PYTHON VISUALIZATION LIBRARIES

A Comprehensive Guide to Matplotlib and Seaborn

Prepared by: Bellamkonda Nokesh
Internship: ShadowFox Data Science Intern
Date: 30-01-2026

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY

Data visualization is the cornerstone of effective data science communication. This comprehensive documentation explores two fundamental Python visualization libraries: **Matplotlib** and **Seaborn**. While Matplotlib provides granular control over every visual element, Seaborn offers statistical plotting capabilities with elegant default aesthetics. This guide covers 20+ graph types with practical code examples, use cases, and comparative insights to help data scientists choose the right tool for their visualization needs.

2. INTRODUCTION TO DATA VISUALIZATION

Data visualization transforms complex datasets into visual representations that reveal patterns, trends, and insights invisible in raw numbers. In the Python ecosystem, visualization libraries serve different purposes:

- **Matplotlib**: The foundation library offering complete control
- **Seaborn**: Statistical graphics with beautiful defaults
- **Plotly**: Interactive web-based visualizations
- **Bokeh**: Interactive plots for large datasets

This documentation focuses on Matplotlib and Seaborn, which together handle 90% of data visualization needs in data science projects.

Why These Two Libraries?

Matplotlib is essential because:

- It's the base for most other visualization libraries
- Offers unmatched customization capabilities
- Industry standard for publication-quality graphics

Seaborn complements Matplotlib by:

- Simplifying complex statistical visualizations
- Providing attractive default themes
- Integrating seamlessly with Pandas Data Frames

3. MATPLOTLIB: THE FOUNDATION OF PYTHON VISUALIZATION

Overview

Matplotlib was created in 2002 by John Hunter to provide MATLAB-like plotting capabilities in Python. Built on NumPy arrays, it has become the most widely used Python visualization library.

Key Features:

- Object-oriented and pyplot interfaces
- Support for multiple output formats (PNG, PDF, SVG)
- Extensive customization options
- Integration with GUI toolkits

- 3D plotting capabilities

Installation:

pip install matplotlib

Basic Import:

```
import matplotlib.pyplot as plt
import numpy as np
```

3.2 Matplotlib Architecture

A Matplotlib plot consists of several components:

1. **Figure**: The overall container holding all plot elements
2. **Axes**: The actual plotting area (not to be confused with axis)
3. **Axis**: The number line objects (x-axis, y-axis)
4. **Artists**: Everything visible on the figure (lines, text, etc.)

```
Figure
  └── Axes (Plot 1)
        ├── X-Axis
        ├── Y-Axis
        └── Artists (lines, markers, text)
```

MATPLOTLIB GRAPH TYPES

GRAPH TYPE 1: LINE PLOT

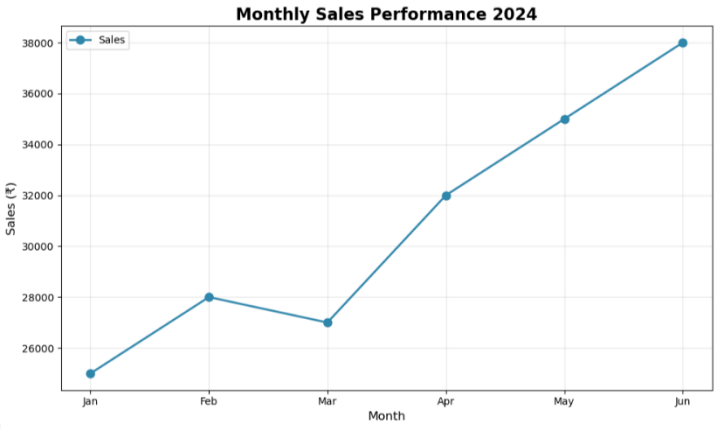
Description:
Line plots connect data points with continuous lines, ideal for showing trends over time or continuous data.

Use Case:

- Time series analysis
- Stock price trends
- Temperature variations
- Sales performance over months

Code Example:

```python
import matplotlib.pyplot as plt
import numpy as np

# Sample data: Monthly sales
months = np.array(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'])
sales = np.array([25000, 28000, 27000, 32000, 35000, 38000])

# Create line plot
plt.figure(figsize=(10, 6))
plt.plot(months, sales, marker='o', linestyle='-', linewidth=2,
         markersize=8, color='#2E86AB', label='Sales')

# Customization
plt.title('Monthly Sales Performance 2024', fontsize=16, fontweight='bold')
plt.xlabel('Month', fontsize=12)
plt.ylabel('Sales (₹)', fontsize=12)
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()

# Display
plt.show()
```



GRAPH TYPE 2: BAR CHART

Description:

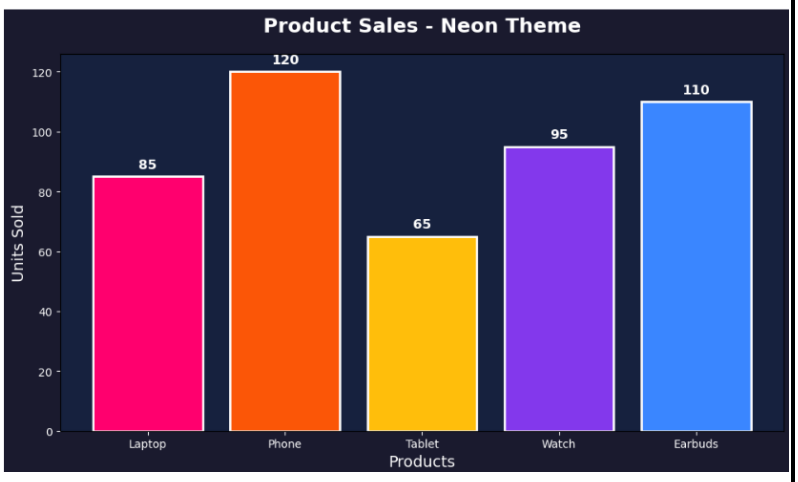
Bar charts use rectangular bars to compare categorical data, with bar heights representing values.

