# Visualizing Numeric & Categorical Data

Data Visualization Course – Lecture 2

Dr. Muhammad Sajjad

R.A: Imran Nawar

September, 2024

# Overview

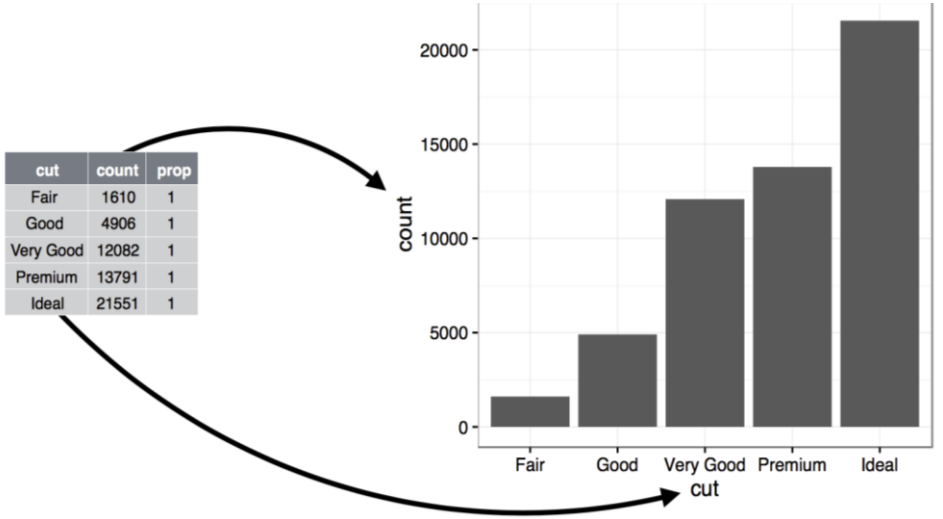
- Data Visualization (Recap)
- **Types of Data:** Numerical and Categorical Data
- **Types of Numerical Data:** Continuous vs Discrete
- **Types of Categorical Data:** Nominal vs Ordinal
- Introduction to Matplotlib and Seaborn
  - Basics of Matplotlib
  - Basics of Seaborn
  - Comparison between the Matplotlib and Seaborn.
- Visualization of Numeric Data
  - Bar chart, line plot, scatter plot, histogram.
- Visualization of Categorical Data
  - Bar plots, count plots, and pie charts.
- Saving and exporting visualizations
- Color theory for data visualization.
  - Importance of accessibility
  - Cultural considerations in color usage.
- **Coding:** Practical coding examples for the topics

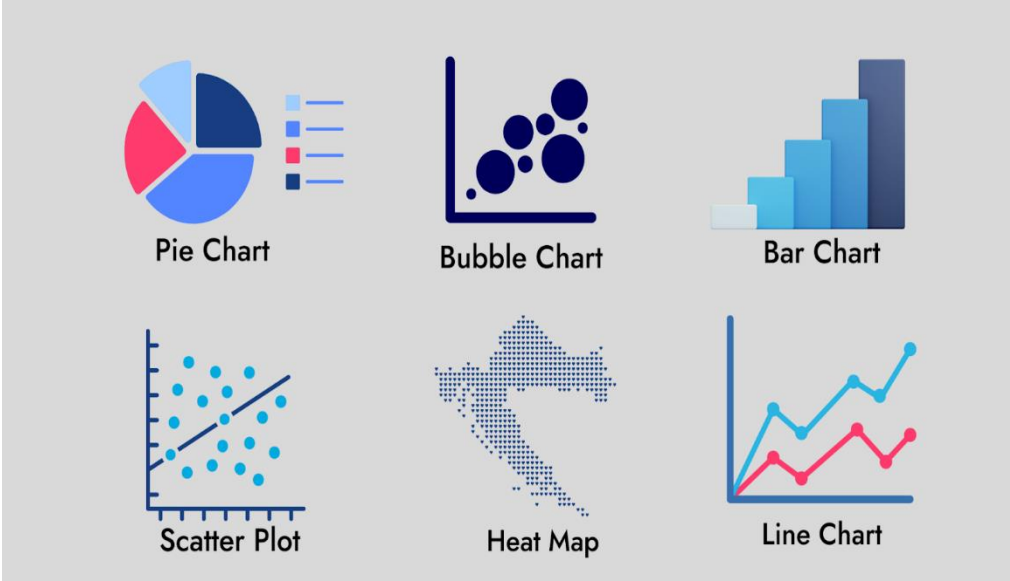
# Data Visualization

- The graphical representation of information and data using visual elements such as charts, graphs, and maps.
- It is a critical skill in data science, helping transform raw data into understandable and actionable insights.

- **Purpose**:
  - Helps in understanding complex data through visual context.
  - Visualizing data allows people to see relationships, patterns, and trends in the information you're trying to communicate.

In this lecture we'll explore techniques for visualizing both numeric and categorical data.



# Types of Data (in terms of measurement)

Understanding data types is crucial for effective data processing, analysis, and visualization.

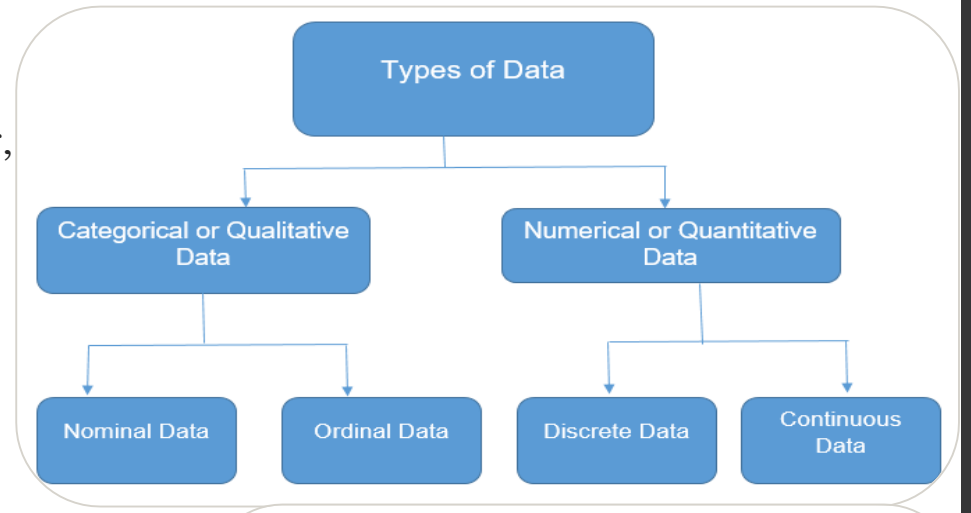
## Two main categories:

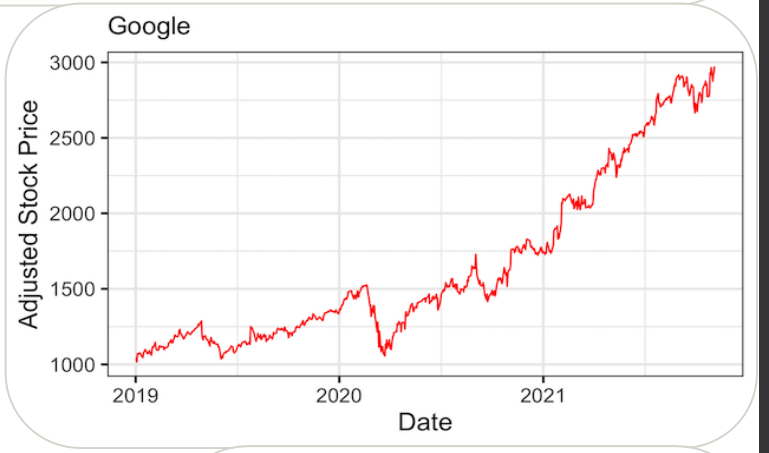
1. **Numerical (Quantitative Data)**
   - Can be measured and expressed in numbers.
   - Suitable for statistical analysis
   - Answers questions like “how much?”, “how many?”, or “how often?”.

