@RAMCHANDRAPADWAL

Practical Guide to Seaborn for Data Science

A STEP-BY-STEP GUIDE

Table of Contents

@RAMCHANDRAPADWAL

CHAPTER N.1

Introduction to Seaborn

A Step-by-Step Guide

1.1 WHAT IS SEABORN?

Data visualization is a crucial component of data science, as it allows us to understand and communicate insights from the data effectively. Seaborn is a Python data visualization library that builds on top of Matplotlib and provides a high-level interface for creating visually appealing and informative plots. In this practical guide, we will explore the various functionalities of Seaborn and learn how to use it to create stunning visualizations for data analysis in data science projects.

CHAPTER N.2

Installation and Setup

A Step-by-Step Guide

2.1 Installing Seaborn

Before we dive into Seaborn, let's make sure we have it installed on our system. Seaborn can be installed using pip, and it also requires Matplotlib and NumPy as dependencies.

```
pip install seaborn
```

2.2 Importing Seaborn

Before using Matplotlib, import it into your Python script or notebook using the following statement:

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

@RAMCHANDRAPADWAL

CHAPTER N.3

Basic Plots with Seaborn

A Step-by-Step Guide

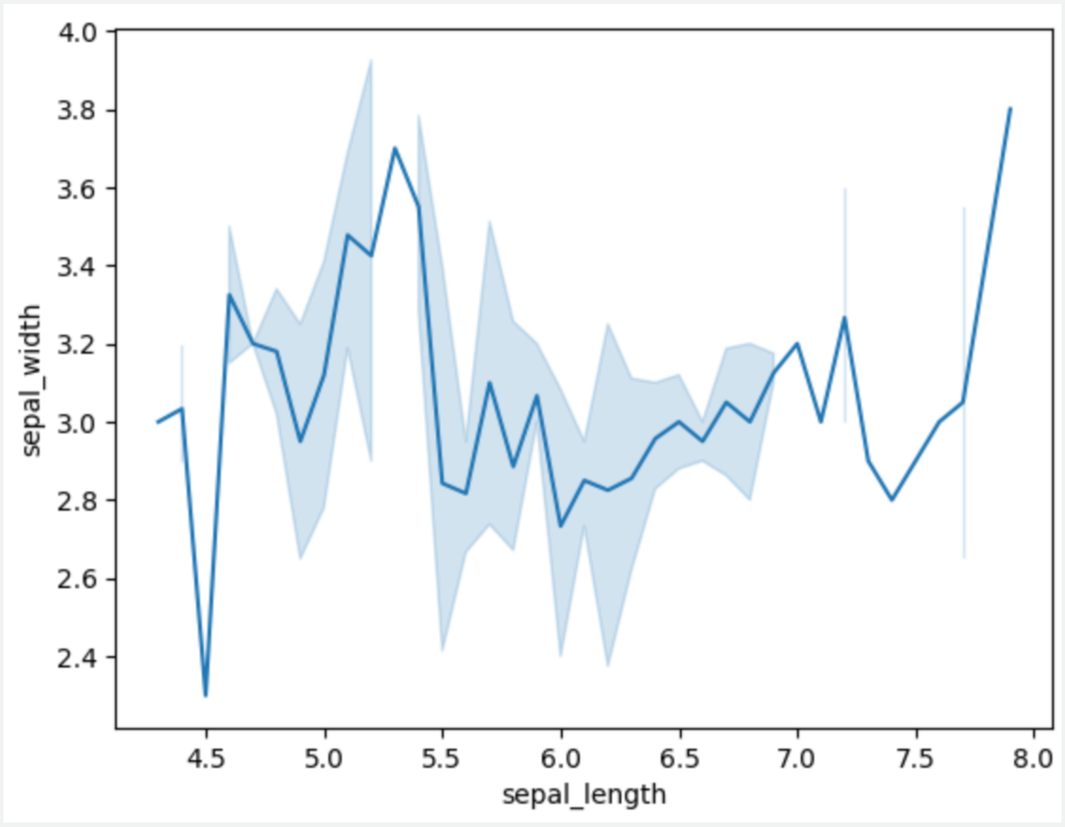
3.1 Line Plots

Line plots are useful for visualizing the trend or pattern in continuous data over a continuous interval. Seaborn provides an easy way to create line plots using the **lineplot()** function.