Use Case:

- Comparing sales across different products
- Survey response frequencies
- Performance comparison between teams
- Category-wise revenue analysis

Code Example:

```python
import matplotlib.pyplot as plt
import numpy as np

# Data
products = ['Laptop', 'Phone', 'Tablet', 'Watch', 'Earbuds']
sales = [85, 120, 65, 95, 110]
colors = ['#FF006E', '#FB5607', '#FFBE0B', '#8338EC', '#3A86FF']

# Plot
fig, ax = plt.subplots(figsize=(10, 6), facecolor='#1a1a2e')
bars = ax.bar(products, sales, color=colors, edgecolor='white', linewidth=2)

# Glow effect
for bar in bars:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 2,
            f'{height}', ha='center', va='bottom',
            fontsize=12, fontweight='bold', color='white')

ax.set_facecolor('#16213e')
ax.set_title('Product Sales - Neon Theme', fontsize=18, fontweight='bold',
             color='white', pad=20)
ax.set_xlabel('Products', fontsize=14, color='white')
ax.set_ylabel('Units Sold', fontsize=14, color='white')
ax.tick_params(colors='white')
plt.tight_layout()
plt.show()
```



GRAPH TYPE 3: HISTOGRAM

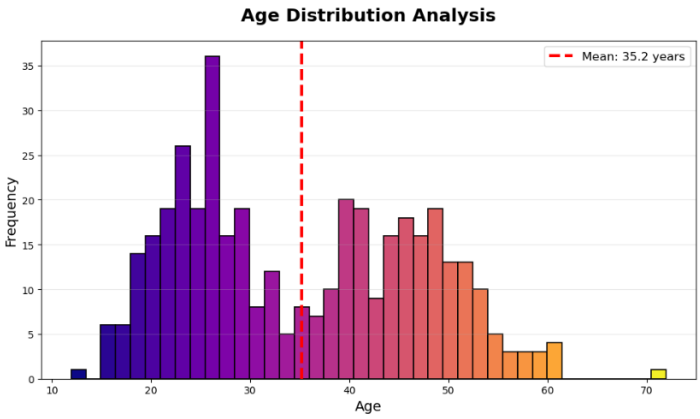
Description:
Histograms display the distribution of numerical data by grouping values into bins.

Use Case:

- Age distribution in population
- Exam score distribution
- Income level analysis
- Response time distribution

Code Example:

```python
import matplotlib.pyplot as plt
import numpy as np

# Data
np.random.seed(42)
ages = np.concatenate([np.random.normal(25, 5, 200),
                       np.random.normal(45, 7, 200)])

# Plot
plt.figure(figsize=(10, 6))
n, bins, patches = plt.hist(ages, bins=40, edgecolor='black', linewidth=1.2)

# Color gradient
cm = plt.cm.plasma
for i, patch in enumerate(patches):
    patch.set_facecolor(cm(i / len(patches)))

plt.axvline(ages.mean(), color='red', linestyle='--', linewidth=3,
            label=f'Mean: {ages.mean():.1f} years')
plt.title('Age Distribution Analysis', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Age', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.legend(fontsize=12)
plt.grid(alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```



GRAPH TYPE 4: SCATTER PLOT

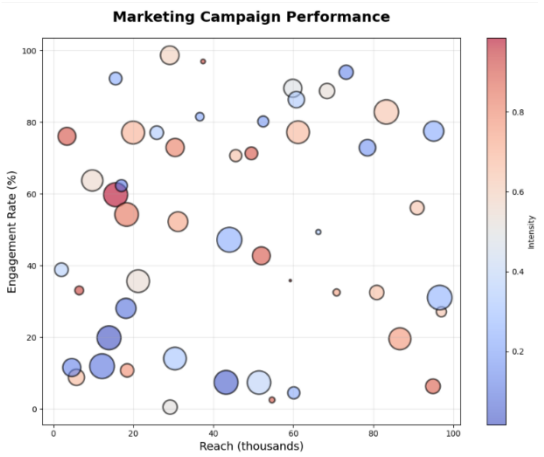
Description:
Scatter plots use dots to show the relationship between two numerical variables.

Use Case:

- Correlation analysis
- Outlier detection
- Pattern recognition
- Relationship between variables (e.g., height vs weight)

Code Example:

```
import matplotlib.pyplot as plt
import numpy as np

# Data
np.random.seed(42)
x = np.random.rand(50) * 100
y = np.random.rand(50) * 100
sizes = np.random.rand(50) * 1000
colors = np.random.rand(50)

# Plot
plt.figure(figsize=(10, 8))
scatter = plt.scatter(x, y, s=sizes, c=colors, cmap='coolwarm',
                      alpha=0.6, edgecolors='black', linewidth=2)

plt.colorbar(scatter, label='Intensity')
plt.title('Marketing Campaign Performance', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Reach (thousands)', fontsize=14)
plt.ylabel('Engagement Rate (%)', fontsize=14)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 5: PIE CHART

Description:

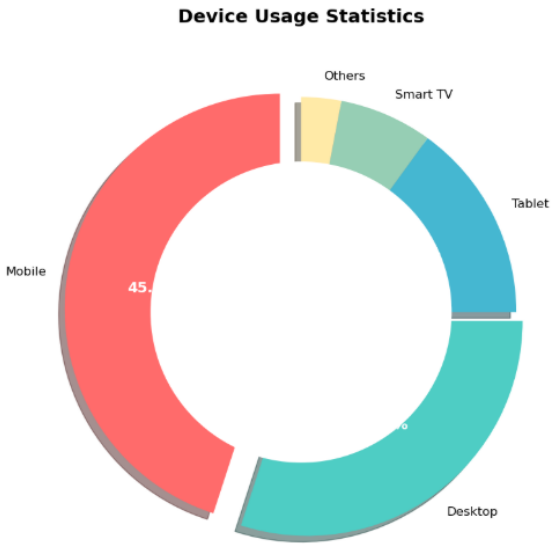
Pie charts show proportions of a whole, with each slice representing a category's percentage.