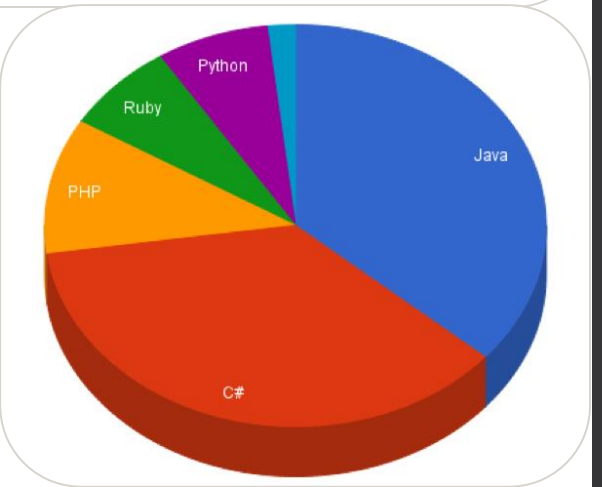
- **Examples**:
  - Stock prices over time
  - Population growth rates
  - Height or weight of a person

2. **Categorical (Qualitative Data)**
   - Represent labels, categories, or classifications
   - Cannot be inherently measured numerically, but can be counted or converted to percentages for analysis.
   - Ideal for visualizing groupings or classifications using charts like pie charts or count plots.
   - Answers questions about categories, preferences, or classifications.

- **Examples:**
  - Product categories in sales data
  - Favorite programming language of developers



# Types of Numerical Data

## a) Discrete Data

- Takes specific, separate values (usually whole numbers)
- Countable and finite.

➢ **Examples**:

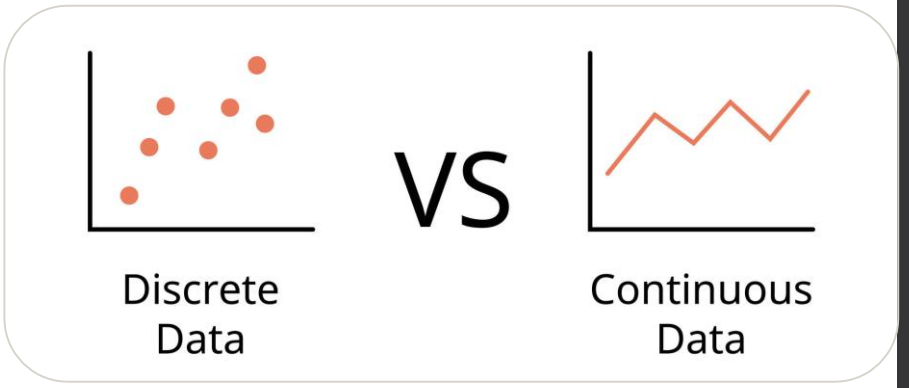
- Number of students in a class
- Number of items sold per day

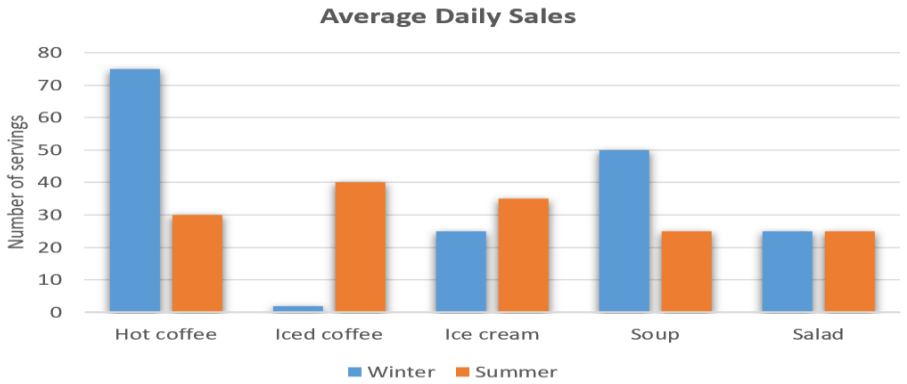
## b) Continuous Data

- Can take any value within a range (including fractions)
- Measurable and divisible

➢ **Examples**:

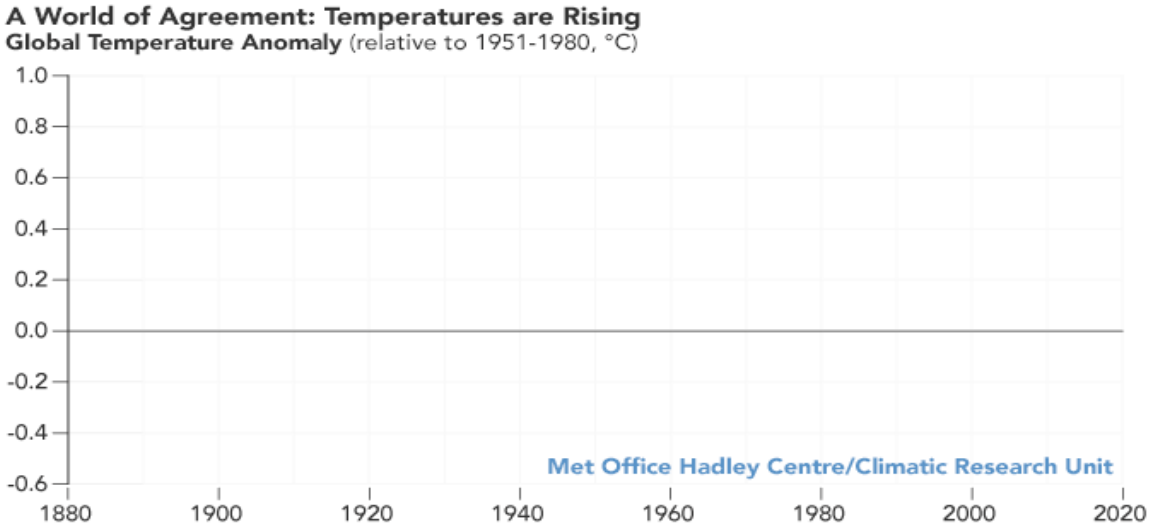
- Time spend on a webpage
- Temperature readings in weather data




## Key Differences

- **Countable vs. Measurable:**
  - Discrete data is countable (e.g., you can count 5 students)
  - Continuous data is measurable (e.g., you measure 98.6°F temperature)
- **Finite vs. Infinite:**
  - Discrete data has a finite number of possible values between two points
  - Continuous data has an infinite number of possible values between two points
- **Divisibility**:
  - Discrete data is not divisible in the context of what's being counted (e.g., 2.5 students doesn't make sense)
  - Continuous data is infinitely divisible (e.g., you can always measure temperature more precisely)




❖The line plot to the right shows yearly temperature anomalies from 1880 to 2020, recorded by NASA, NOAA, Berkeley Earth, the Met Office Hadley Centre (UK), and Cowtan and Way.

