**Matplotlib**

**What is Matplotlib?**

Matplotlib is a widely used Python library for creating high-quality visualizations, such as plots, charts, graphs, and figures. It allows you to generate static, animated, or interactive graphics in a variety of formats (e.g., PNG, PDF, SVG). Matplotlib is highly customizable and integrates well with other libraries like NumPy and Pandas, making it ideal for data analysis, scientific computing, and exploratory data visualization. It was created by John D. Hunter in 2003 and is inspired by MATLAB's plotting capabilities, hence the name "matplotlib."

Key features include:

* Support for a wide range of plot types (line, bar, scatter, histogram, pie, etc.).
* Customization of axes, labels, titles, colors, and styles.
* Ability to create subplots and complex layouts.
* Exporting figures to files or displaying them interactively (e.g., in Jupyter notebooks).
*  Line charts 📈
*  Bar charts 📊
*  Pie charts 🥧
*  Histograms 📉
*  Scatter plots ⚪

**Advantages**

**1. Highly Customizable**

* Complete control over every aspect of the plot
* Fine-grained customization of colors, fonts, styles, etc.

**2. Publication Quality**

* Produces high-quality figures suitable for academic papers
* Supports multiple output formats (PDF, SVG, PNG, etc.)

**3. Wide Range of Plot Types**

* Supports numerous chart types (line, scatter, bar, histogram, etc.)
* 2D and 3D plotting capabilities

**4. MATLAB-like Syntax**

* Familiar interface for MATLAB users
* Easy learning curve for beginners

**5. Integration with Other Libraries**

* Works well with NumPy, Pandas, and SciPy
* Foundation for higher-level libraries like Seaborn

**6. Cross-Platform Compatibility**

* Works on Windows, macOS, and Linux
* Consistent output across platforms

**Disadvantages**

**1. Verbose Syntax**

**2. Steep Learning Curve for Advanced Features**

* Complex for intricate visualizations
* Requires understanding of object-oriented interface

**3. Default Aesthetics**

* Default styling is less modern compared to newer libraries
* Requires additional customization for polished looks

**4. Performance with Large Datasets**

* Can be slow with very large datasets
* Not optimized for real-time data streaming

**5. Limited Interactive Features**

* Basic interactivity compared to Plotly or Bokeh
* Requires additional libraries for advanced interactivity
* **COMPREHENSIVE MATPLOTLIB FUNCTION GUIDE**
* 1. BASIC PLOTTING FUNCTIONS:
* plt.plot(x, y) - Line plot
* plt.scatter(x, y) - Scatter plot
* plt.bar(x, y) - Vertical bar chart
* plt.barh(x, y) - Horizontal bar chart
* plt.pie(sizes) - Pie chart
* plt.hist(data) - Histogram
* plt.boxplot(data) - Box plot
* 2. FIGURE AND SUBPLOT FUNCTIONS
* plt.figure(figsize=(w,h)) - Create new figure
* plt.subplot(rows, cols, index) - Create subplots
* plt.subplots() - Create figure and subplots
* plt.tight\_layout() - Adjust spacing
* 3. CUSTOMIZATION FUNCTIONS:
* plt.title('Title') - Add title
* plt.xlabel('X Label') - X-axis label
* plt.ylabel('Y Label') - Y-axis label
* plt.xlim(min, max) - Set x-axis limits
* plt.ylim(min, max) - Set y-axis limits
* plt.xticks(positions, labels) - Custom x-ticks
* plt.yticks(positions, labels) - Custom y-ticks
* plt.grid(True) - Add grid
* plt.legend() - Show legend
* plt.colorbar() - Add color bar
* 4. LINE AND MARKER PROPERTIES
* color='red' or color='#FF0000' - Set color
* linewidth=2 or lw=2 - Line thickness
* linestyle='--' or ls='--' - Line style
* marker='o' - Marker style
* markersize=8 or ms=8 - Marker size
* markeredgecolor='black' - Marker edge color
* alpha=0.5 - Transparency
* 5. BAR CHART PROPERTIES:
* width=0.8 - Bar width
* align='center' - Bar alignment
* bottom=array - For stacked bars
* edgecolor='black' - Bar edge color
* hatch='/' - Pattern filling
* 6. PIE CHART PROPERTIES:
* labels=list - Slice labels
* autopct='%1.1f%%' - Percentage format
* startangle=90 - Starting angle
* explode=tuple - Explode slices
* shadow=True - Add shadows
* wedgeprops=dict() - Wedge properties
* 7. HISTOGRAM PROPERTIES:
* bins=10 - Number of bins
* range=(min, max) - Data range
* density=True - Normalize to PDF
* cumulative=True - Cumulative histogram
* histtype='step' - Histogram type
* 8. SCATTER PLOT PROPERTIES:
* s=50 - Marker size
* c=array - Color mapping
* cmap='viridis' - Color map
* marker='s' - Marker shape
* edgecolors='black' - Marker edges
* 9. SAVING AND SHOWING:
* plt.savefig('plot.png') - Save figure
* plt.savefig('plot.pdf') - Save as PDF
* plt.savefig('plot.jpg', dpi=300) - High resolution
* plt.show() - Display plot
* plt.close() - Close figure