Use Case:

- Market share analysis
- Budget allocation
- Survey results
- Demographic breakdowns

Code Example:

```
import matplotlib.pyplot as plt

# Data
categories = ['Mobile', 'Desktop', 'Tablet', 'Smart TV', 'Others']
usage = [45, 30, 15, 7, 3]
colors = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4', '#FFEAA7']
explode = (0.1, 0.05, 0, 0, 0)

# Plot
fig, ax = plt.subplots(figsize=(10, 8))
wedges, texts, autotexts = ax.pie(usage, labels=categories, autopct='%1.1f%%',
                                  colors=colors, explode=explode, shadow=True,
                                  startangle=90, textprops={'fontsize': 12})

# Donut style
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig.gca().add_artist(centre_circle)

# Bold percentage
for autotext in autotexts:
    autotext.set_color('white')
    autotext.set_fontweight('bold')
    autotext.set_fontsize(14)

plt.title('Device Usage Statistics', fontsize=18, fontweight='bold', pad=20)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 6: HORIZONTAL BAR CHART

Description:

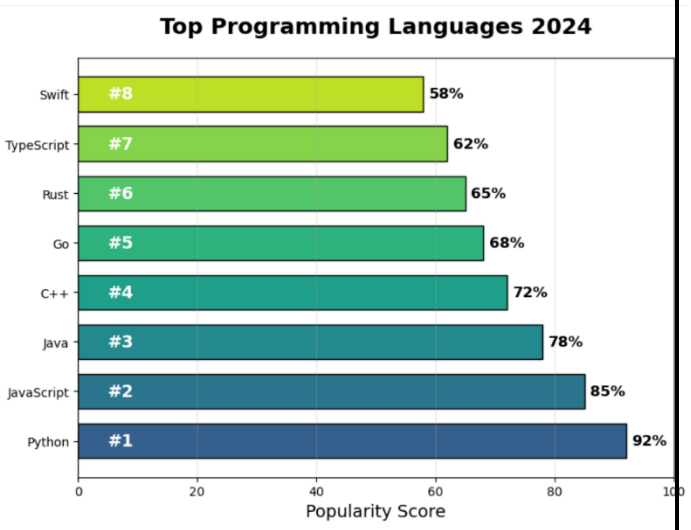
Horizontal bars are useful when category names are long or when comparing many categories.

Use Case:

- Ranking visualization
- Long category names
- Comparison of many items
- Survey responses

Code Example:

```
import matplotlib.pyplot as plt
import numpy as np

# Data
languages = ['Python', 'JavaScript', 'Java', 'C++', 'Go', 'Rust', 'TypeScript', 'Swift']
popularity = [92, 85, 78, 72, 68, 65, 62, 58]

# Plot
fig, ax = plt.subplots(figsize=(10, 8))

# Gradient colors
colors = plt.cm.viridis(np.linspace(0.3, 0.9, len(languages)))
bars = ax.barh(languages, popularity, color=colors, height=0.7, edgecolor='black')

# Add rank numbers
for i, (bar, value) in enumerate(zip(bars, popularity)):
    ax.text(5, bar.get_y() + bar.get_height()/2, f'#{i+1}',
            ha='left', va='center', fontsize=14, fontweight='bold', color='white')
    ax.text(value + 1, bar.get_y() + bar.get_height()/2, f'{value}%',
            ha='left', va='center', fontsize=12, fontweight='bold')

ax.set_xlim(0, 100)
ax.set_title('Top Programming Languages 2024', fontsize=18, fontweight='bold', pad=20)
ax.set_xlabel('Popularity Score', fontsize=14)
ax.grid(alpha=0.3, axis='x')
plt.tight_layout()
plt.show()
```



GRAPH TYPE 7: AREA CHART

Description:

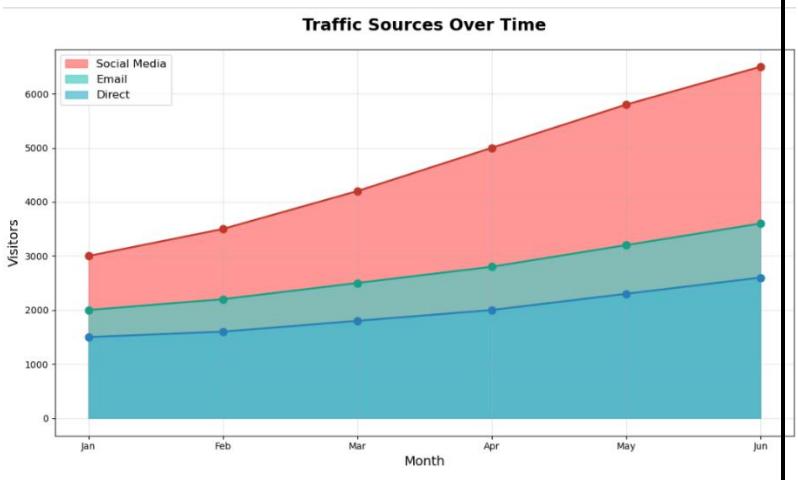
Area charts show quantitative data over time with the area below the line filled.

Use Case:

- Cumulative trends
- Multiple time series comparison
- Volume over time
- Stacked data representation

Code Example:

```
import matplotlib.pyplot as plt
import numpy as np

# Data
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
social_media = [3000, 3500, 4200, 5000, 5800, 6500]
email = [2000, 2200, 2500, 2800, 3200, 3600]
direct = [1500, 1600, 1800, 2000, 2300, 2600]

# Plot
plt.figure(figsize=(12, 7))
plt.fill_between(range(len(months)), 0, social_media, alpha=0.7,
                 label='Social Media', color='#FF6B6B')
plt.fill_between(range(len(months)), 0, email, alpha=0.7,
                 label='Email', color='#4ECDC4')
plt.fill_between(range(len(months)), 0, direct, alpha=0.7,
                 label='Direct', color='#45B7D1')

plt.plot(range(len(months)), social_media, 'o-', color='#C0392B', linewidth=2, markersize=8)
plt.plot(range(len(months)), email, 'o-', color='#16A085', linewidth=2, markersize=8)
plt.plot(range(len(months)), direct, 'o-', color='#2980B9', linewidth=2, markersize=8)

plt.xticks(range(len(months)), months)
plt.title('Traffic Sources Over Time', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Visitors', fontsize=14)
plt.legend(fontsize=12, loc='upper left')
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 8: BOX PLOT

Description:

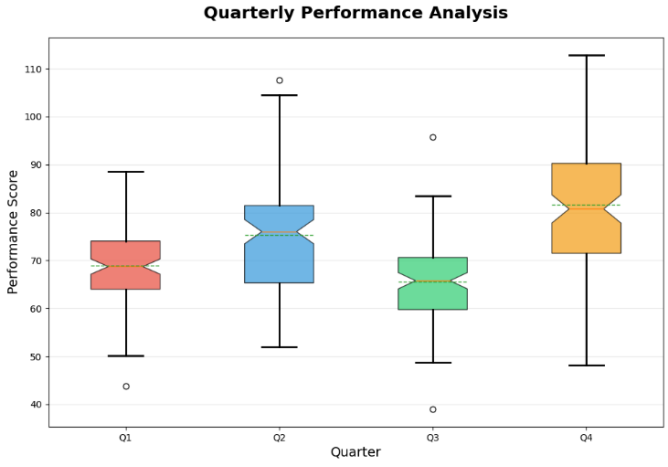
Box plots display the distribution of data through quartiles, showing median, outliers, and spread.