# Types of Categorical Data

a) **Nominal Data**

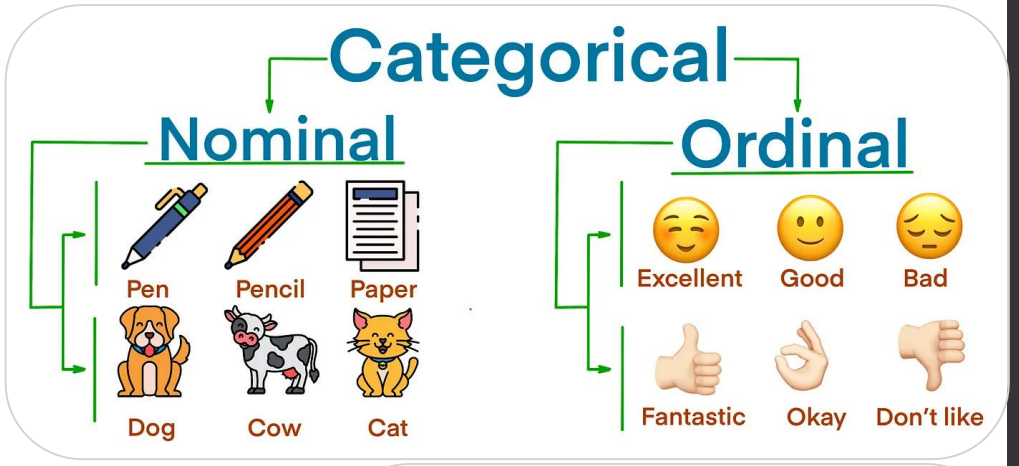
- Categories without any intrinsic order or ranking.
  - **Examples**:
    - City names (New York, London, Islamabad)
    - Vehicle brands (Toyota, Ford, BMW)

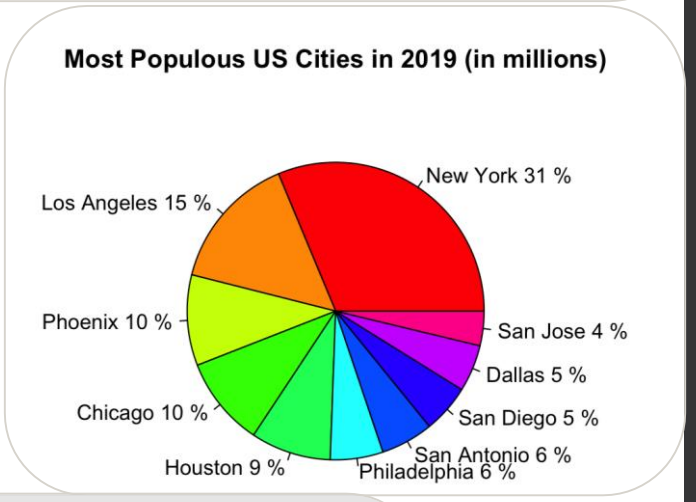
b) **Ordinal Data**

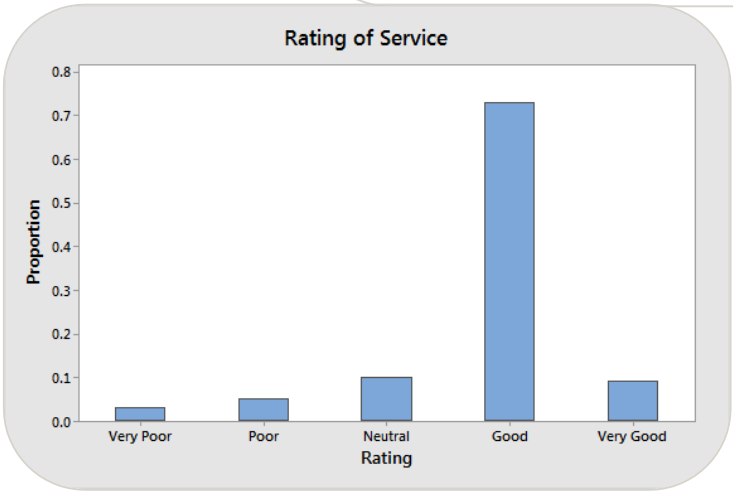
- Categories with a meaningful order or ranking, but uneven or undefined intervals between levels.
  - **Examples:**
    - Education levels (High Schools, Bachelor's, Master's)
    - Customer satisfaction ratings. (Poor, Fair, Good, Excellent)

## Key Differences:

- **Order:**
  - **Nominal:** No natural order (like colors: red, blue, green)
  - **Ordinal:** Has a clear order (like rankings: 1st, 2nd, 3rd)
- **Comparisons:**
  - **Nominal:** Can only say if two items are the same or different
  - **Ordinal:** Can say if one item is greater than, less than, or equal to another
- **Typical visualizations:**
  - **Nominal:** Bar charts, pie charts
  - **Ordinal:** Bar charts, stacked bar charts, sometimes line graphs



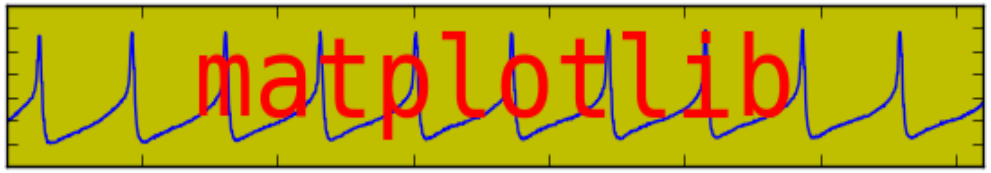



# Introduction to Matplotlib and Seaborn

## Matplotlib:

- **Initial release:** 2003
- First Python data visualization library.
  - Most popular and widely used data visualization library.
- Matplotlib is the grandfather of python visualization packages.
  - Foundation for many other libraries.
  - Popular libraries like Seaborn and Plotly are built on Matplotlib, using it as the core for rendering plots.
- **Highly Customizable:**
  - Low-level, highly customizable plotting library.
  - It offering fine-grained control over plot elements (axes, labels, ticks, colors), enabling a wide range of visualizations from basic bar charts to complex interactive 2D graphs.
- **Key Strengths:**
  - **Powerful but Complex:**
    - While flexible, it can be challenging for beginners compared to higher-level libraries like Seaborn.
  - **Multiple Visualization Modes:**
    - Supports static, animated, and interactive plots for diverse use cases.




Matplotlib's original logo (2003 -- 2008).


Matplotlib's logo (2008 -- 2015).

# Basics of Matplotlib

## Core Components:

- **Figure:** The overall window or page that everything is drawn on.
- **Axes:** A single plot within the figure.

## Common Plot Types:

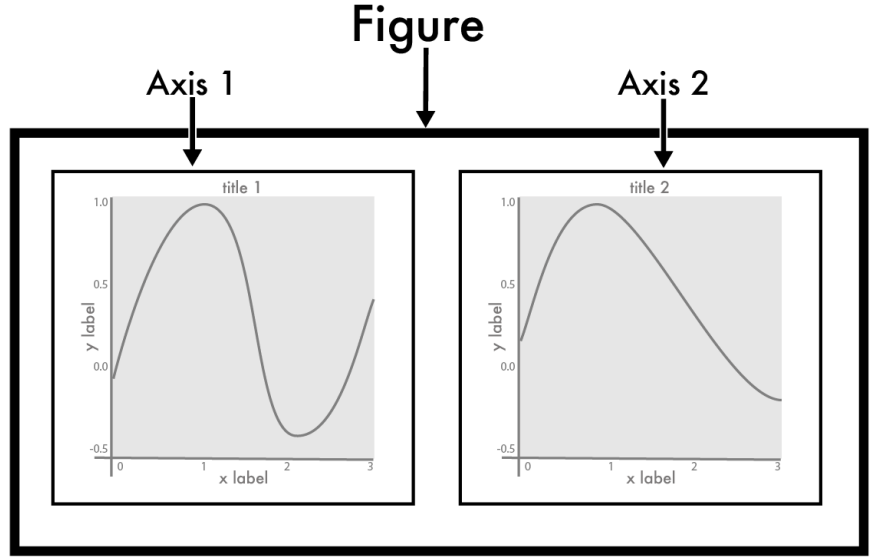
- Line plot, bar chart, scatter plot, histogram.
- **Basic Plotting Functions:**
  - **plot()**: Creates line plots.
  - **scatter()**: For scatter plots.
  - **bar()**: For bar charts.

