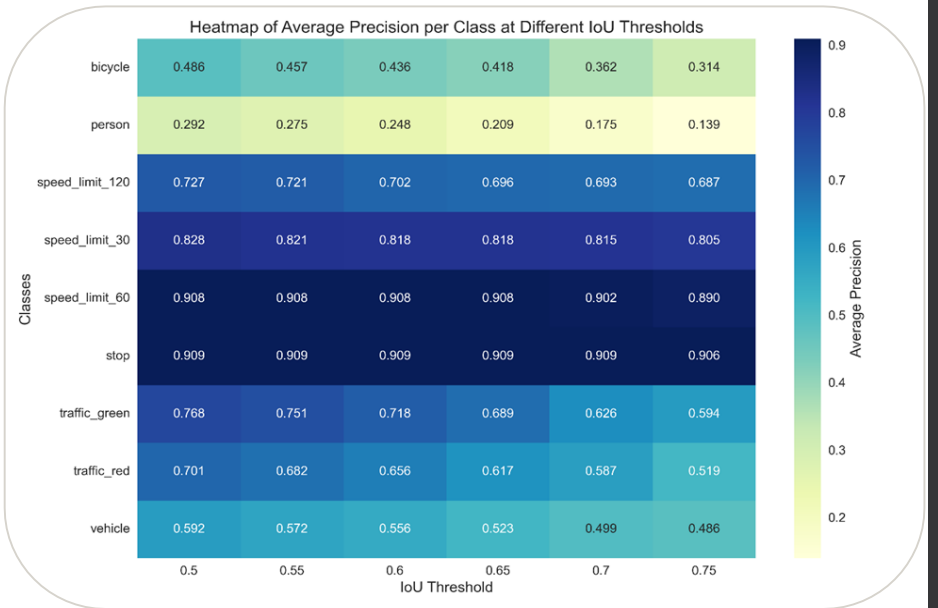
Subplots (multiple plots in one figure):

- **Use case:** Comparing different visualizations side by side.
- **Example:**
  - Line plots comparing stock prices of multiple companies over time.
- **Features:**
  - Enables comparison of multiple visualizations in a single figure.
  - Enhances clarity reduces clutter.
  - Allows for easy identification of trends and patterns across different datasets.

Heatmaps:

- **Use case:** Represent data points in a matrix format, where colors indicate magnitude or intensity
- **Example:** Visualizing correlation between variables in a dataset.
- **Features:**
  - Colors help easily identify high or low correlation points.
  - Ideal for visualizing correlations, confusion matrices, or any grid-based data.
  - Customizable color palettes for better clarity.