Use Case:

- Identifying outliers
- Comparing distributions across groups
- Statistical analysis
- Quality control

Code Example:

```python
import matplotlib.pyplot as plt
import numpy as np

# Data
np.random.seed(42)
data = [np.random.normal(70, 10, 100),
        np.random.normal(75, 12, 100),
        np.random.normal(65, 8, 100),
        np.random.normal(80, 15, 100)]
labels = ['Q1', 'Q2', 'Q3', 'Q4']

# Plot
fig, ax = plt.subplots(figsize=(10, 7))
bp = ax.boxplot(data, labels=labels, patch_artist=True, notch=True,
                showmeans=True, meanline=True)

# Colors
colors = ['#E74C3C', '#3498DB', '#2ECC71', '#F39C12']
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)
    patch.set_alpha(0.7)

# Style whiskers and caps
for whisker in bp['whiskers']:
    whisker.set(linewidth=2, linestyle='-', color='black')
for cap in bp['caps']:
    cap.set(linewidth=2, color='black')

ax.set_title('Quarterly Performance Analysis', fontsize=18, fontweight='bold', pad=20)
ax.set_xlabel('Quarter', fontsize=14)
ax.set_ylabel('Performance Score', fontsize=14)
ax.grid(alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```



GRAPH TYPE 9: SUBPLOT - MULTIPLE GRAPHS

Description:

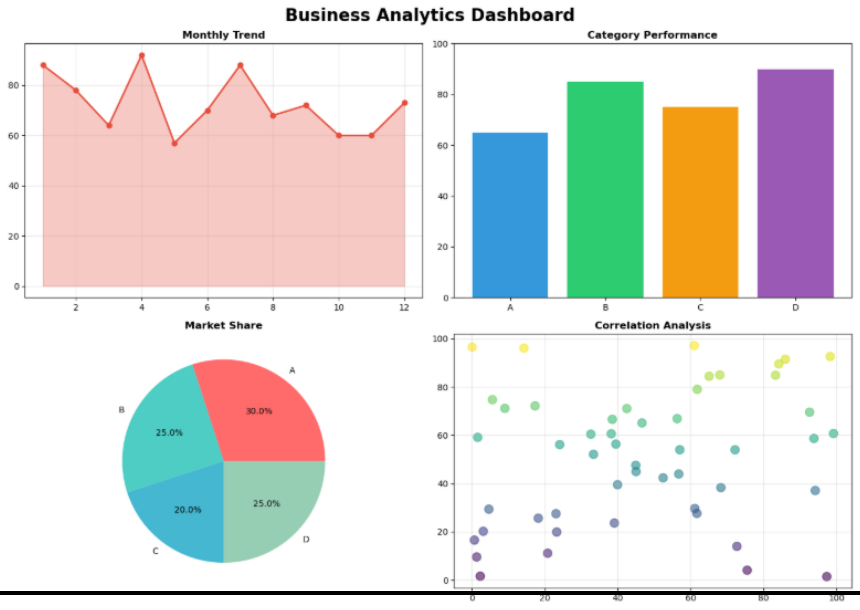
Subplots allow multiple graphs in a single figure for comprehensive analysis.

Use Case:

- Dashboard creation
- Comparative analysis
- Multi-dimensional data exploration
- Report generation

Code Example:

```python
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(42)

# Create figure
fig = plt.figure(figsize=(14, 10))
fig.suptitle('Business Analytics Dashboard', fontsize=20, fontweight='bold')

# Subplot 1: Line Chart
ax1 = plt.subplot(2, 2, 1)
x = np.arange(1, 13)
y = np.random.randint(50, 100, 12)
ax1.plot(x, y, marker='o', linewidth=2, color='#E74C3C')
ax1.fill_between(x, y, alpha=0.3, color='#E74C3C')
ax1.set_title('Monthly Trend', fontweight='bold')
ax1.grid(alpha=0.3)

# Subplot 2: Bar Chart
ax2 = plt.subplot(2, 2, 2)
categories = ['A', 'B', 'C', 'D']
values = [65, 85, 75, 90]
ax2.bar(categories, values, color=['#3498DB', '#2ECC71', '#F39C12', '#9B59B6'])
ax2.set_title('Category Performance', fontweight='bold')
ax2.set_ylim(0, 100)

# Subplot 3: Pie Chart
ax3 = plt.subplot(2, 2, 3)
sizes = [30, 25, 20, 25]
colors = ['#FF6B6B', '#4ECDC4', '#45B7D1', '#96CEB4']
ax3.pie(sizes, labels=['A', 'B', 'C', 'D'], autopct='%1.1f%%', colors=colors)
ax3.set_title('Market Share', fontweight='bold')

# Subplot 4: Scatter
ax4 = plt.subplot(2, 2, 4)
x = np.random.rand(50) * 100
y = np.random.rand(50) * 100
ax4.scatter(x, y, c=y, cmap='viridis', s=100, alpha=0.6)
ax4.set_title('Correlation Analysis', fontweight='bold')
ax4.grid(alpha=0.3)

plt.tight_layout()
plt.show()
```



GRAPH TYPE 10: HEATMAP

Description:

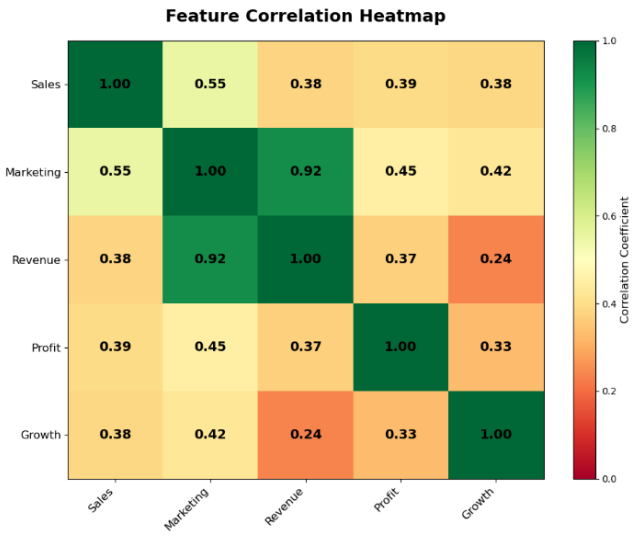
Heatmaps use color intensity to represent data values in a matrix format.

Use Case:

- Correlation matrices
- Confusion matrices
- Time-based patterns (day of week vs hour)
- Geographic data

Code Example:

```
import matplotlib.pyplot as plt
import numpy as np

# Data
np.random.seed(42)
features = ['Sales', 'Marketing', 'Revenue', 'Profit', 'Growth']
corr_matrix = np.random.rand(5, 5)
corr_matrix = (corr_matrix + corr_matrix.T) / 2
np.fill_diagonal(corr_matrix, 1)

# Plot
fig, ax = plt.subplots(figsize=(10, 8))
im = ax.imshow(corr_matrix, cmap='RdYlGn', aspect='auto', vmin=0, vmax=1)

# Ticks
ax.set_xticks(np.arange(len(features)))
ax.set_yticks(np.arange(len(features)))
ax.set_xticklabels(features, fontsize=12)
ax.set_yticklabels(features, fontsize=12)

# Rotate labels
plt.setp(ax.get_xticklabels(), rotation=45, ha="right")

# Annotations
for i in range(len(features)):
    for j in range(len(features)):
        text = ax.text(j, i, f'{corr_matrix[i, j]:.2f}',
                       ha="center", va="center", color="black",
                       fontsize=14, fontweight='bold')

# Colorbar
cbar = plt.colorbar(im, ax=ax)
cbar.set_label('Correlation Coefficient', fontsize=12)

ax.set_title('Feature Correlation Heatmap', fontsize=18, fontweight='bold', pad=20)
plt.tight_layout()
plt.show()
```



4. SEABORN: STATISTICAL DATA VISUALIZATION

Overview

Seaborn is built on top of Matplotlib and provides a high-level interface for creating attractive statistical graphics. Created by Michael Waskom, it integrates closely with Pandas DataFrames.

Key Features:

- Beautiful default styles and color palettes
- Statistical plotting functions
- Built-in themes
- Automatic legend generation
- Seamless Pandas integration

Installation:

pip install seaborn

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

4.2 Seaborn Plot Categories

Seaborn organizes plots into categories:

1. **Relational Plots**: Show relationships (scatterplot, lineplot)
2. **Categorical Plots**: Compare categories (barplot, boxplot, violinplot)
3. **Distribution Plots**: Show distributions (histplot, kdeplot)
4. **Regression Plots**: Show statistical relationships (regplot, lmplot)
5. **Matrix Plots**: Show matrix data (heatmap, clustermap)

4.3 SEABORN GRAPH TYPES

GRAPH TYPE 1: SCATTER PLOT

Description:

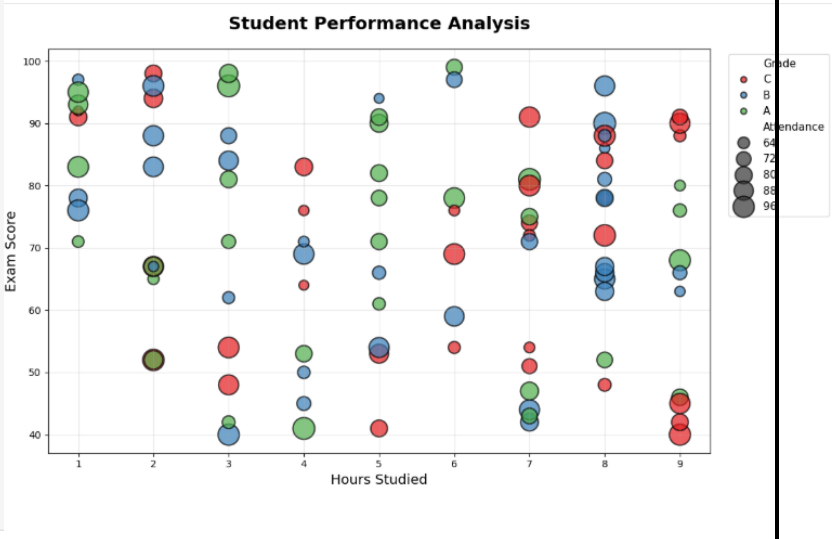
Seaborn scatter plots with enhanced statistical features and automatic legend handling.

Use Case:

- Multi-dimensional data exploration
- Group comparison
- Pattern recognition with categories

Code Example:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
df = pd.DataFrame({
    'Hours_Studied': np.random.randint(1, 10, 100),
    'Exam_Score': np.random.randint(40, 100, 100),
    'Attendance': np.random.randint(60, 100, 100),
    'Grade': np.random.choice(['A', 'B', 'C'], 100)
})

# Plot
plt.figure(figsize=(12, 7))
sns.scatterplot(data=df, x='Hours_Studied', y='Exam_Score',
                size='Attendance', hue='Grade', sizes=(100, 500),
                palette='Set1', alpha=0.7, edgecolor='black', linewidth=1.5)

plt.title('Student Performance Analysis', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Hours Studied', fontsize=14)
plt.ylabel('Exam Score', fontsize=14)
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', fontsize=11)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 2: LINE PLOT

Description:

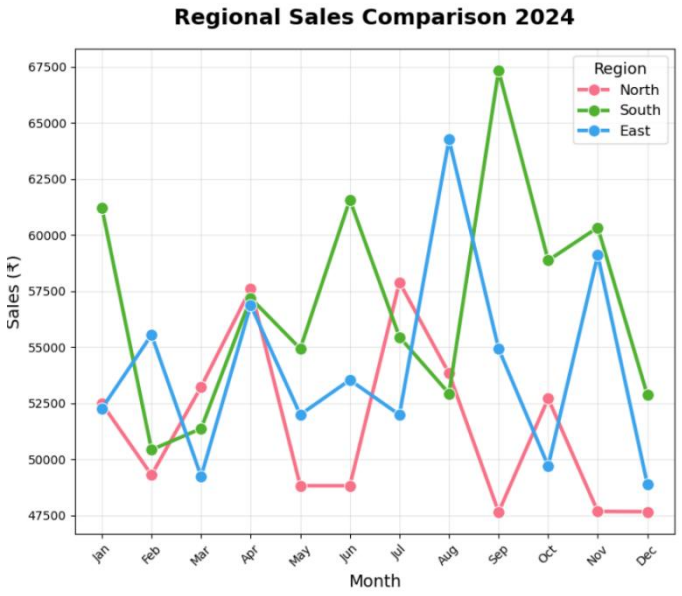
Seaborn line plots with confidence intervals and multiple series support.

Use Case:

- Time series with uncertainty
- Trend comparison across groups
- Average trends with variability

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
df = pd.DataFrame({
    'Month': months * 3,
    'Sales': np.concatenate([
        np.random.normal(50000, 5000, 12),
        np.random.normal(60000, 5000, 12),
        np.random.normal(55000, 5000, 12)
    ]),
    'Region': ['North']*12 + ['South']*12 + ['East']*12
})

# Plot
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x='Month', y='Sales', hue='Region',
             marker='o', markersize=10, linewidth=3, palette='husl')

plt.title('Regional Sales Comparison 2024', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Sales (₹)', fontsize=14)
plt.xticks(rotation=45)
plt.legend(title='Region', fontsize=12, title_fontsize=13)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 3: BAR PLOT

Description:

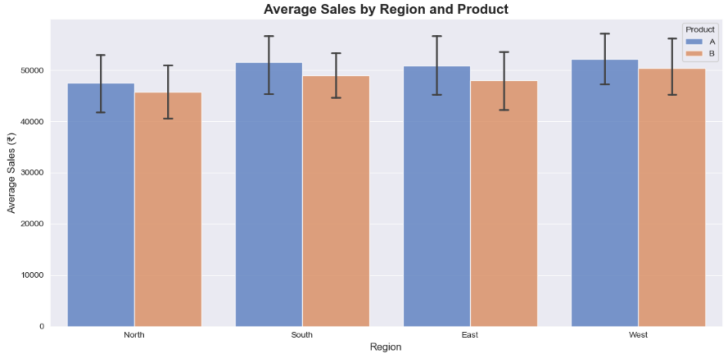
Seaborn bar plots with automatic statistical aggregation and error bars.

Use Case:

- Mean comparison with confidence intervals
- Categorical data summarization
- Group-wise averages

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Sample data: Sales by region and product
np.random.seed(42)
df = pd.DataFrame({
    'Region': np.repeat(['North', 'South', 'East', 'West'], 50),
    'Product': np.tile(np.repeat(['A', 'B'], 25), 4),
    'Sales': np.random.normal(50000, 15000, 200)
})

# Set style
sns.set_style("darkgrid")
sns.set_palette("muted")

# Create bar plot
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x='Region', y='Sales', hue='Product',
            ci=95, capsize=0.1, errwidth=2, alpha=0.8)

# Customization
plt.title('Average Sales by Region and Product', fontsize=16, fontweight='bold')
plt.xlabel('Region', fontsize=12)
plt.ylabel('Average Sales (₹)', fontsize=12)
plt.legend(title='Product', fontsize=10)
plt.tight_layout()

plt.show()
```