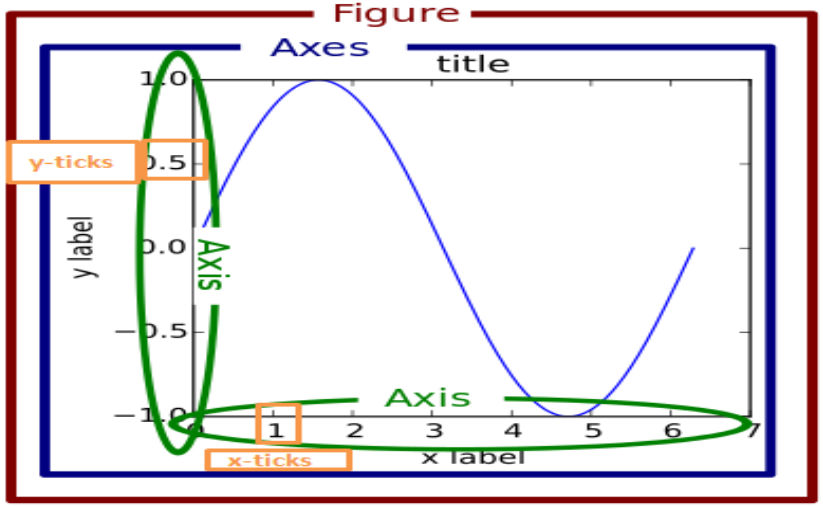
## Customization:

- Change colors, labels, titles, legends and more with ease.

## Example Code:

```
import matplotlib.pyplot as plt
x, y = [1, 2, 3, 4], [1, 4, 9, 16]
plt.plot(x, y)
plt.ylabel('Squared Numbers')
plt.show()
```



# Seaborn:

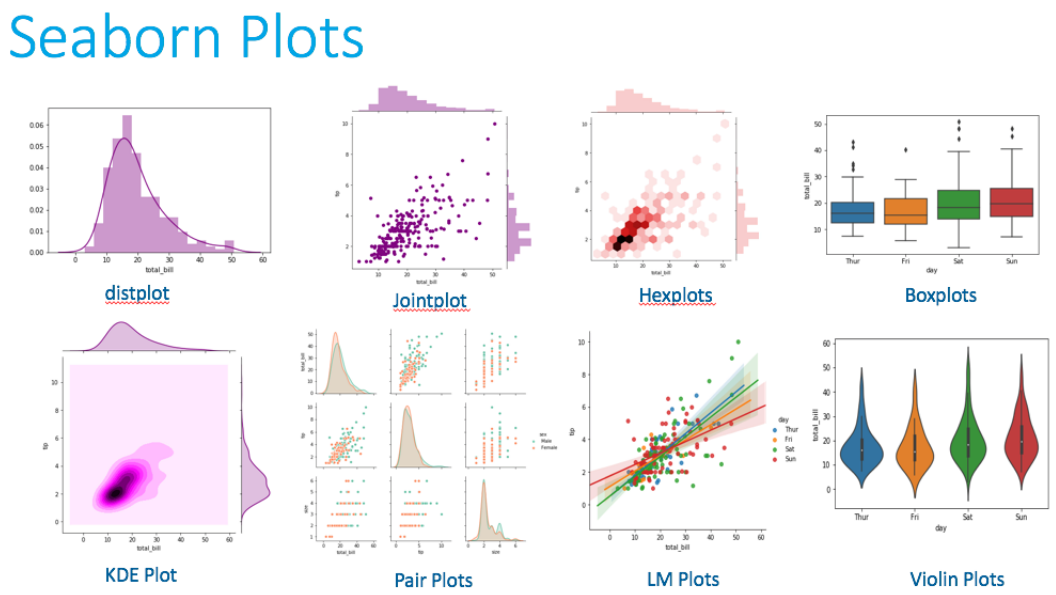
- **Initial Release:** 2014
- It is an advanced data visualization library built on top of Matplotlib, designed to make complex statistical plots easier to create.

## Key Features

- **High-Level Interface**
  - Intuitive API for statistical graphics
  - Minimal effort required
- **Advanced Visualizations**
  - Built-in functions for complex plots
  - **Examples:** heatmaps, pair plots, violin plots, regression plots
- **Data Integration**
  - Seamless integration with Pandas DataFrames
  - Efficient handling of large datasets

## Key Strengths

- **User-Friendly and Less Complex**
  - Simplifies statistical visualizations
  - Maintains flexibility for customization
- **Default Aesthetic Settings**
  - Default style is suitable for publication
  - More refined than basic Matplotlib plots



# Basics of Seaborn

## Core Features:

- Built on top of Matplotlib
- Built-in themes and color palettes for aesthetic visualizations.
- Supports complex statistical plots (e.g., pair plots, heatmaps).

## Integration with Pandas:

- Seamless integration with Pandas DataFrames

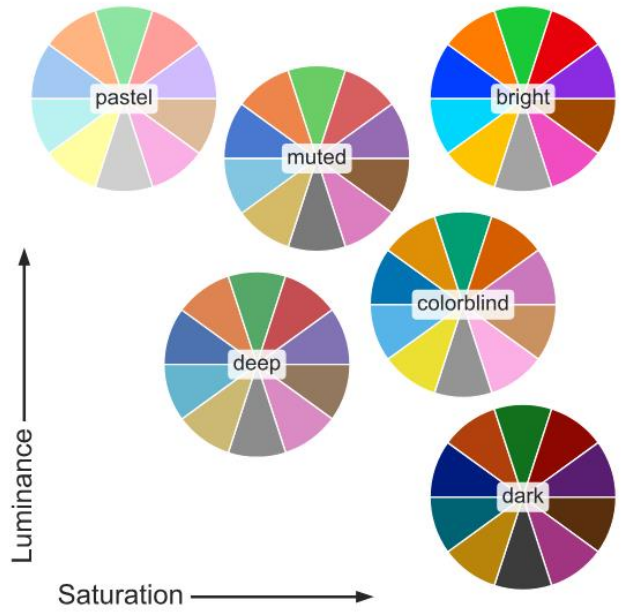
## Example Code:

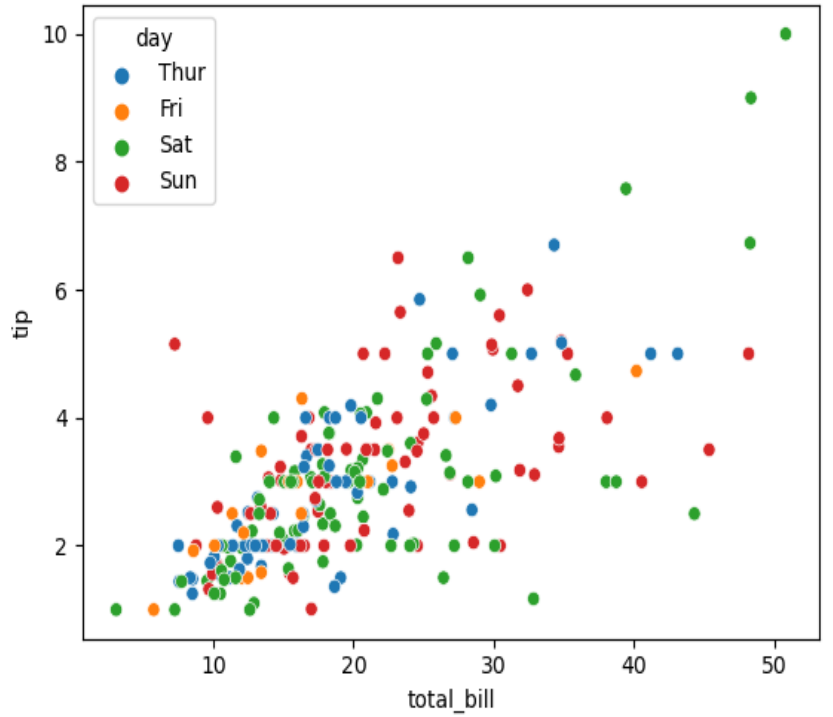
```
import seaborn as sns

tips = sns.load_dataset('tips')

sns.scatterplot(data=tips, x='total_bill', y='tip',
hue='day')
```

- Output: A scatter plot visualizing tips based on the total bill, differentiated by day.



# Comparison between Matplotlib and Seaborn

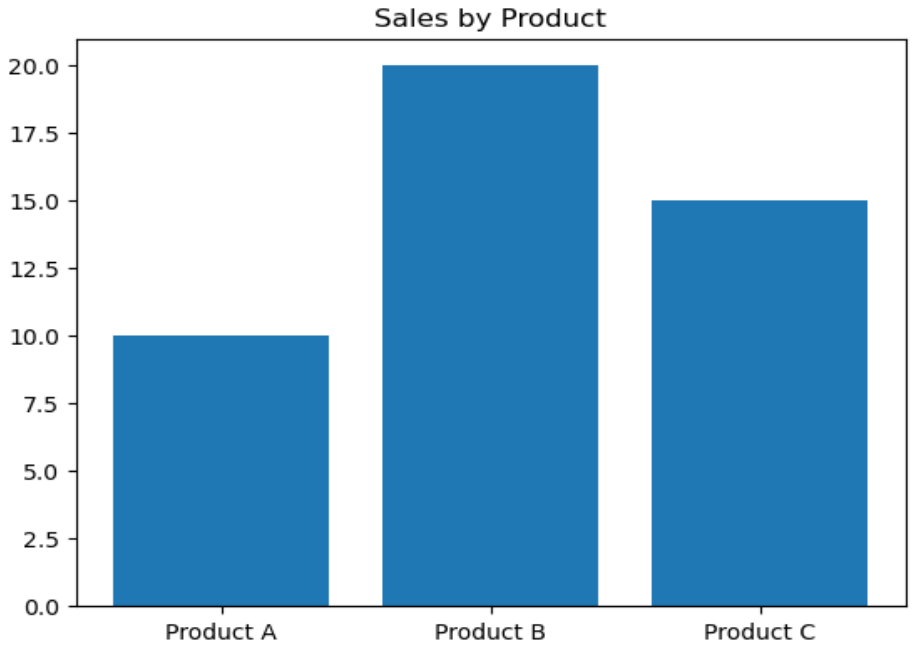
- **Ease of use:**
  - Seaborn generally requires less code
- **Customization:**
  - Matplotlib offers more fine-grained control
- **Statistical functionality:**
  - Seaborn has built-in statistical features
- **When to use each:**
  - Depends on specific needs and complexity of the visualization
  - Use Matplotlib for fine-tuned control and Seaborn for quick, beautiful statistical plots.

# Visualization of Numeric Data

### Bar Chart

- **Usage**: Comparing quantities across categories.
- **Example**: Sales of different products.
- **Code**:

```
plt.bar(['Product A', 'Product B', 'Product C'], [10, 20, 15])
plt.title('Sales by Product')
plt.show()
```




### Line Plot

- **Usage**: Show trends over time
- **Example**: Stock prices over a year
- **Code:**

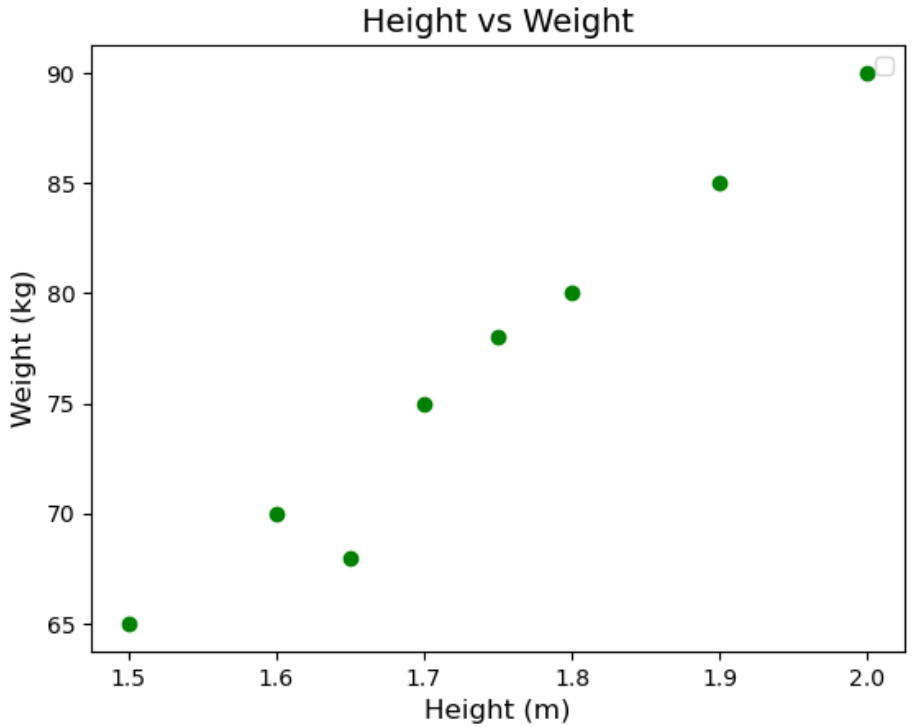
```
x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y, marker='o', linestyle='-', color='b', label='Stock A')
plt.title('Stock Prices Over Time', fontsize=14)
plt.xlabel('Time (Days)', fontsize=12)
plt.ylabel('Stock Price (USD)', fontsize=12)
plt.grid(True)
plt.legend()
plt.show()
```

# Visualization of Numeric Data

## Scatter Plot

- **Usage:** Show the relationship between two variables.
- **Example**: Height vs. weight.
- **Code:**