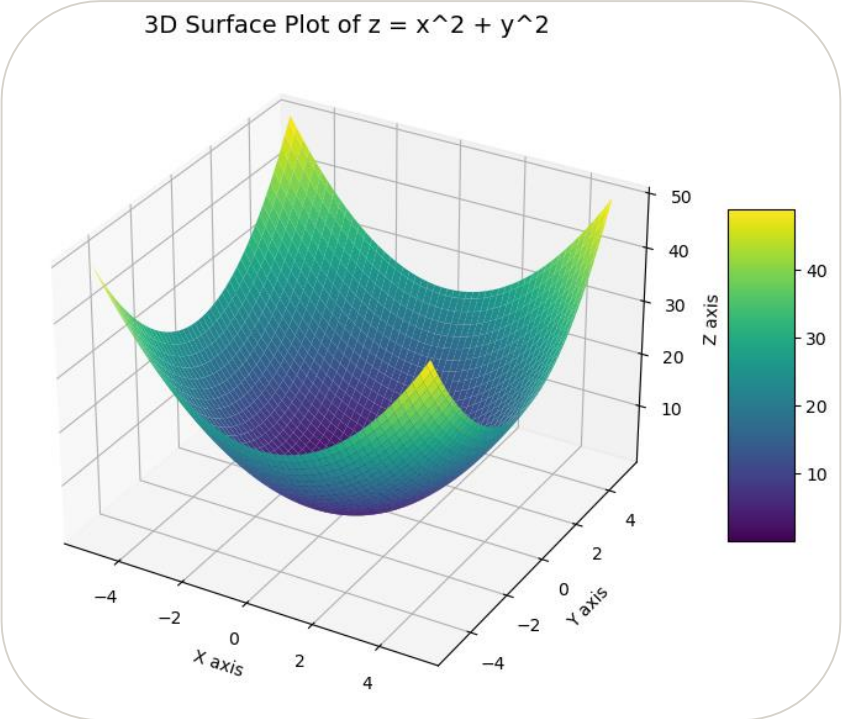




Advanced Plotting Techniques

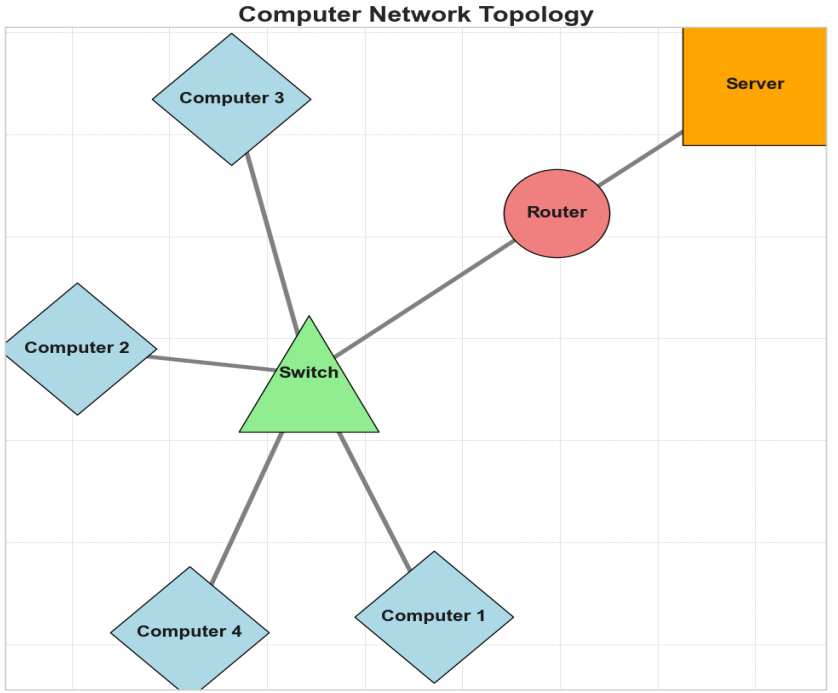
3D Plots:

- **Use case:**
  - Visualizing data in three dimensions to offer a deeper understanding of complex relationships.
- **Example**:
  - Visualize mathematical function. (e.g., $z^2 = x^2 + y^2$) using a 3D surface plot.
- **Features:**
  - Adds an extra dimension (z-axis) for visualizing spatial relationships.
  - Commonly used for surface plots, 3D scatter plots and contour plots.
  - Helps identify relationships between variables that are hard to see in 2D.
  - Allows rotation and interaction for better data exploration.



Network Graphs:

- **Use case:**
  - Visualize relationships and connections between entities in complex systems.
- **Example:**
  - Computer network topology visualization
- **Features:**
  - Ideal for analyzing and visualizing complex relationships and interconnections.
  - Can highlight clusters, hubs, and central nodes in a network.
  - Useful for understanding relationships in social, biological, and communication networks.

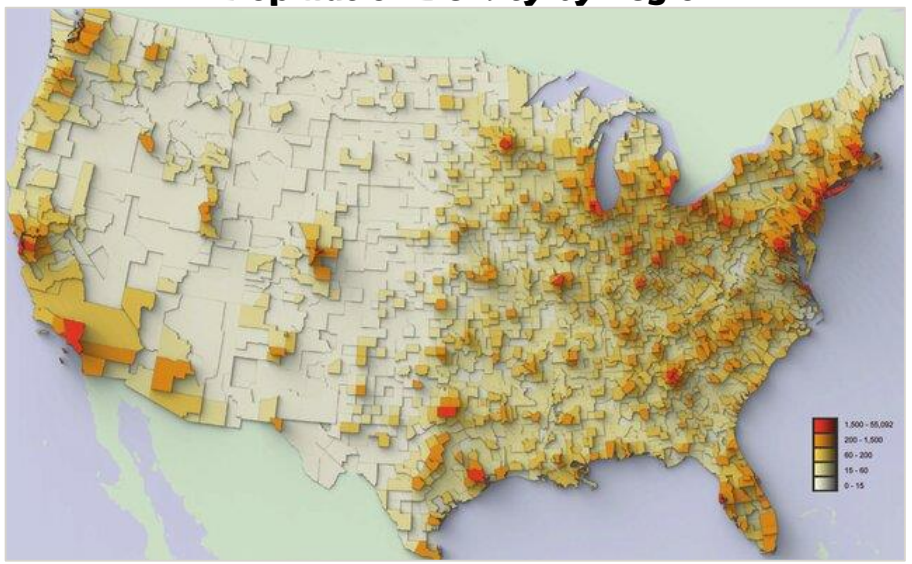


Advanced Plotting Techniques

Choropleth Maps:

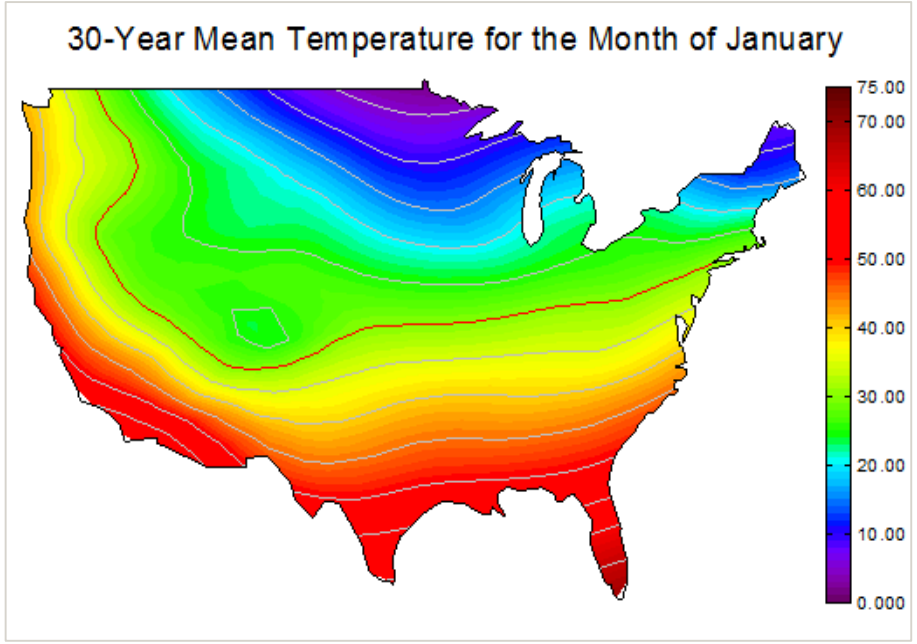
- **Use case:**
  - Displaying data distribution across geographic regions.
- **Example:**
  - Visualizing population density across different countries or states.
- **Features:**
  - Uses color gradients to represent data values over geographic areas.
  - Ideal for showing data trends across spatial regions.
  - Customizable color palettes to highlight key data points, allowing for clearer differentiation between regions.

Choropleth Map of U.S. Population Density by Region



Contour Plots:

- **Use Case**:
  - Often used to represent 3D data in two dimensions using contour lines to show different levels.
- **Example**:
  - Visualizing weather patterns like pressure or temperature levels.
- **Features**:
  - Contour lines indicate areas of equal value, helping to identify trends and patterns in the data.
  - Useful for visualizing gradients and changes in continuous data.
  - Can display complex relationships between three variables on a 2D plane
  - Helps in understanding topographical information in fields like meteorology and geography.



Creating Subplots in Matplotlib

Subplots (multiple plots in one figure):

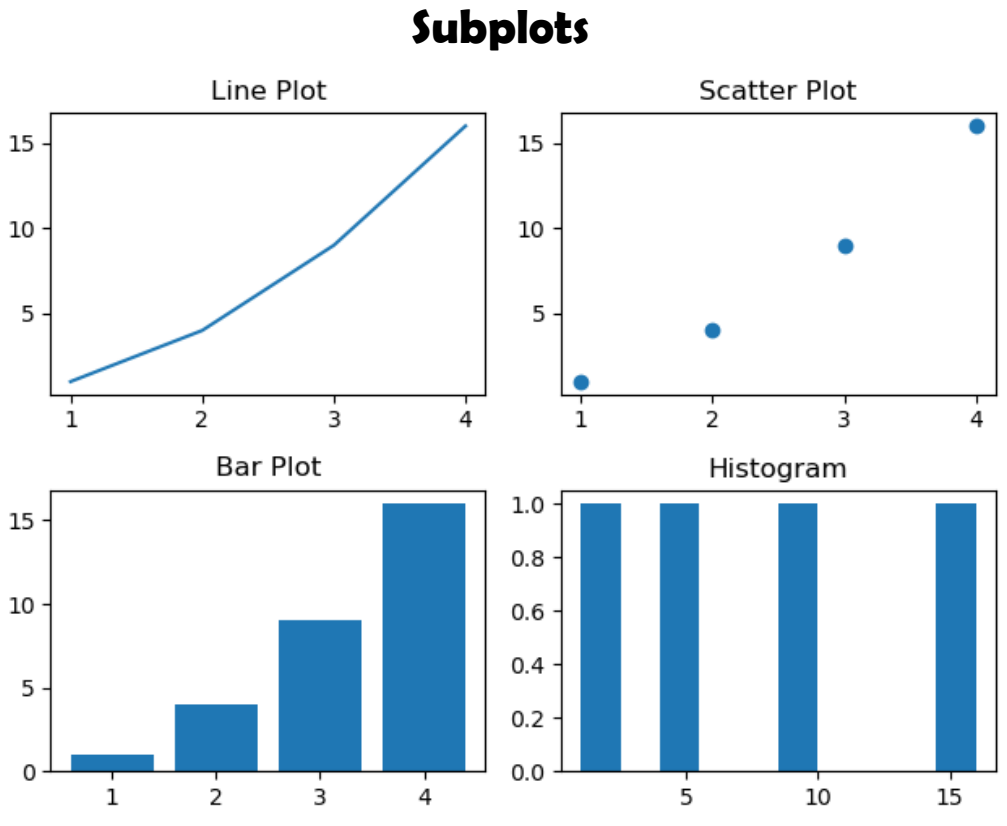
- **Use Case:** Comparing different visualizations side by side.
- **Code:**

```
x = [1,2, 3,4]
y = [1, 4, 9, 16]

fig, axs = plt.subplots(2, 2)
axs[0, 0].plot(x, y), axs[0, 0].set_title('Line Plot')
axs[0, 1].scatter(x, y), axs[0, 1].set_title('Scatter Plot')
axs[1, 0].bar(x, y), axs[1, 0].set_title('Bar Plot')
axs[1, 1].hist(y), axs[1, 1].set_title('Histogram')
plt.tight_layout()

plt.show()
```