EXAMPLE:

```python
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt

# loading dataset
data = sns.load_dataset("iris")

# draw lineplot
sns.lineplot(x="sepal_length", y="sepal_width", data=data)
plt.show()
```

OUTPUT:



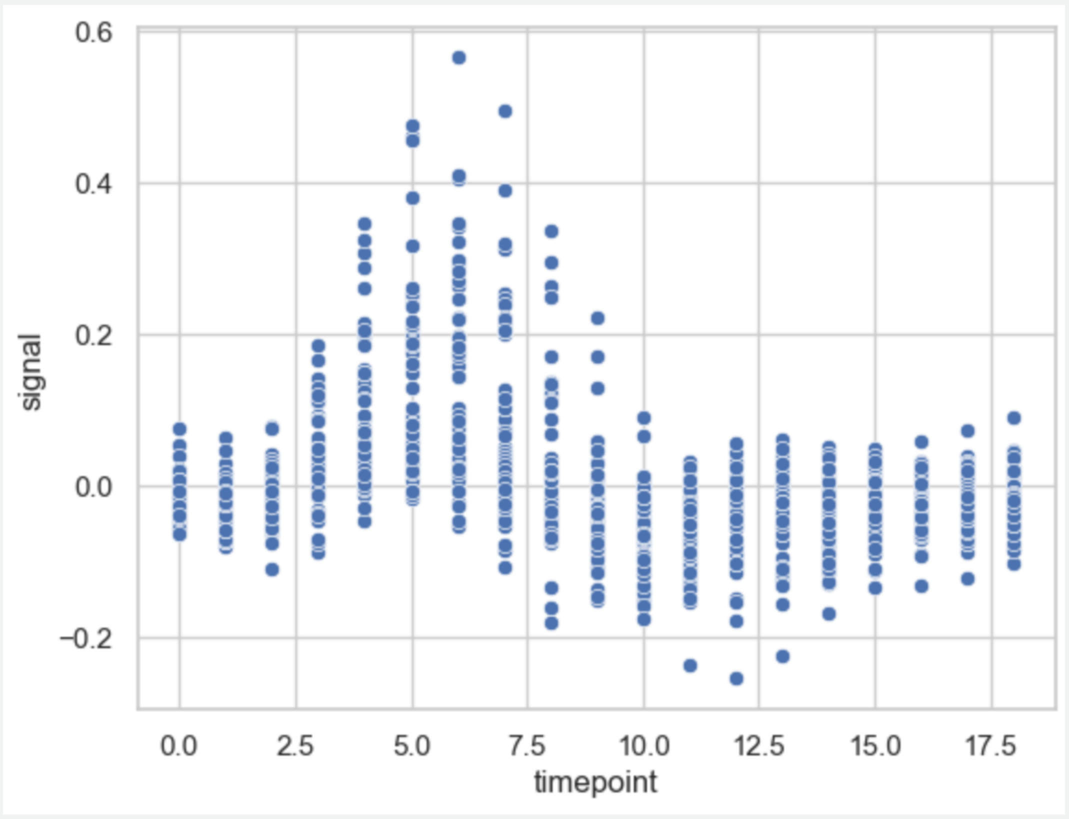
3.2 Scatter Plots

Scatter plots are ideal for visualizing the relationship between two continuous variables. Seaborn's **scatterplot()** function makes it easy to create scatter plots.

EXAMPLE:

```
import seaborn

seaborn.set(style='whitegrid')
fmri = seaborn.load_dataset("fmri")

seaborn.scatterplot(x="timepoint",
                    y="signal",
                    data=fmri)
```

OUTPUT:



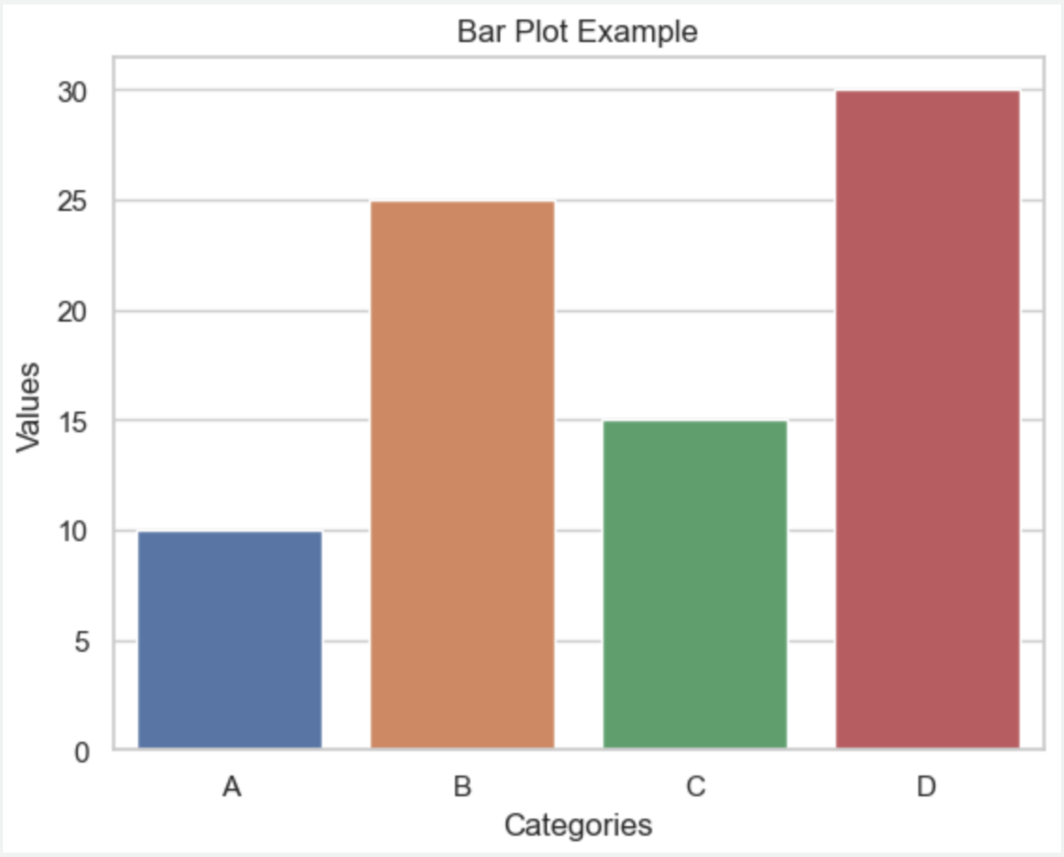
3.3 Bar Plots

Bar plots are used to compare categorical data. Seaborn's **barplot()** function allows you to create bar plots with ease.

EXAMPLE:

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

# Sample data
categories = ['A', 'B', 'C', 'D']
values = [10, 25, 15, 30]

# Create a bar plot
sns.barplot(x=categories, y=values)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Bar Plot Example')
plt.show()
```

OUTPUT:



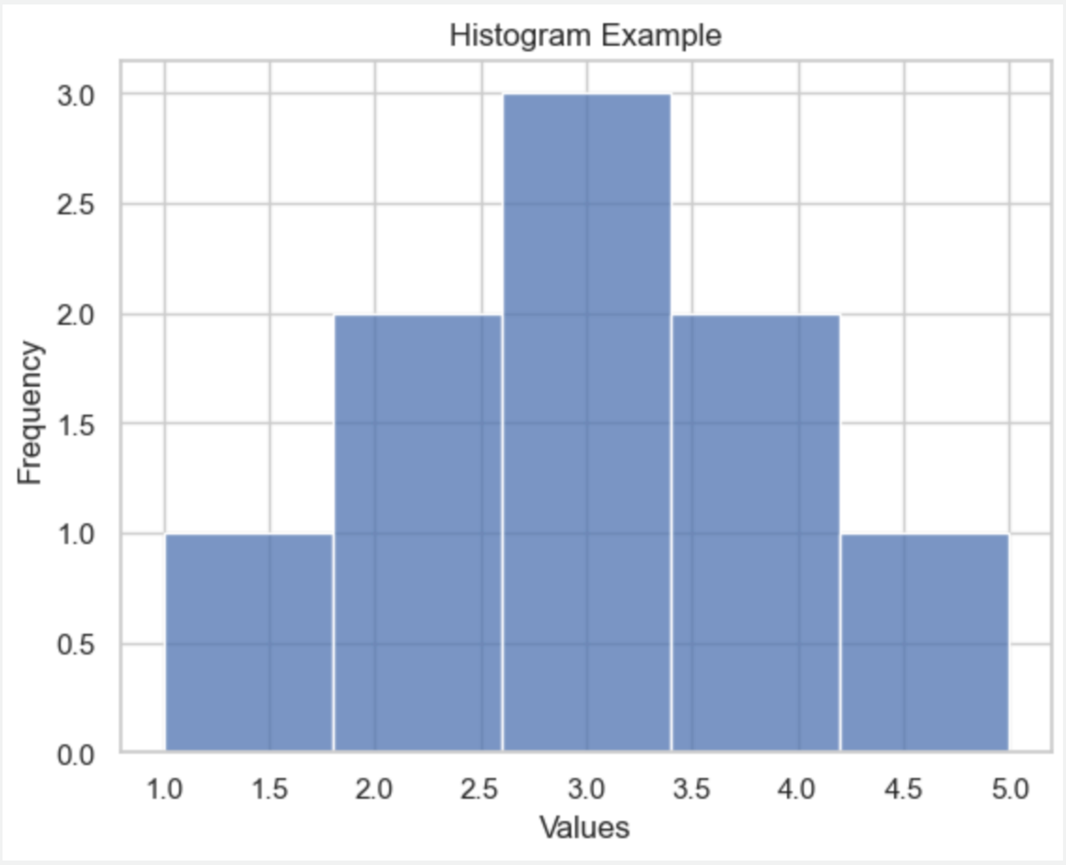
3.4 Histograms

Histograms are useful for visualizing the distribution of a continuous variable. Seaborn's **histplot()** function can be used to create histograms.

EXAMPLE:

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

# Sample data
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]

# Create a histogram
sns.histplot(data, bins=5)
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Histogram Example')
plt.show()
```