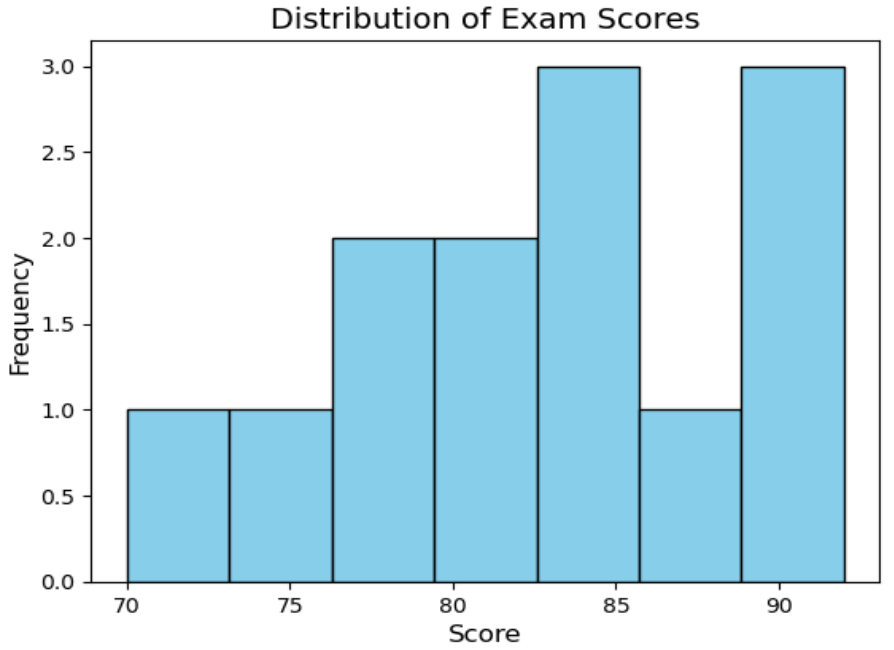
```
heights = [1.5, 1.8, 1.6, 2.0, 1.7, 1.9, 1.75, 1.65]
weights = [65, 80, 70, 90, 75, 85, 78, 68]
plt.scatter(heights, weights, color="g", marker='o')
plt.title('Height vs Weight', fontsize=14)
plt.xlabel('Height (m)', fontsize=12)
plt.ylabel('Weight (kg)', fontsize=12)
plt.show()
```




## Histogram

- **Usage:** Display the frequency distribution of a variable.
- **Example:** Distribution of exam scores.
- **Code:**

```
scores = [70, 85, 90, 75, 80, 90, 85, 80, 88, 92, 77, 84, 79]
plt.hist(scores, bins = 7, color='skyblue', edgecolor='black')
plt.title('Distribution of Exam Scores', fontsize=14)
plt.xlabel('Score', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.show()
```



# Visualization of Categorical Data

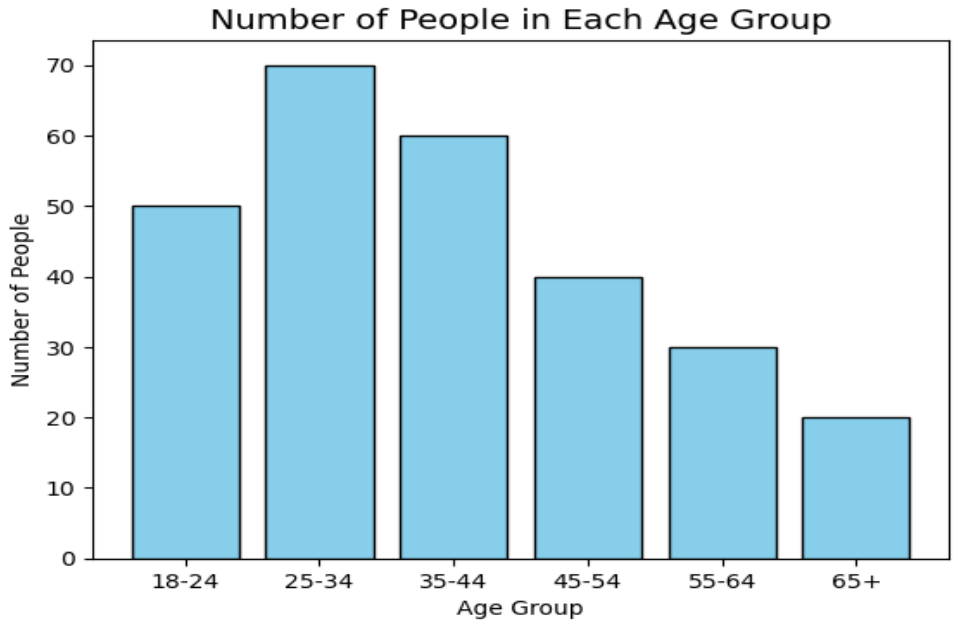
## Bar Plots

- **Usage:** Shows frequency of categories with bars.
- **Example:** Number of people in each age group.
- **Code:**

```
age_groups = ['18-24', '25-34', '35-44', '45-54', '55-64', '65+']
number_of_people = [50, 70, 60, 40, 30, 20]

plt.bar(age_groups, number_of_people, color='skyblue',
edgecolor='black')

plt.title('Number of People in Each Age Group', fontsize=14)
plt.xlabel('Age Group')
plt.ylabel('Number of People')
plt.show()
```



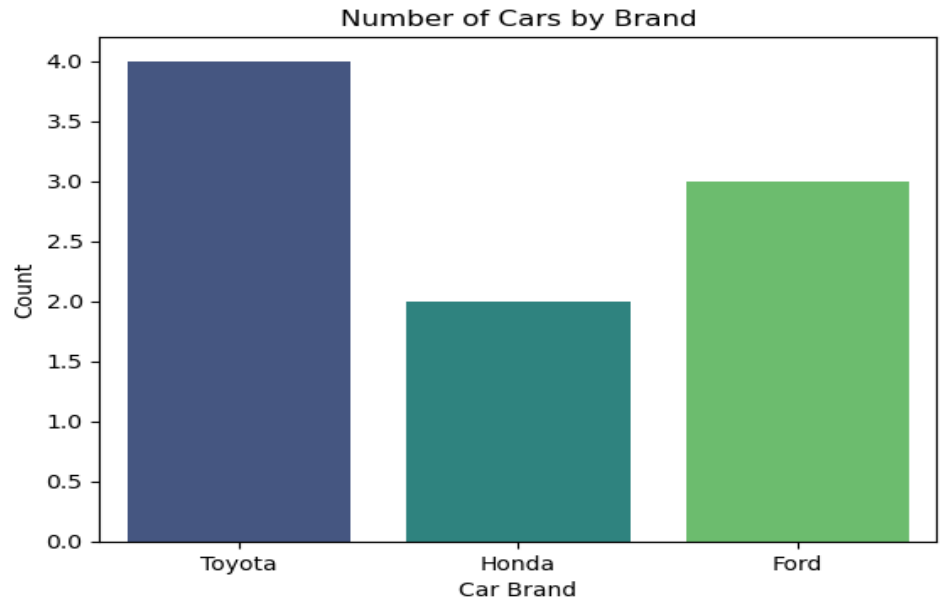

## Count Plots

- **Usage:** Count occurrences of categorical variables.
- **Example:** Number of cars by brand in a dataset.
- **Code:**

```
data = {'Brand': ['Toyota', 'Honda', 'Ford', 'Toyota', 'Ford', 'Honda', 'Toyota', 'Ford',
'Toyota']}
df = pd.DataFrame(data)

sns.countplot(x='Brand', data=df, palette='viridis')

plt.title('Number of Cars by Brand')
plt.xlabel('Car Brand')
plt.ylabel('Count')
plt.show()
```