- **Best practices for readability:**
  - Use appropriate spacing (`plt.tight_layout()`) to avoid overlapping elements.
  - Add descriptive titles and labels to each subplot



Colormaps and Colorbars in Matplotlib

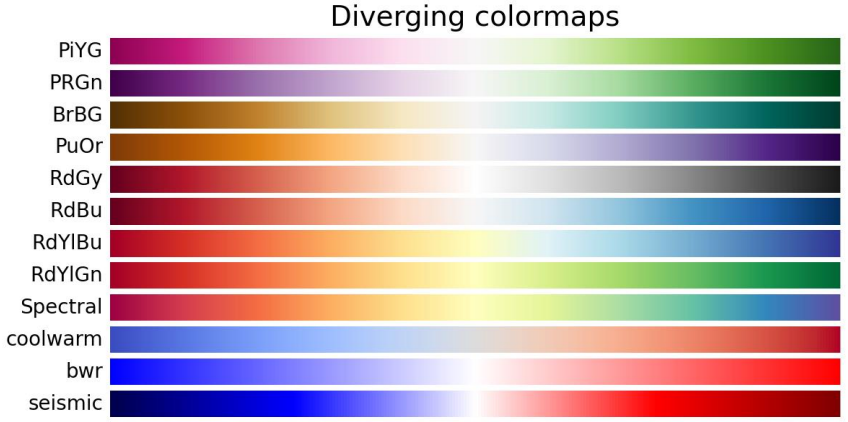
- Colormaps enhance data visualizations by using color to represent values.
- Commonly used in Heatmaps, bar plots, scatter plots, and colorbars.
- Matplotlib has a number of built-in colormaps accessible via *`matplotlib.colormaps`*

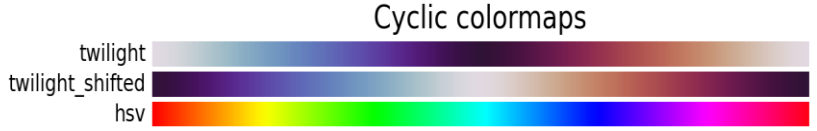
Types of Colormaps:

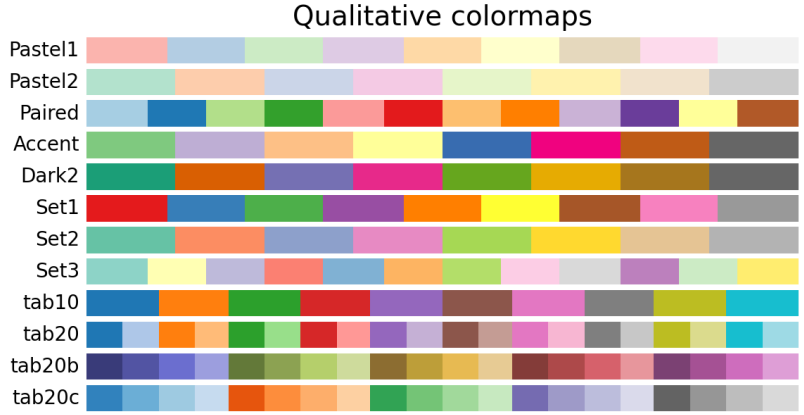
- **Sequential:**
  - Best for: Data with a meaningful order (e.g., temperature, population density).
  - **Example:** viridis, plasma, inferno, magma
  - **Usage**: Ideal for visualizing continuous data that moves from low to high values.
- **Diverging:**
  - Best for: Data deviating from a central point (e.g., profit/loss, temperature anomalies).
  - **Examples:** coolwarm, Spectral, PiYG
  - **Usage**: Used to highlight both positive and negative deviations from a midpoint.
- **Cyclic:**
  - Best for: Data that wraps around (e.g., angels, seasons, time of day)
  - **Examples:** twilight, hsv
- **Qualitative (Categorical):**
  - Best for: Discrete categories without inherent order
  - **Examples:** Set1, Set2, Paired
- **Custom Colormaps:** Users can also create their own colormaps to match specific needs.