**PANDAS FUNCTIONS FOR MATPLOTLIB**

* *# 1. DATA LOADING AND PREPARATION FUNCTIONS*
* pd.read\_csv() - Load data from CSV files
* pd.DataFrame() - Create DataFrame from dict/list
* df.head() / df.tail() - View first/last rows
* df.info() - DataFrame information
* df.describe() - Statistical summary
* *# 2. DATA SELECTION AND FILTERING FUNCTIONS*
* df['column'] - Select single column
* df[['col1', 'col2']] - Select multiple columns
* df.loc[] - Label-based selection
* df.iloc[] - Integer-based selection
* df.query() - Query with conditions
* *# 3. DATA AGGREGATION AND GROUPING FUNCTIONS*
* df.groupby() - Group data by categories
* df.agg() - Multiple aggregations
* df.value\_counts() - Count unique values
* df.pivot\_table() - Create pivot tables
* df.corr() - Correlation matrix
* *# 4. DATA CLEANING AND MANIPULATION FUNCTIONS*
* df.isnull() - Detect missing values
* df.fillna() - Fill missing values
* df.dropna() - Drop missing values
* df.astype() - Change data types
* df.sort\_values() - Sort DataFrame
* *# 5. STATISTICAL FUNCTIONS*
* df.mean() - Mean of columns
* df.median() - Median of columns
* df.std() - Standard deviation
* df.count() - Count non-null values
* df.max() / df.min() - Maximum/Minimum values

**2. NUMPY FUNCTIONS FOR MATPLOTLIB**

* *# 1. ARRAY CREATION FUNCTIONS*
* np.array() - Create array from list
* np.arange() - Create array with range
* np.linspace() - Evenly spaced numbers
* np.zeros() / np.ones() - Arrays of zeros/ones
* np.random - Random number generation
* *# 2. MATHEMATICAL OPERATION FUNCTIONS*
* np.add() / np.subtract() - Addition/Subtraction
* np.multiply() / np.divide() - Multiplication/Division
* np.sqrt() - Square root
* np.exp() / np.log() - Exponential/Logarithm
* np.sin() / np.cos() / np.tan() - Trigonometric functions
* *# 3. STATISTICAL FUNCTIONS*
* np.mean() - Mean/average
* np.median() - Median
* np.std() - Standard deviation
* np.var() - Variance
* np.percentile() - Percentiles
* np.corrcoef() - Correlation coefficient
* *# 4. ARRAY MANIPULATION FUNCTIONS*
* np.reshape() - Change array shape
* np.concatenate() - Join arrays
* np.split() - Split array
* np.sort() - Sort array
* np.unique() - Find unique elements
* *# 5. RANDOM NUMBER GENERATION FUNCTIONS*
* np.random.rand() - Uniform distribution [0,1)
* np.random.randn() - Standard normal distribution
* np.random.randint() - Random integers
* np.random.normal() - Normal distribution
* np.random.choice() - Random choice from array

**Key Takeaways:**

**🐼 Pandas is best for:**

* Data loading and cleaning
* Data aggregation and grouping
* Handling missing values
* Working with labeled data
* Database-like operations

**🔢 NumPy is best for:**

* Mathematical operations
* Array manipulations
* Random number generation
* Statistical calculations
* Numerical computations

**📊 Matplotlib integrates with both:**

* Pandas DataFrames can be plotted directly
* NumPy arrays are the fundamental data structure
* Combine Pandas for data manipulation + NumPy for calculations + Matplotlib for visualization

**SEABORN**

**What is Seaborn?**

Seaborn is a Python data visualization library built on top of Matplotlib that provides a high-level interface for creating attractive statistical graphics. It integrates closely with pandas DataFrames and provides beautiful default styles and color palettes to make statistical plots more visually appealing.