GRAPH TYPE 4: COUNT PLOT

Description:

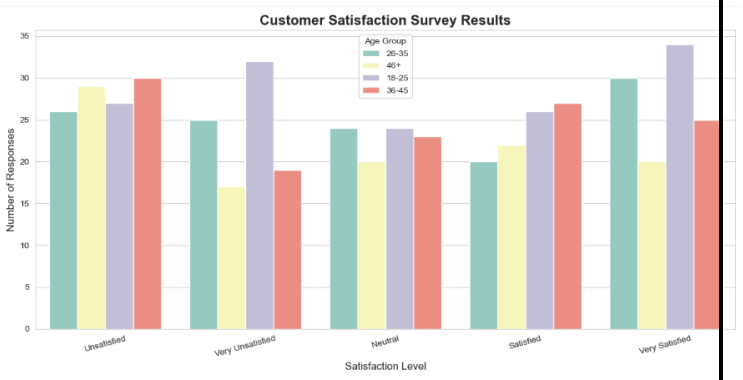
Count plots show the frequency of categorical variables.

Use Case:

- Survey response counts
- Categorical data distribution
- Frequency analysis

Code Example:

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Sample data: Customer feedback
np.random.seed(42)
df = pd.DataFrame({
    'Satisfaction': np.random.choice(['Very Satisfied', 'Satisfied', 'Neutral',
                                      'Unsatisfied', 'Very Unsatisfied'], 500),
    'Age_Group': np.random.choice(['18-25', '26-35', '36-45', '46+'], 500)
})

# Set style
sns.set_style("whitegrid")

# Create count plot
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Satisfaction', hue='Age_Group', palette='Set3')

# Customization
plt.title('Customer Satisfaction Survey Results', fontsize=16, fontweight='bold')
plt.xlabel('Satisfaction Level', fontsize=12)
plt.ylabel('Number of Responses', fontsize=12)
plt.legend(title='Age Group', fontsize=10)
plt.xticks(rotation=15)
plt.tight_layout()