Colorbars:

- Used to interpret the mapping between data values and colors.
- Matplotlib makes it easy to add colorbars alongside heatmaps or scatter plots to improve clarity.







Creating Heatmaps in Seaborn

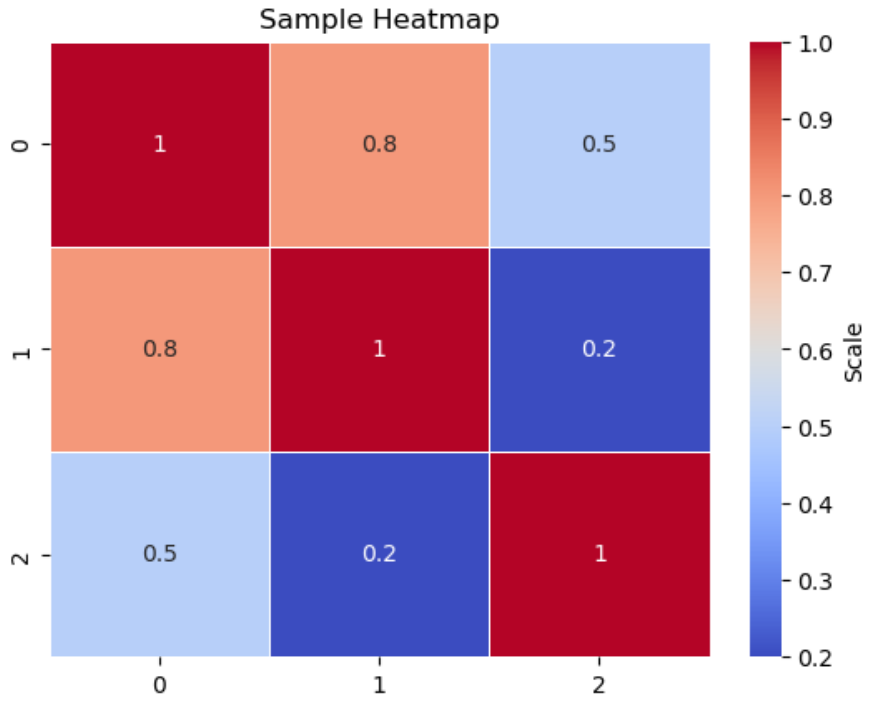
- **Use case:**
  - Represent data points in a matrix format.
  - Colors indicate magnitude or intensity of values.
  - Ideal for visualizing correlations, confusion matrices, or any grid-based data.
- **Code:**

```
import seaborn as sns
import matplotlib.pyplot as plt

data = [[1, 0.8, 0.5], [0.8, 1, 0.2], [0.5, 0.2, 1]]

sns.heatmap(data, annot=True, cmap='coolwarm',
linewidths=0.5, cbar_kws={'label': 'Scale'})
plt.title('Sample Heatmap')
plt.show()
```

- **Best Practices for Readability:**
  - **Annotate Values:** Use `annot=True` to display data values directly on the heatmap.
  - **Color Maps**: Select colormaps that improve contrast and interpretation (coolwarm, viridis, etc.).
  - **Axis Labels:** Add clear and descriptive labels for both axes.
  - **Figure Size:** Adjust the size to prevent overcrowding `(plt.figure(figsize=(8,6)))`



Customizing and Enhancing Visualizations

Why Customization is important:

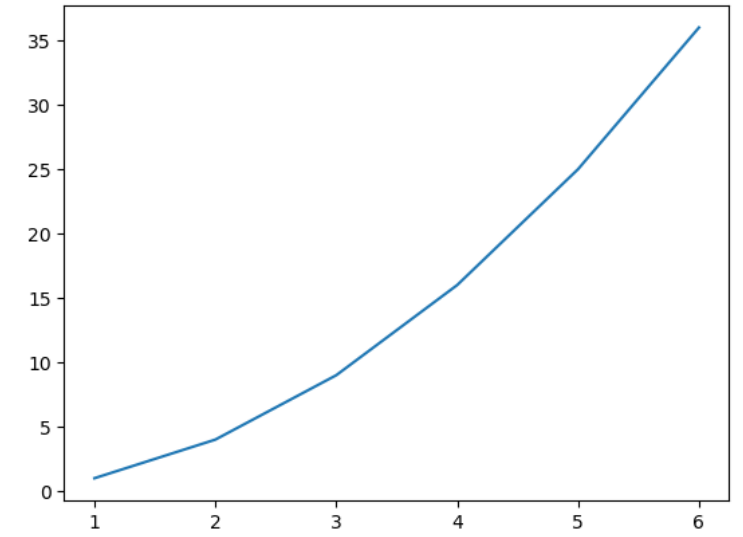
- **Enhances clarity**
  - Helps highlight key data points or trends
- **Improves readability**
  - Makes data easier to interpret at a glance
- **Tailors plots to your audience**
  - Adapts visualization style to viewer preferences and expertise

Adding Labels, Titles, and Legends

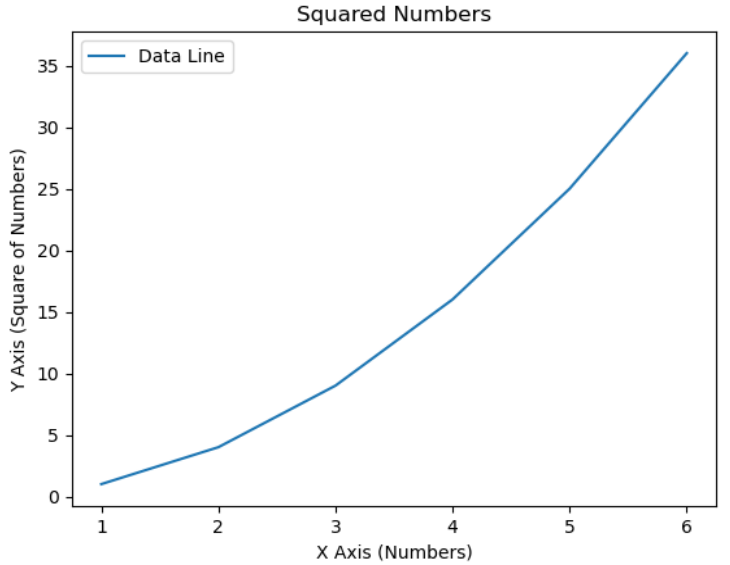
A graphs can be self-explanatory, if it have a title to the graph, labels on the axis, and a legend that explains what each line is can be necessary.

- **Best Practices:**
  - Provides clear, concise, and informative labels
  - Provide meaningful titles
- **Functions:**
  - `plt.xlabel()`, `plt.ylabel()`, and `plt.title()`
  - Use `fontdict` to adjust fonts (size, style, weight)
- **Legend Placement**:
  - Add a legend using `plt.legend()`
  - Position effectively (`loc` parameter)

Simple Line Plot without any customization



Simple Line Plot with labels, title and legend



Customizing and Enhancing Visualizations

Customizing Font Styles and Sizes

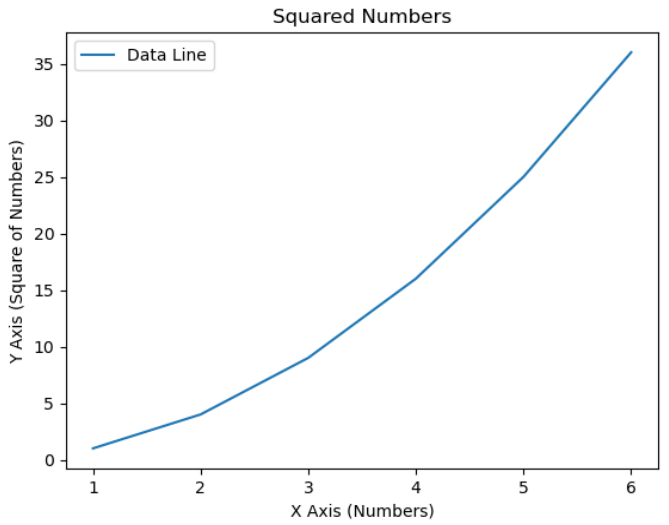
- Customize fonts to improve readability
  - Use `fontdict` to adjust fonts (size, style, weight)
- **Example:**
  - Use `fontstyle='italic'` for emphasis
  - Use `fontsize=14` for larger titles

Customizing Legends:

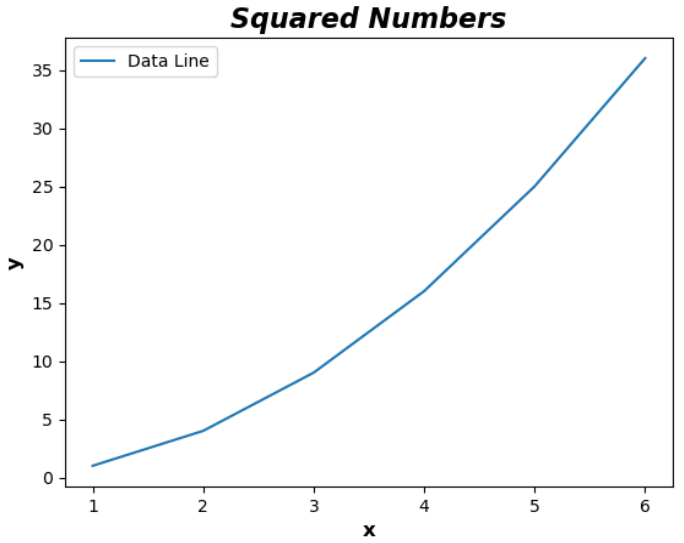
A legend is an area describing the elements of the graph.

- **Effective Legend Tips:**
  - Avoid blocking important data
  - Make sure the legend is clear and easy to read
- **Customizing Legends with `plt.legend()`:**
  - **Parameters:**
    - `loc:` Position the legend (e.g., 'upper right', 'lower left').
    - `fontsize:` Control font size of the legend text.
    - `frameon=False:` Remove the box frame around the legend.

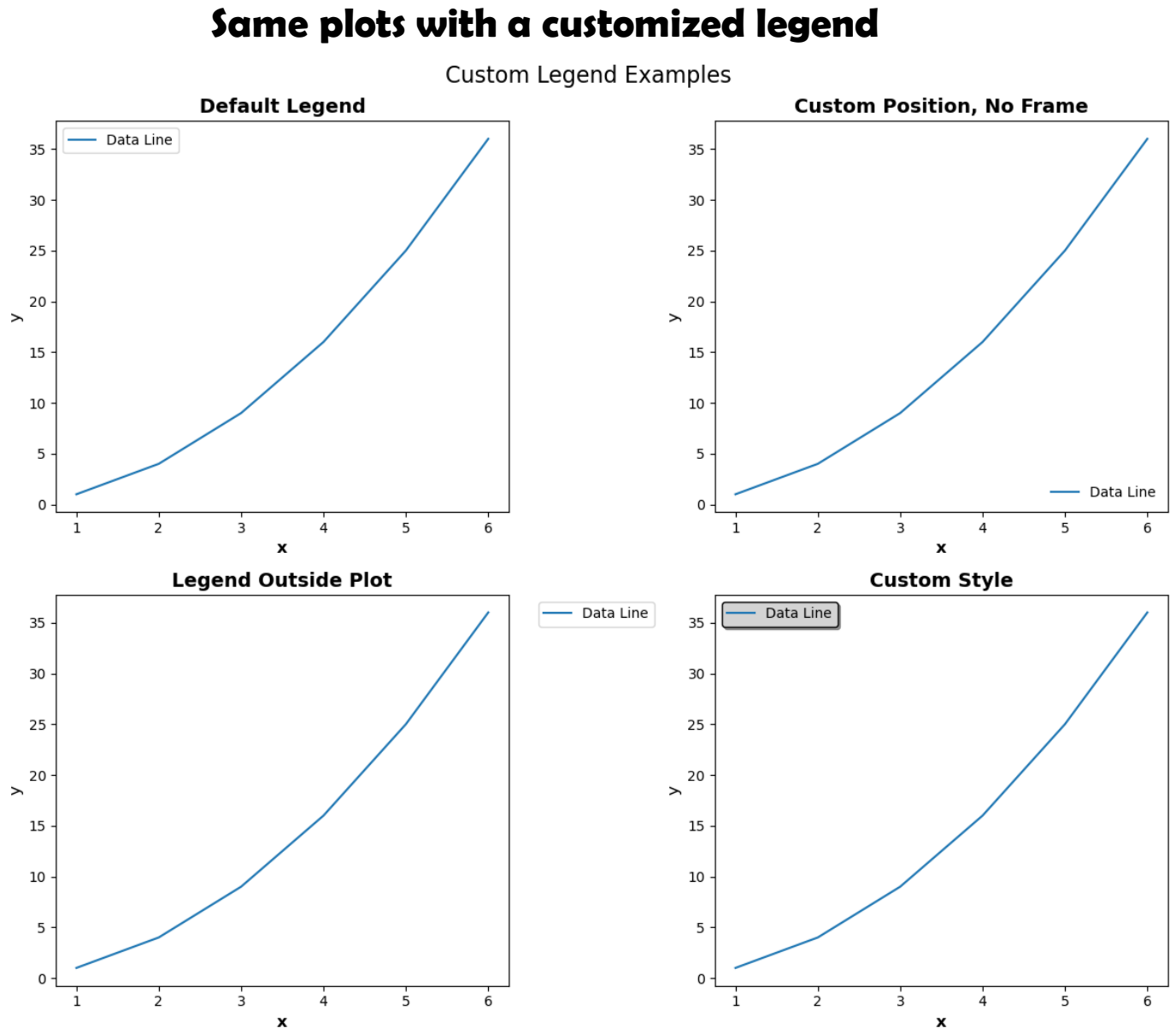
Plot with labels and title with default font style



