

# # [ Data Visualization and Interpretation ] {CheatSheet}

## 1. Basic Plotting with Matplotlib

- **Line Plot:** `plt.plot(x, y)`
- **Scatter Plot:** `plt.scatter(x, y)`
- **Bar Chart:** `plt.bar(x, height)`
- **Histogram:** `plt.hist(data)`
- **Pie Chart:** `plt.pie(sizes, labels=labels)`
- **Boxplot:** `plt.boxplot(data)`
- **Subplots:** `fig, ax = plt.subplots()`
- **Error Bars:** `plt.errorbar(x, y, yerr=error)`

## 2. Plot Customization

- **Title and Labels:** `plt.title('Title'); plt.xlabel('X Label'); plt.ylabel('Y Label')`
- **Legend:** `plt.legend(['label1', 'label2'])`
- **Axis Limits:** `plt.xlim([xmin, xmax]); plt.ylim([ymin, ymax])`
- **Color and Style:** `plt.plot(x, y, color='blue', linestyle='--', marker='o')`
- **Custom Ticks:** `plt.xticks([ticks], [labels]); plt.yticks([ticks], [labels])`
- **Gridlines:** `plt.grid(True)`
- **Text Annotations:** `plt.text(x, y, 'annotation')`

## 3. Plotting with Seaborn

- **Density Plot:** `sns.kdeplot(data)`
- **Pair Plot:** `sns.pairplot(df)`
- **Heatmap:** `sns.heatmap(data)`
- **Violin Plot:** `sns.violinplot(x='x', y='y', data=df)`
- **Facet Grid:** `g = sns.FacetGrid(df, col='col'); g.map(plt.hist, 'target')`
- **Count Plot:** `sns.countplot(x='x', data=df)`
- **Joint Plot:** `sns.jointplot(x='x', y='y', data=df)`

## 4. Time Series Visualization

- **Time Series Line Plot:** `plt.plot(date, values)`
- **Seasonal Decompose Plot:** `from statsmodels.tsa.seasonal import seasonal_decompose; decomposition = seasonal_decompose(ts); decomposition.plot()`
- **Autocorrelation Plot:** `pd.plotting.autocorrelation_plot(series)`

## 5. Geospatial Data Visualization

- **Basic Map with Folium:** `import folium; m = folium.Map(location=[lat, long]); m`
- **Choropleth Maps:** `folium.Choropleth(geo_data=geo_json, data=df).add_to(m)`
- **Geopandas Basic Plot:** `gdf.plot(column='column')`

## 6. Advanced Plotting Techniques

- **3D Plotting with Matplotlib:** `from mpl_toolkits.mplot3d import Axes3D; fig = plt.figure(); ax = fig.add_subplot(111, projection='3d')`
- **Parallel Coordinates:** `pd.plotting.parallel_coordinates(df, 'class_column')`
- **Radar/Spider Chart:** `from math import pi; N = len(categories); angles = [n / float(N) * 2 * pi for n in range(N)]; ax = plt.subplot(111, polar=True)`

## 7. Plotting with Pandas

- **DataFrame Line Plot:** `df.plot()`
- **DataFrame Bar Plot:** `df.plot.bar()`
- **DataFrame Histogram:** `df.hist()`
- **Area Plot:** `df.plot.area()`
- **Scatter Matrix:** `pd.plotting.scatter_matrix(df)`

## 8. Interactive Visualizations

- **Interactive Plot with Plotly:** `import plotly.express as px; fig = px.line(df, x='x', y='y'); fig.show()`
- **Bokeh Line Plot:** `from bokeh.plotting import figure, show; p = figure(); p.line(x, y); show(p)`
- **Streamlit for Web Apps:** `import streamlit as st; st.line_chart(df)`

## 9. Visualization Customization and Aesthetics

- **Seaborn Themes:** `sns.set_style('whitegrid')`
- **Matplotlib Colormap:** `plt.scatter(x, y, c=z, cmap='viridis')`
- **Seaborn Palette:** `sns.set_palette('pastel')`
- **Customizing with Matplotlib rcParams:**  
`plt.rcParams.update({'font.size': 12, 'figure.figsize': (10, 8)})`

## 10. Statistical Data Visualization

- **Distribution Plot:** `sns.distplot(data)`
- **Boxplot with Outliers:** `sns.boxplot(x='x', y='y', data=df)`
- **Swarm Plot:** `sns.swarmplot(x='x', y='y', data=df)`
- **Violin Plot with Split:** `sns.violinplot(x='x', y='y', data=df, split=True)`

## 11. Advanced Seaborn Plots

- **Regplot for Regression:** `sns.regplot(x='x', y='y', data=df)`
- **LM Plot for Linear Models:** `sns.lmplot(x='x', y='y', data=df)`
- **Cluster Map:** `sns.clustermap(data)`
- **PairGrid Customization:** `g = sns.PairGrid(df);  
g.map_upper(plt.scatter); g.map_lower(sns.kdeplot);  
g.map_diag(sns.histplot)`

## 12. Multi-plot Grids and Layouts

- **Subplots in Matplotlib:** `fig, axs = plt.subplots(2, 2)`
- **Facet Grid in Seaborn:** `g = sns.FacetGrid(df, col='col');  
g.map(plt.hist, 'target')`
- **PairGrid in Seaborn:** `g = sns.PairGrid(df); g.map(plt.scatter)`

- **JointGrid in Seaborn:** `g = sns.JointGrid(x='x', y='y', data=df);  
g.plot(sns.regplot, sns.histplot)`