# Visualization of Categorical Data

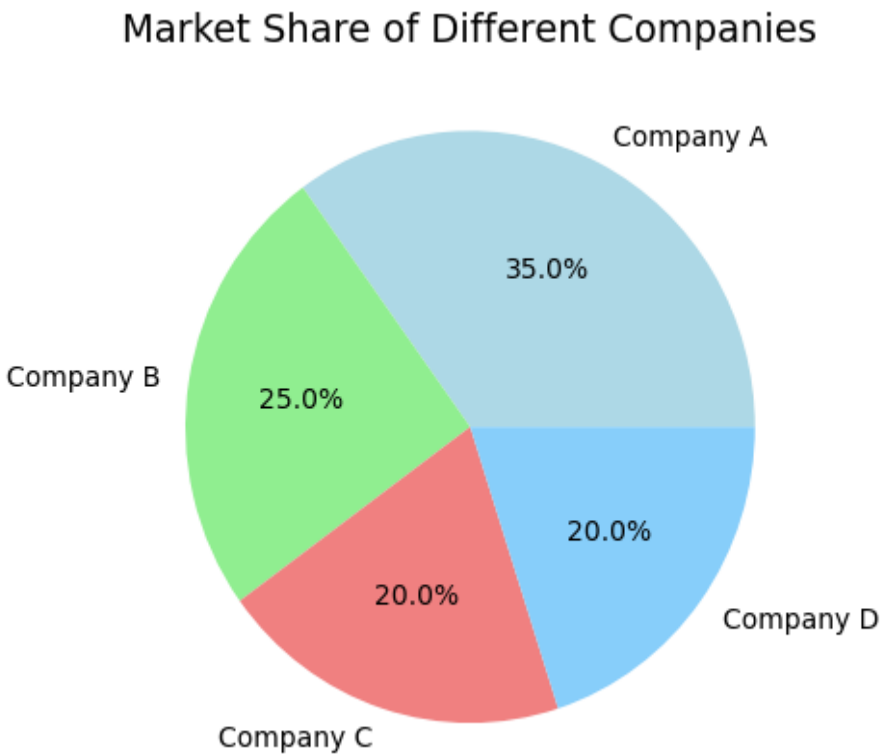
### Pie Charts

- **Usage**: Show proportions within a  whole.
- Example: Market share of different companies.
- **Note**: Use sparingly, as they can be hard to interpret accurately
- **Code**:

```
companies = ['Company A', 'Company B', 'Company C', 'Company D']
market_share = [35, 25, 20, 20]

plt.pie(market_share, labels=companies, autopct='%1.1f%%',
colors=['lightblue', 'lightgreen', 'lightcoral', 'lightskyblue'])

plt.title('Market Share of Different Companies', fontsize=14)
plt.show()
```



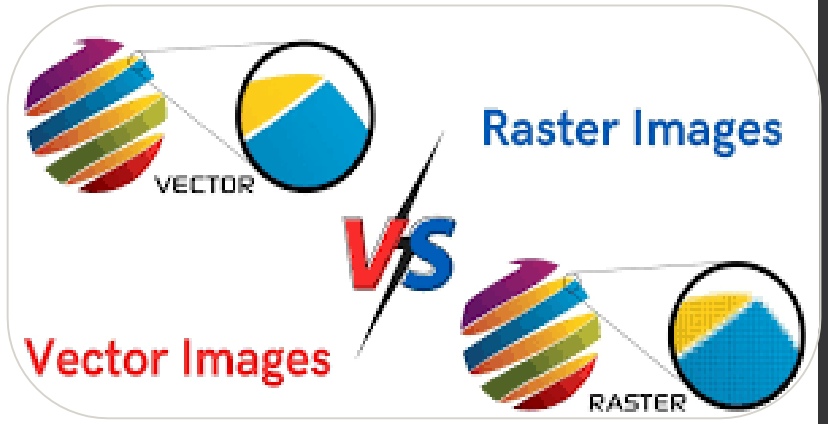
# Saving and Exporting Visualizations

## Why Export?

- Preserve work for later use
- Share with others (reports, presentations, publications)
- Use in different applications or platforms

## Common File Formats:

1. **Raster Graphics**: PNG, JPEG, TIFF
   - It made up of tiny pixels, are ideal for detailed and colorful images like photographs.
   - Fixed resolution (quality may decrease when enlarged)
   - **PNG**: Lossless compression (no loss of image quality when saved), best for web graphics and images with text or sharp edges.
   - **JPEG**: Lossy compression (slight reduction in image quality to achieve smaller file sizes), best for photographs, complex images with many colors.
2. **Vector Graphics**: SVG, PDF, EPS
   - Use mathematical equations to define shapes, lines, and curves
   - Scalable without quality loss (can be resized infinitely)
   - Ideal for: Logos, illustrations, charts, and graphs
   - Best for print and high-resolution displays





## Best Practices:

- Choose appropriate format based on intended use
- Consider resolution requirements (e.g., print vs. web)
- Maintain aspect ratio when resizing
- Test exported files in target environment

**Matplotlib**: savefig() function

**Seaborn:** plt.savefig() (uses Matplotlib backend).

## Example Code:

```
plt.savefig('my_plot.png', dpi=300)
```

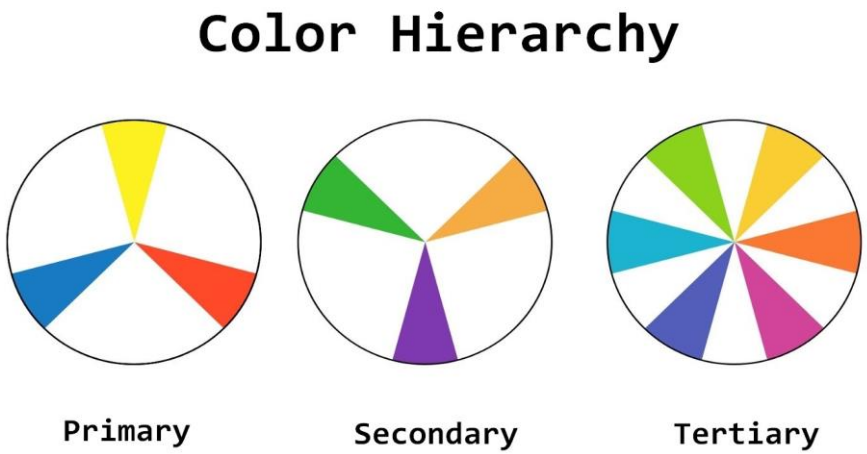
```
plt.savefig('plot.svg', format='svg')
```

# Color Theory for Data Visualization

## Understanding Color Theory Basics

### ➢ Primary, Secondary, and Tertiary Colors

1. **Primary Colors**: The base colors (RYB model: Red, Yellow, Blue).
2. **Secondary Colors**: Made by mixing primary colors (Red + Yellow = Orange, etc.).
3. **Tertiary Colors**: Made by mixing primary and secondary colors (Red + Orange = Red-Orange).




### ➢ Color Wheels

- A visual representation showing the relationships between primary, secondary, and tertiary colors.