Plot with labels and title in custom font style



Customizing Legends



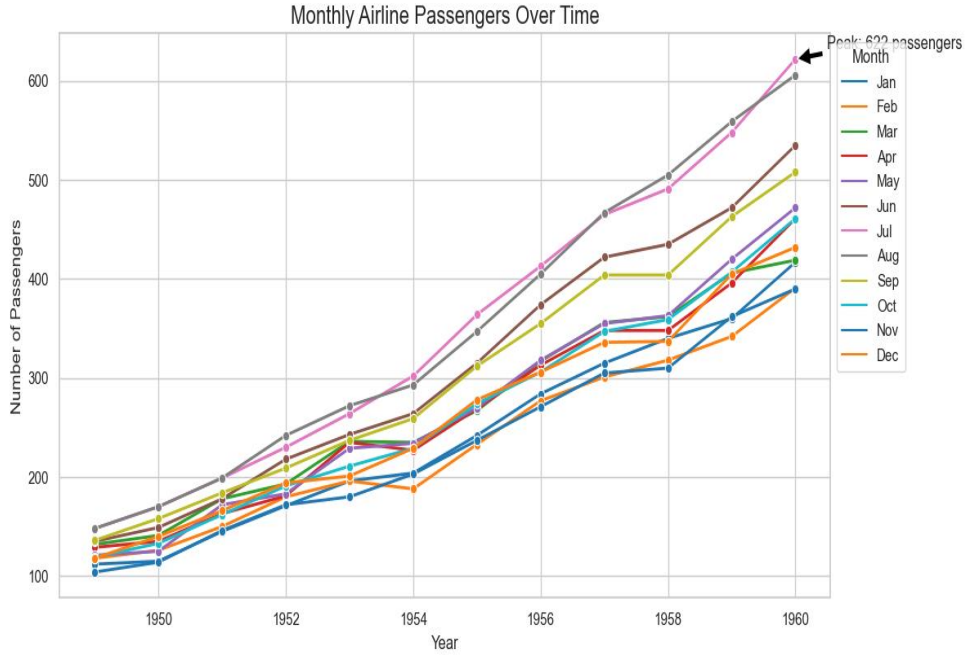
Combining Matplotlib and Seaborn for Complex Visualizations

Why Combine?

- **Matplotlib**:
  - Low-level control and customization
  - Fine-grained control over every aspect of the plot
- **Seaborn**:
  - High-level interface and statistical visualizations
  - Built-in themes for attractive plots

Key Benefits:

1. Leveraging the strengths of both libraries.
2. Enhanced aesthetics with Seaborn's styles
3. Statistical plots from Seaborn with Matplotlib's flexibility
4. Create complex, multi-layered visualizations



Case study

Analyzing Flight Data

- In this case study, we analyze the **'flights'** dataset, which records monthly airline number of passenger from 1949 to 1960.
- This dataset has three variables (year, month, and number of passengers)
- We visualize the monthly airline passenger trends over time, combining Seaborn's statistical capabilities with Matplotlib's fine-tuned control to create a multi-layered visualization.

Summary and Next Steps

- **Key Takeaways:**
  - **Importance of Advanced Plotting Techniques:**
    - Enhance clarity and insight by utilizing advanced visualizations that go beyond basic charts.
  - **Data Preprocessing**
    - Ensure clean, reliable data by handling missing values and outliers.
    - Apply transformations for more effective and accurate visual representation.
  - **Advanced Plot Types**
    - Explore various advanced plots such as subplots, heatmaps, 3D plots, network graphs, choropleth maps, and contour plots.
    - Utilize Matplotlib for creating subplots, color maps and detailed plot layouts.
    - Utilize Seaborn to implement visually appealing and informative Heatmaps.
  - **Visualization Enhancement**
    - Add meaningful titles, labels, and legends to improve comprehension.
    - Combine Matplotlib and Seaborn to create complex, multi-layered visualization.
    - Customizing font styles, sizes, and legend appearances to refine your plots.
- **Next Steps:** Apply these techniques to your own datasets.
- **Next Lecture:** Visualizing and Analyzing Time Series Data.

Thank You