**Key Functions in Seaborn**

**1. Plotting Functions**

* **sns.scatterplot()**- Scatter plots
* **sns.lineplot() -** Line plots
* **sns.barplot() -** Bar plots
* **sns.histplot() -** Histograms
* **sns.boxplot() -** Box plots
* **sns.violinplot(**) - Violin plots
* **sns.heatmap(**) - Heatmaps
* **sns.pairplot()**- Pair plots
* **sns.jointplot()** - Joint distributions
* **sns.lmplot() -** Regression plots

**2. Styling Functions**

* **sns.set\_style() -** Set plot style
* **sns.set\_palette() -** Set color palette
* **sns.color\_palette() -** Color palettes

**3. Statistical Functions**

* **sns.regplot() -** Regression plots
* **sns.distplot() -** Distribution plots (deprecated, use histplot)
* **sns.kdeplot() -** Kernel Density Estimation

**Complete List of Seaborn Functions**

**1. Relational Plots (Relationships between variables)**

* **sns.relplot() - F**igure-level interface for relational plots
* **sns.scatterplo**t() - Scatter plot
* **sns.lineplot() -** Line plot

**2. Categorical Plots**

* **sns.catplot() -** Figure-level interface for categorical plots
* **sns.boxplot()**- Box plot
* **sns.violinplot()**- Violin plot
* **sns.boxenplot()** - Enhanced box plot
* **sns.stripplot() -** Strip plot
* **sns.swarmplot(**) - Swarm plot
* **sns.barplot() -** Bar plot
* **sns.countplot()** - Count plot
* **sns.pointplot()**- Point plot

**3. Distribution Plots**

* **sns.displot() -** Figure-level interface for distribution plots
* **sns.histplot() -** Histogram
* **sns.kdeplot() -** Kernel Density Estimate plot
* **sns.ecdfplot()** - Empirical Cumulative Distribution Function
* **sns.rugplot() -** Rug plot

**4. Regression Plots**

* **sns.regplot() -** Regression plot
* **sns.lmplot() -** Figure-level regression plot
* **sns.residplot(**) - Residual plot

**5. Matrix Plots**

* **sns.heatmap() -** Heatmap
* **sns.clustermap(**) - Clustered heatmap

**6. Multi-plot Grids**

* **sns.FacetGrid(**) - Multi-plot grid for plotting conditional relationships
* **sns.PairGrid() -** Grid for pairwise relationships
* **sns.JointGrid()**- Grid for joint and marginal distributions
* **sns.pairplot() -** Pairwise relationships
* **sns.jointplot() -** Joint distribution plot

**7. Styling and Themes**

* **sns.set\_style() -** Set the aesthetic style
* **sns.axes\_style()** - Get the parameters of the aesthetic style
* **sns.set\_palette**() - Set the color palette
* **sns.color\_palette()** - Get or set the color palette
* **sns.set\_context() -** Set the plotting context parameters
* **sns.plotting\_context(**) - Get the plotting context parameters
* **sns.set() -** Set aesthetic parameters in one step
* **sns.reset\_defaults() -** Reset all parameters to default
* **sns.reset\_orig() -** Reset to original matplotlib parameters

**8. Color Palettes**

* **sns.husl\_palette() -** HUSL color palette
* **sns.hls\_palette() - H**LS color palette
* **sns.cubehelix\_palette()** - Cubehelix color palette
* **sns.dark\_palette() -** Dark color palette
* **sns.light\_palette() -** Light color palette
* **sns.diverging\_pa**lette() - Diverging color palette
* **sns.mpl\_palette() -** matplotlib color palette
* **sns.blend\_palette()** - Blend colors to create a palette
* **sns.xkcd\_palette() - X**KCD color palette
* **sns.crayon\_palette()** - Crayon color palette

**9. Utility Functions**

* **sns.axisgrid.Grid -** Base class for grid-like plots
* **sns.utils.ci\_to\_size() -** Convert CI to sample size
* **sns.utils.load\_dataset() -** Load example datasets
* **sns.despine() -** Remove the spines of the plot

**Differences between Seaborn and Matplotlib**

| **Feature** | **Matplotlib** | **Seaborn** |
| --- | --- | --- |
| **Level** | Low-level | High-level |
| **Syntax** | More verbose | Concise |
| **Default Styles** | Basic | Attractive defaults |
| **Pandas Integration** | Manual | Seamless |
| **Statistical Plots** | Limited | Extensive |
| **Color Palettes** | Basic | Advanced |
| **Learning Curve** | Steeper | Easier |

**Key Advantages of Each Library**

**Seaborn Advantages:**

* Concise syntax for complex plots
* Beautiful default styles
* Built-in statistical functions
* Better color palettes
* Easier to create complex plots

**Matplotlib Advantages:**

* More control over plot elements
* Wider range of plot types
* Better for custom, non-standard plots
* Foundation for many other plotting libraries

**When to Use Which?**

* **Use Seaborn** for: Statistical plots, quick exploratory data analysis, when you want attractive defaults
* **Use Matplotlib** for: Highly customized plots, publication-quality figures, when you need fine-grained control