### 13. Text and Annotation

- **Adding Text Annotation:** `plt.text(x, y, 'Text')`
- **Annotating with Arrows:** `plt.annotate('Text', xy=(x, y), xytext=(x2, y2), arrowprops=dict(facecolor='black'))`
- **Styling Text:** `plt.text(x, y, 'Styled text', style='italic',  
fontsize=12)`

### 14. Saving and Exporting Plots

- **Save Plot as Image File:** `plt.savefig('filename.png')`
- **Save Plotly Figure:** `fig.write_image('filename.png')`
- **Export Plot to HTML with Bokeh:** `from bokeh.io import output_file;  
output_file('plot.html')`

### 15. Specialized Plots

- **Hexbin Plot for Density:** `plt.hexbin(x, y, gridsize=30,  
cmap='Blues')`
- **Treemap with Squarify:** `import squarify; squarify.plot(sizes,  
label=labels, color=colors)`
- **Bubble Chart:** `plt.scatter(x, y, s=size)`

### 16. Plotting with Plotly

- **Interactive Scatter Plot with Plotly:** `fig = px.scatter(df, x='x',  
y='y', color='color')`
- **3D Surface Plot with Plotly:** `fig = px.surface(df, x='x', y='y',  
z='z')`
- **Candlestick Chart for Financial Data:** `fig =  
go.Figure(data=[go.Candlestick(x=df['date'], open=df['open'],  
high=df['high'], low=df['low'], close=df['close'])])`

### 17. Plotting with Bokeh

- **Bokeh Scatter Plot:** `p = figure(); p.scatter(x, y, size=10, fill_color='color')`
- **Bokeh Time Series Plot:** `p = figure(x_axis_type='datetime'); p.line(df['date'], df['value'])`
- **Adding Hover Tool in Bokeh:** `from bokeh.models import HoverTool; hover = HoverTool(tooltips=[('X-value', '@x'), ('Y-value', '@y')]); p.add_tools(hover)`

## 18. Dashboard and Reporting

- **Creating Dashboards with Dash:** `import dash; import dash_core_components as dcc; import dash_html_components as html; app = dash.Dash(); app.layout = html.Div([dcc.Graph(figure=fig)])`
- **Automated Reporting with Jupyter Notebook:** `!jupyter nbconvert --to html notebook.ipynb`

## 19. Advanced Customization

- **Creating a Custom Matplotlib Style:** `plt.style.use({'figure.facecolor': 'white', 'axes.facecolor': 'lightgray'})`
- **Customizing Seaborn Context:** `sns.set_context('talk', font_scale=1.2)`

## 20. Data Interpretation Techniques

- **Analyzing Trends in Time Series:** Identify patterns and trends using moving averages or smoothing techniques
- **Interpreting Correlation Matrices:** Use heatmaps or `corrplot` to assess relationships between variables
- **Identifying Outliers and Anomalies:** Use boxplots or scatter plots to spot outliers in data
- **Comparing Groups or Categories:** Employ bar charts or violin plots for comparison between different groups
- **Understanding Distribution of Data:** Utilize histograms, density plots, or Q-Q plots to explore data distribution
- **Analyzing Impact of a Variable:** Use bar charts, line graphs, or area plots to understand how changes in one variable affect another

## 21. Interactive Dashboard Tools

- **Building Dashboards with Tableau or Power BI:** Use business intelligence tools for creating interactive and business-focused dashboards
- **Interactive Web Dashboards with Plotly Dash:** Create web applications with interactive Plotly graphs using Dash framework
- **Real-time Data Visualization with Bokeh Server:** Deploy live data streams in visualizations using Bokeh server applications

## 22. Advanced Graphical Techniques

- **Network Graphs with NetworkX or Gephi:** Visualize complex relationships and network structures
- **Creating Geographical Maps with GeoPandas:** Use GeoPandas along with Matplotlib or Bokeh for plotting geographical data
- **3D Visualizations with Plotly or Mayavi:** Develop 3D plots for more complex data representations

## 23. Data Storytelling

- **Narrative Visualization with Sequential Panels:** Combine multiple plots with annotations to tell a story
- **Interactive Storytelling with Jupyter Widgets:** Use Jupyter widgets to create an interactive narrative around the data
- **Combining Visuals and Text in Reports:** Integrate visualizations with descriptive text for comprehensive reports

## 24. Visualization for Machine Learning

- **Feature Importance Plot:** Visualize model's feature importances to interpret which features contribute most to the prediction
- **Confusion Matrix Visualization:** Graphically represent the performance of a classification model
- **ROC Curve Plotting:** Plot ROC curves to assess the performance of binary classifiers

## 25. Performance and Scalability in Visualization

- **Optimizing Plot Performance in Matplotlib:** Use Matplotlib's interactive mode wisely for large datasets
- **Handling Large Datasets with Datashader:** Render massive datasets as images with Datashader
- **Efficient Plotting with HDF5 or Parquet Files:** Utilize HDF5 or Parquet formats for efficient loading and plotting of large data

## 26. Custom Visualization Tools and Libraries

- **Using D3.js for Custom Web Visualizations:** Leverage D3.js for intricate and interactive web visualizations
- **Highcharts or Echarts for Interactive Charts:** Use JavaScript libraries like Highcharts or Echarts for rich, interactive charts
- **Creating Custom Plots with ggplot2 in R:** For R users, utilize ggplot2 for creating sophisticated and layered graphics

## 27. Scientific and Statistical Visualization

- **Visualizing Statistical Models with Seaborn or Statsmodels:** Plot statistical estimates using Seaborn's advanced plots or Statsmodels' graphics
- **Scientific Visualization with SciPy or Matplotlib:** Use SciPy and Matplotlib for detailed scientific plots, such as spectrograms or advanced histograms