plt.show()
```



GRAPH TYPE 5: BOX PLOT

Description:

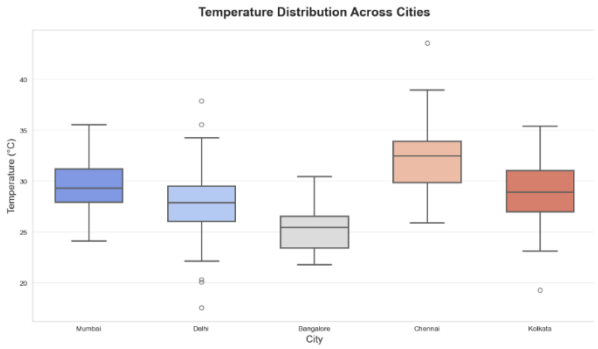
Seaborn box plots with enhanced aesthetics and easy group comparisons.

Use Case:

- Distribution comparison across categories
- Outlier detection by group
- Statistical summary visualization

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
df = pd.DataFrame({
    'City': np.repeat(['Mumbai', 'Delhi', 'Bangalore', 'Chennai', 'Kolkata'], 60),
    'Temperature': np.concatenate([
        np.random.normal(30, 3, 60),
        np.random.normal(28, 4, 60),
        np.random.normal(25, 2, 60),
        np.random.normal(32, 3, 60),
        np.random.normal(29, 3, 60)
    ])
})

# Plot
plt.figure(figsize=(12, 7))
sns.boxplot(data=df, x='City', y='Temperature',
            palette='coolwarm', linewidth=2, width=0.6)

plt.title('Temperature Distribution Across Cities', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('City', fontsize=14)
plt.ylabel('Temperature (°C)', fontsize=14)
plt.grid(alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```



GRAPH TYPE 6: VIOLIN PLOT

Description:

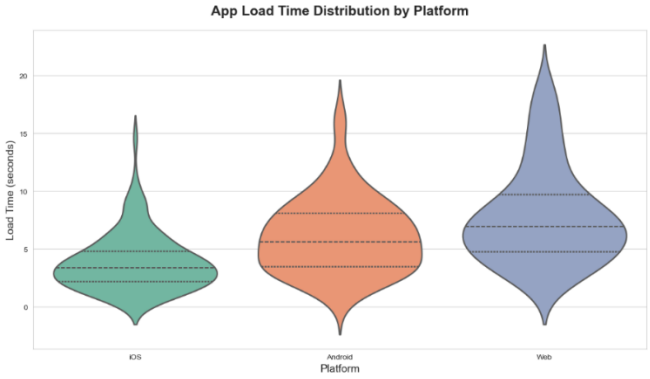
Violin plots combine box plots with kernel density plots to show distribution shape.

Use Case:

- Understanding distribution density
- Comparing distribution shapes
- Multimodal distribution detection

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
df = pd.DataFrame({
    'Platform': np.repeat(['iOS', 'Android', 'Web'], 100),
    'Load_Time': np.concatenate([
        np.random.gamma(2, 2, 100),
        np.random.gamma(3, 2, 100),
        np.random.gamma(4, 2, 100)
    ])
})

# Plot
plt.figure(figsize=(12, 7))
sns.violinplot(data=df, x='Platform', y='Load_Time',
               palette='Set2', inner='quartile', linewidth=2)

plt.title('App Load Time Distribution by Platform', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Platform', fontsize=14)
plt.ylabel('Load Time (seconds)', fontsize=14)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 7: HISTOGRAM WITH KDE

Description:

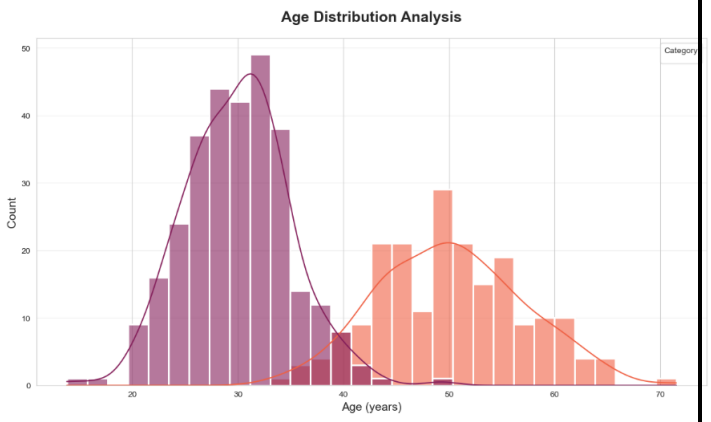
Seaborn histograms with kernel density estimation overlay.

Use Case:

- Understanding data distribution
- Identifying skewness
- Comparing theoretical vs actual distribution

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
df = pd.DataFrame({
    'Age': np.concatenate([
        np.random.normal(30, 5, 300),
        np.random.normal(50, 7, 200)
    ]),
    'Category': ['Young']*300 + ['Senior']*200
})

# Plot
plt.figure(figsize=(12, 7))
sns.histplot(data=df, x='Age', hue='Category', kde=True,
             bins=30, palette='rocket', alpha=0.6, linewidth=2)

plt.title('Age Distribution Analysis', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Age (years)', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.legend(title='Category', fontsize=12)
plt.grid(alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```



GRAPH TYPE 8: HEATMAP

Description:

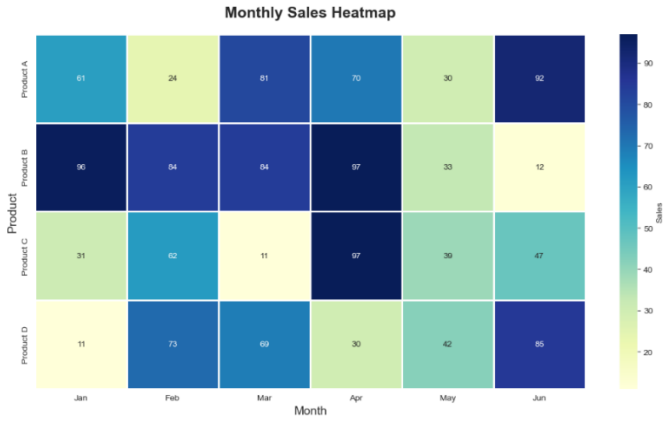
Seaborn heatmaps with advanced customization and annotations.

Use Case:

- Correlation matrices
- Pivot table visualization
- Confusion matrices
- Time-based patterns

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
products = ['Product A', 'Product B', 'Product C', 'Product D']
data = np.random.randint(10, 100, size=(len(products), len(months)))

df = pd.DataFrame(data, index=products, columns=months)

# Plot
plt.figure(figsize=(12, 7))
sns.heatmap(df, annot=True, fmt='d', cmap='YlGnBu',
            linewidths=2, linecolor='white', cbar_kws={'label': 'Sales'})

plt.title('Monthly Sales Heatmap', fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Product', fontsize=14)
plt.tight_layout()
plt.show()
```



GRAPH TYPE 10: REGRESSION PLOT

Description:

Regression plots show linear relationships with confidence intervals.