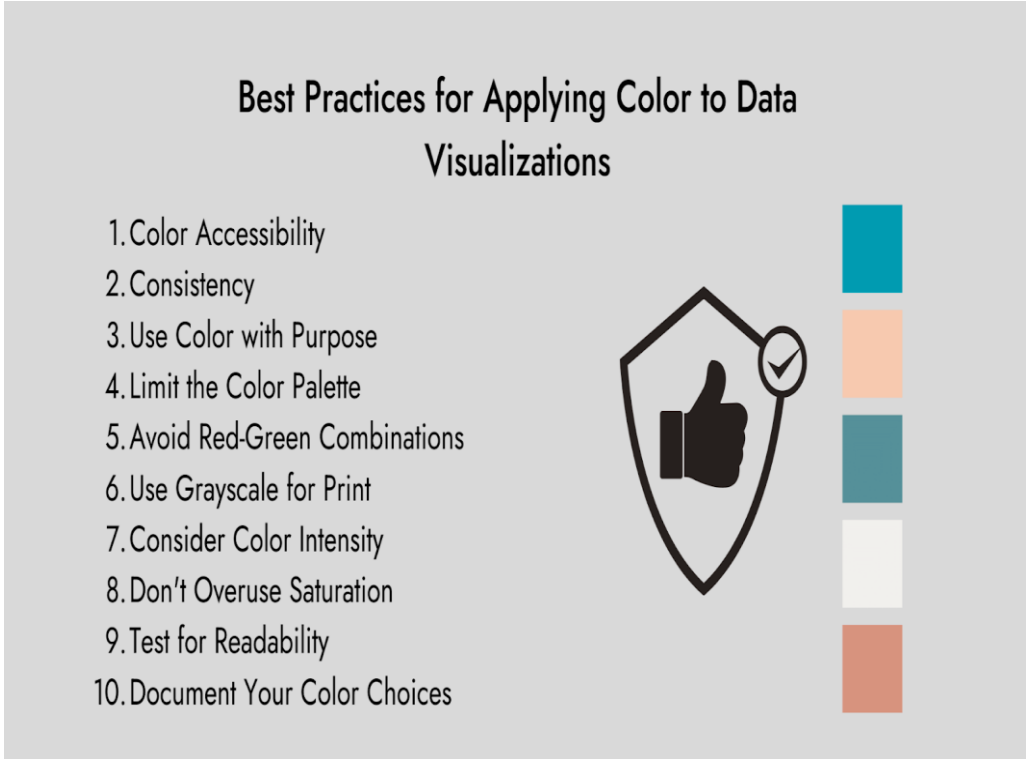
### ➢ Color Harmony

➢ Color harmony is a basic color theory technique for combining colors.

- **Complementary Colors**: Opposite on the color wheel (e.g., Red and Green).
- **Analogous Colors**: Next to each other on the color wheel (e.g., Blue, Blue-Green, Green).
- **Triadic Colors**: Three evenly spaced colors on the color wheel (e.g., Red, Yellow, Blue)


# Best Practices for Applying Color to Data Visualizations

1. **Color Accessibility:**
   - Consider color blindness.
   - Ensure sufficient contrast for easy readability.
   - Test visualizations using color-blind simulation tools to ensure inclusivity.
2. **Consistency:**
   - Maintain consistency in color usage across different visualizations and data representations.
   - Assign specific colors to specific categories.
3. **Use Color with Purpose:**
   - Assign meaningful colors to data points to convey information, trends, or categories.
   - Consider cultural and contextual associations with colors to ensure accurate interpretation.
4. **Limit the Color Palette:**
   - Keep the number of colors in your palette to a minimum to avoid visual clutter.
5. **Color Combinations**
   - Avoid Red-Green Combinations, as they can be challenging for individuals with color blindness.
6. **Color Properties**
   - Use intensity for data values/hierarchy
   - Avoid oversaturation
   - Ensure text readability
7. **Documentation:**
   - Provide legends explaining color meanings.



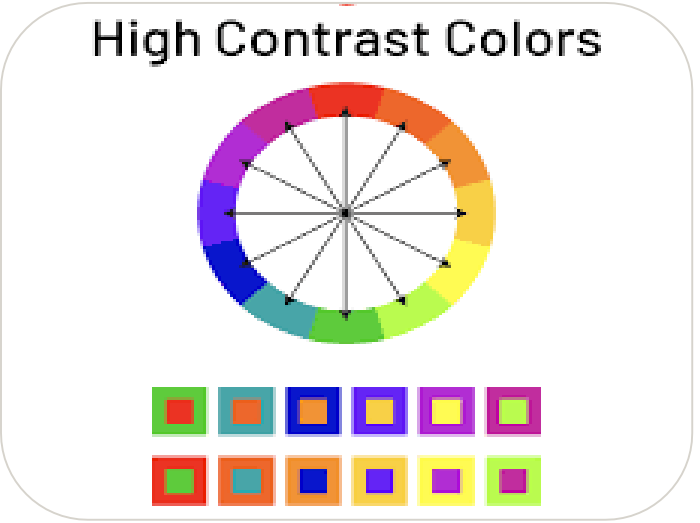
# Importance of Accessibility

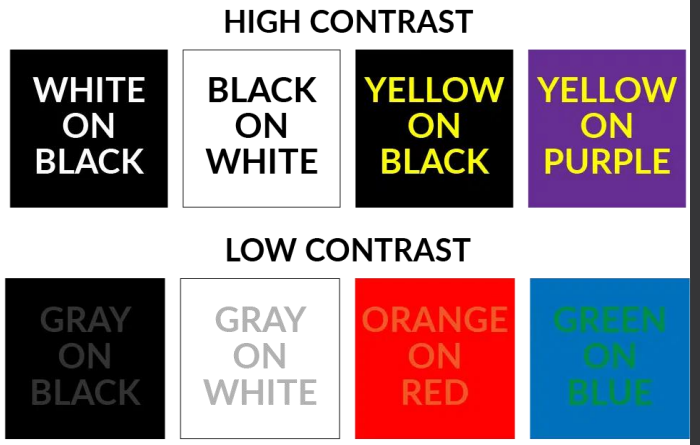
## Key Considerations:

- Use high contrast colors to ensure readability.
  - **High Contrast:** Colors that are directly opposite one another on the color wheel have the highest contrast possible, while colors next to one another have a low contrast.
- Test your visualizations for colorblindness.
  - Avoid red-green combinations
  - Use color and shape to differentiate
- Provide alternative test descriptions for visual content.

## Tools:

- Use tool like the Coblis Color Blindness Simulator to test accessibility.




# Cultural Considerations in Color Usage

- Colors have different meanings across cultures.
  - Perceptions vary by region, and a single color can convey different, even contrasting meanings worldwide.
- **Examples:**
  - **Red**
    - In western and middle east countries red evokes danger, caution and urgency.
    - In India, red is associated with purity, and brides traditionally wear red at weddings.
    - In china, red symbolizes luck and happiness.
  - **White**
    - In Western societies, white is associated with weddings and purity.
    - In many Asian cultures, white represents death, mourning, and humility.
  - **Green**
    - In Western countries, green represents the environment, progress, and luck, symbolizing safety and growth.
    - In Mexico, it stands for independence and patriotism, while in South America, it represents death.
    - Green is strongly associated with Islam and appears in several national flags as a religious symbol.
- **Best practice:**
  - Research your audience and choose colors that align with their cultural context.

# Summary and Next Steps


- **Key Takeaways:**
  - Understanding the types of data: Numerical and Categorical
    - Numerical: Discrete and Continuous
    - Categorical: Nominal and Ordinal
  - Visualization Tools:
    - Matplotlib: Foundation for Python plotting.
    - Seaborn: Statistical data visualization
    - Comparison between Matplotlib and Seaborn
  - Visualization Techniques:
    - For Numeric Data: Bar chart, line plot, scatter plot, histogram
    - For Categorical Data: Bar plots, count plot, pie chart
  - Best Practices
    - Saving and exporting visualizations
    - Applying Color Theory in Data Visualization
    - Ensuring accessibility in your visualizations

- **Next Lecture:** Advanced Plotting with Matplotlib & Seaborn.

# Thank You