OUTPUT:



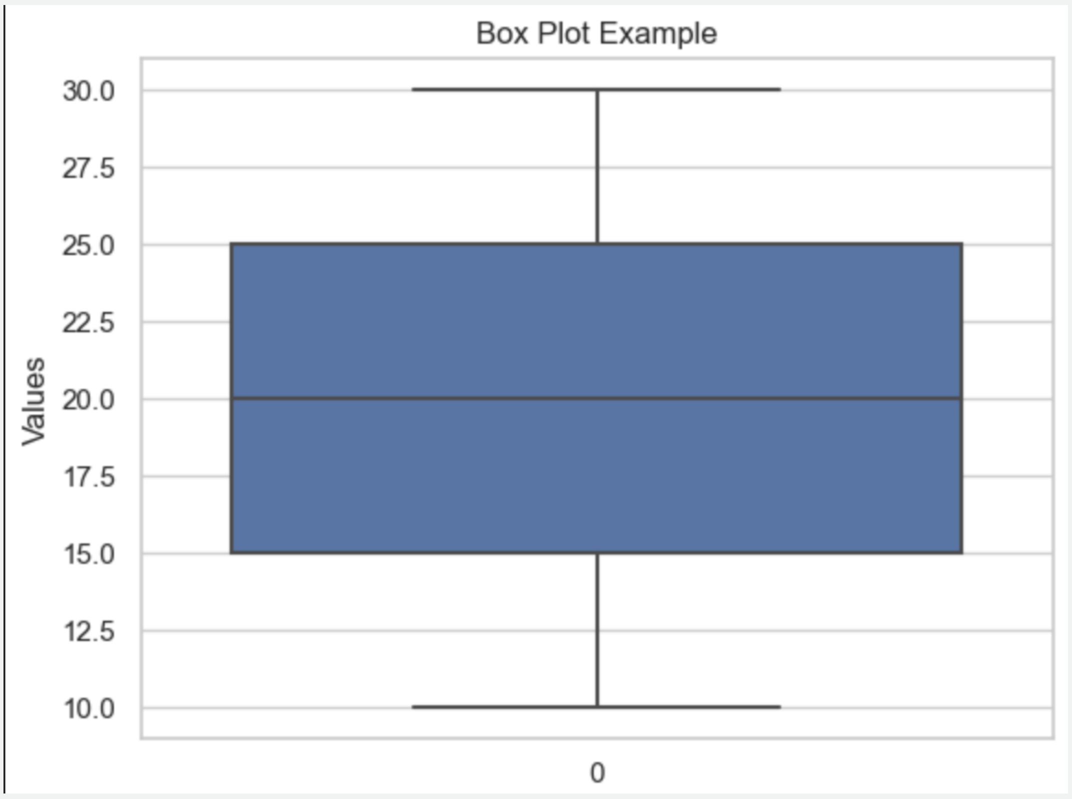
3.5 Box Plots

Box plots (box-and-whisker plots) provide a visual summary of the data's distribution. Seaborn's **boxplot()** function can be used to create box plots.