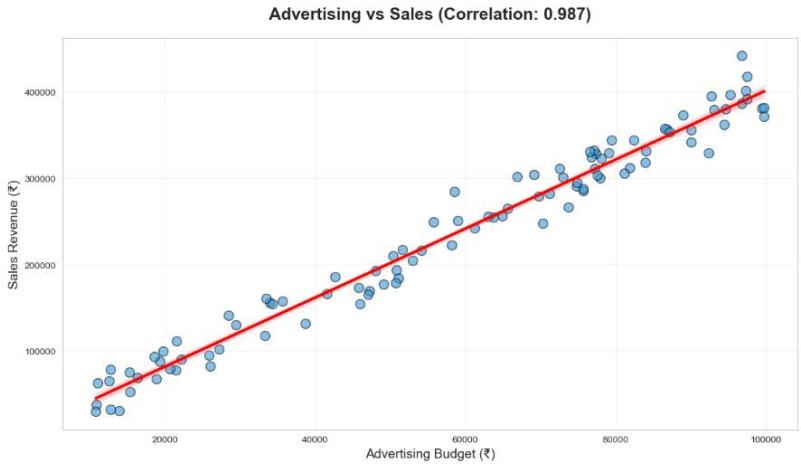
Use Case:

- Trend analysis

- Prediction visualization
- Correlation strength assessment

Code Example:

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
np.random.seed(42)
df = pd.DataFrame({
    'Advertising_Budget': np.random.randint(10000, 100000, 100),
    'Sales': np.random.randint(50000, 500000, 100)
})
df['Sales'] = df['Advertising_Budget'] * 4 + np.random.normal(0, 20000, 100)

# Plot
plt.figure(figsize=(12, 7))
sns.regplot(data=df, x='Advertising_Budget', y='Sales',
            scatter_kws={'s': 100, 'alpha': 0.6, 'edgecolor': 'black'},
            line_kws={'color': 'red', 'linewidth': 3},
            color='#3498DB')

# Calculate correlation
corr = df['Advertising_Budget'].corr(df['Sales'])

plt.title(f'Advertising vs Sales (Correlation: {corr:.3f})',
          fontsize=18, fontweight='bold', pad=20)
plt.xlabel('Advertising Budget (₹)', fontsize=14)
plt.ylabel('Sales Revenue (₹)', fontsize=14)
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



5. COMPARATIVE ANALYSIS: MATPLOTLIB VS SEABORN

Feature Comparison Table

Feature	Matplotlib	Seaborn
Learning Curve	Moderate to Steep	Easy to Moderate
Customization	Extremely High	Moderate (can use Matplotlib underneath)
Default Aesthetics	Basic	Beautiful & Modern
Statistical Functions	Manual implementation	Built-in
Pandas Integration	Requires conversion	Native support
Plot Types	100+	20+ (specialized)
Interactivity	Limited (with mplcursors)	Limited
Performance	Excellent	Good (slightly slower)
Documentation	Extensive	Good
Code Length	Longer	Shorter
3D Plotting	Yes (mpl_toolkits)	No
Themes	Manual	Built-in (5 themes)
Best For	Complete control, Publication	Quick EDA, Statistical viz

Detailed Comparison

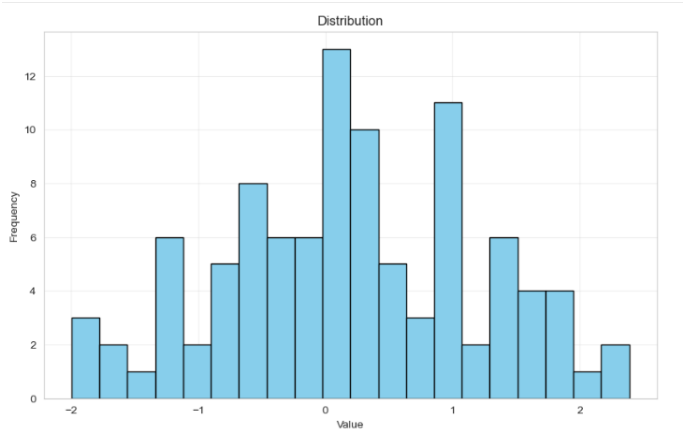
A. Ease of Use

Matplotlib:

- Requires more code for complex visualizations
- Steep learning curve for beginners
- Need to understand figure, axes architecture
- More manual configuration needed

Example:

```
# Matplotlib - More code
import matplotlib.pyplot as plt
import numpy as np

data = np.random.randn(100)
plt.figure(figsize=(10, 6))
plt.hist(data, bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Distribution')
plt.grid(True, alpha=0.3)
plt.show()
```

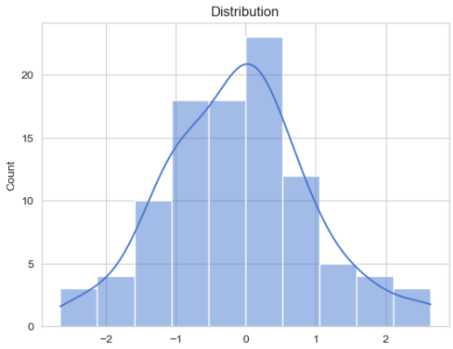


Seaborn:

- Simpler syntax for complex plots
- Easier for beginners
- Automatic statistical calculations
- Less boilerplate code

Example:

```
# Seaborn -
import seaborn as sns
import numpy as np

data = np.random.randn(100)
sns.histplot(data, kde=True)
plt.title('Distribution')
plt.show()
```



Use Both When:

1. Start with Seaborn for quick insights
2. Fine-tune with Matplotlib for customization
3. Best practice: Use Seaborn for the plot, Matplotlib for annotations

7. CONCLUSION

Both Matplotlib and Seaborn are indispensable tools in a data scientist's toolkit. Matplotlib offers unparalleled control and customization, making it ideal for creating publication-ready graphics and handling complex visualization requirements. Seaborn excels at statistical visualizations with its beautiful default aesthetics and seamless Pandas integration, perfect for rapid exploratory data analysis.

Key Takeaways:

- **Matplotlib** = Complete control + Customization + Performance
- **Seaborn** = Quick insights + Beautiful defaults + Statistical focus
- **Best Strategy** = Use both together for maximum effectiveness

The choice between libraries depends on your specific needs:

- Need speed and beauty? → **Seaborn**
- Need control and precision? → **Matplotlib**
- Need both? → **Use them together**

As you progress in your data science journey, mastering both libraries will enable you to create compelling, informative visualizations that effectively communicate insights from data.

8. REFERENCES

1. Matplotlib Official Documentation: https://matplotlib.org/
2. Seaborn Official Documentation: https://seaborn.pydata.org/
3. Python Data Science Handbook by Jake VanderPlas
4. Effective Data Visualization by Stephanie D.H. Evergreen
5. Pandas Documentation: https://pandas.pydata.org/
6. NumPy Documentation: https://numpy.org/
7. ColorBrewer: https://colorbrewer2.org/
8. Matplotlib Cheat Sheet: https://github.com/matplotlib/cheatsheets