EXAMPLE:

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

# Sample data
data = [10, 15, 20, 25, 30]

# Create a box plot
sns.boxplot(data)
plt.ylabel('Values')
plt.title('Box Plot Example')
plt.show()
```

OUTPUT:



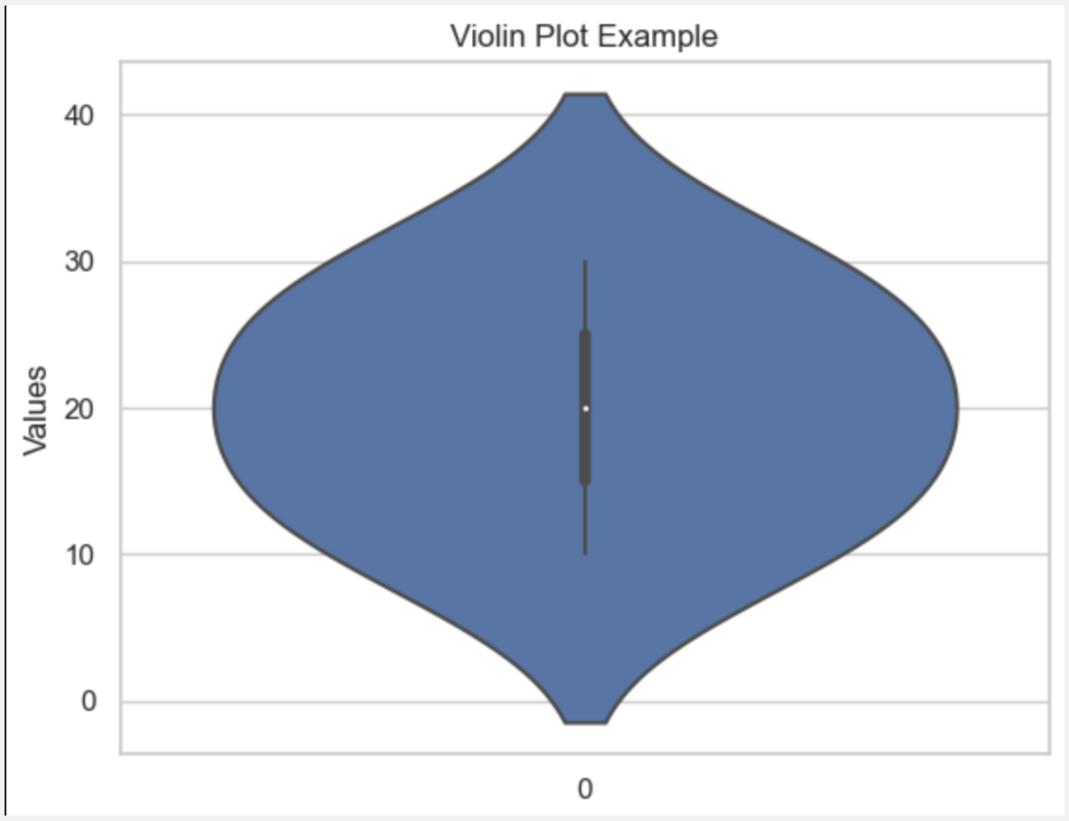
3.6 Violin Plots

Violin plots combine a box plot with a kernel density plot to provide a more detailed view of the data's distribution. Seaborn's **violinplot()** function can be used to create violin plots.

EXAMPLE:

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

# Sample data
data = [10, 15, 20, 25, 30]

# Create a violin plot
sns.violinplot(data)
plt.ylabel('Values')
plt.title('Violin Plot Example')
plt.show()
```

OUTPUT:



@RAMCHANDRAPADWAL

CHAPTER N.4

Customizing Plots

A Step-by-Step Guide

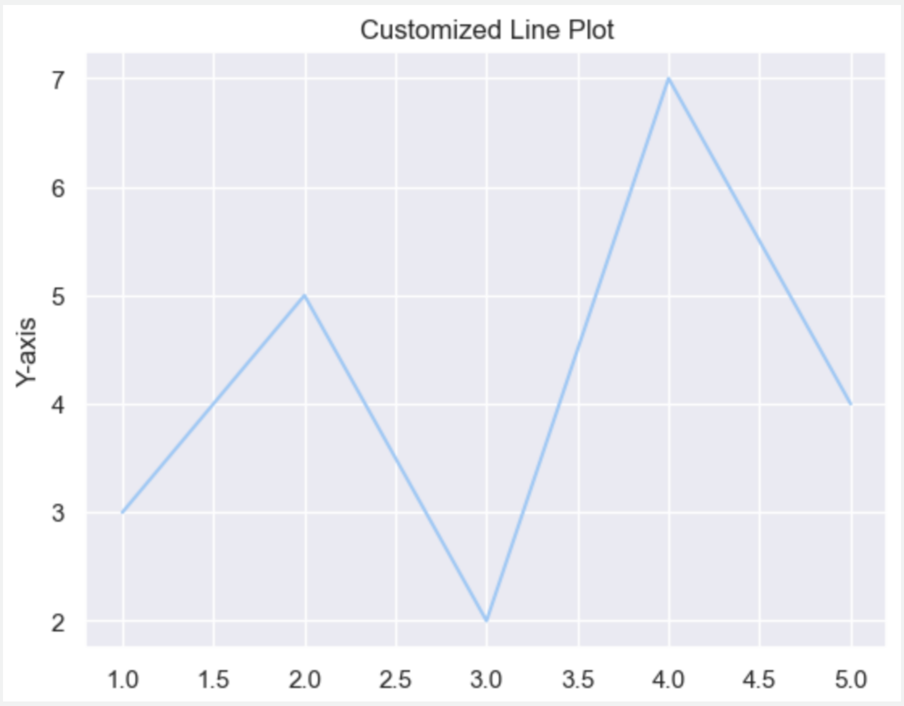
4.1 Setting the Figure Aesthetics

Seaborn allows you to customize the figure's aesthetics, such as style, palette, context, and font scale. This enables you to create visually appealing plots.

EXAMPLE:

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

# Set the style
sns.set_style('darkgrid')

# Sample data
length = [1, 2, 3, 4, 5]
width = [3, 5, 2, 7, 4]

# Create a line plot
sns.lineplot(x=length, y=width)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Customized Line Plot')
plt.show()
```

OUTPUT:



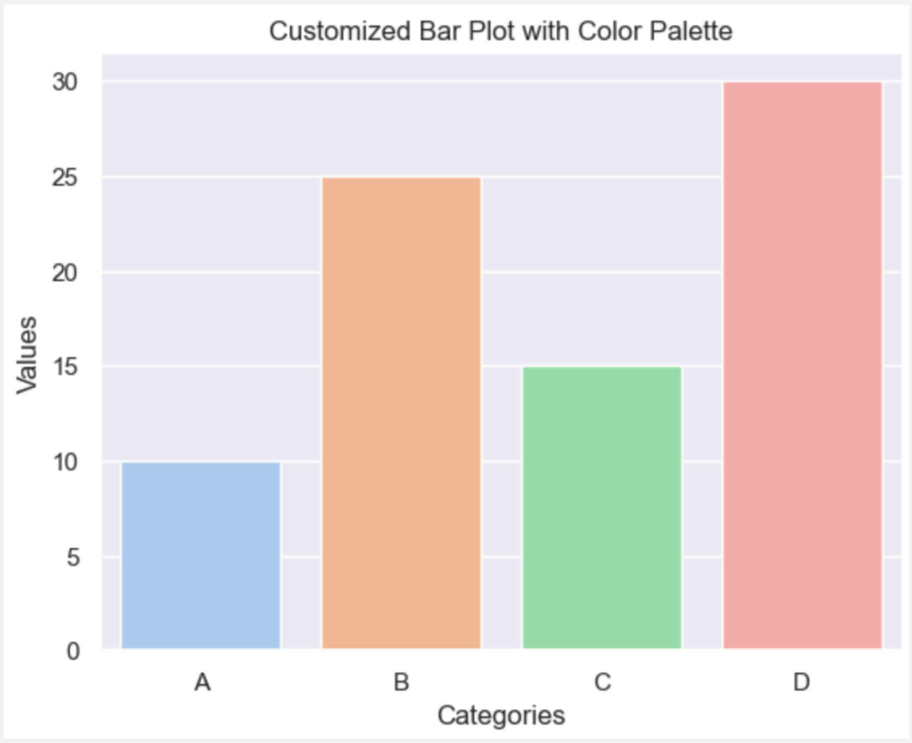
4.2 Color Palettes

Seaborn provides a variety of color palettes that can be used to enhance the visual appeal of plots. You can choose from default palettes or create your own.

EXAMPLE:

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

# Set the color palette
sns.set_palette('pastel')

# Sample data
categories = ['A', 'B', 'C', 'D']
values = [10, 25, 15, 30]

# Create a bar plot
sns.barplot(x=categories, y=values)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Customized Bar Plot with Color Palette')
plt.show()
```

OUTPUT:



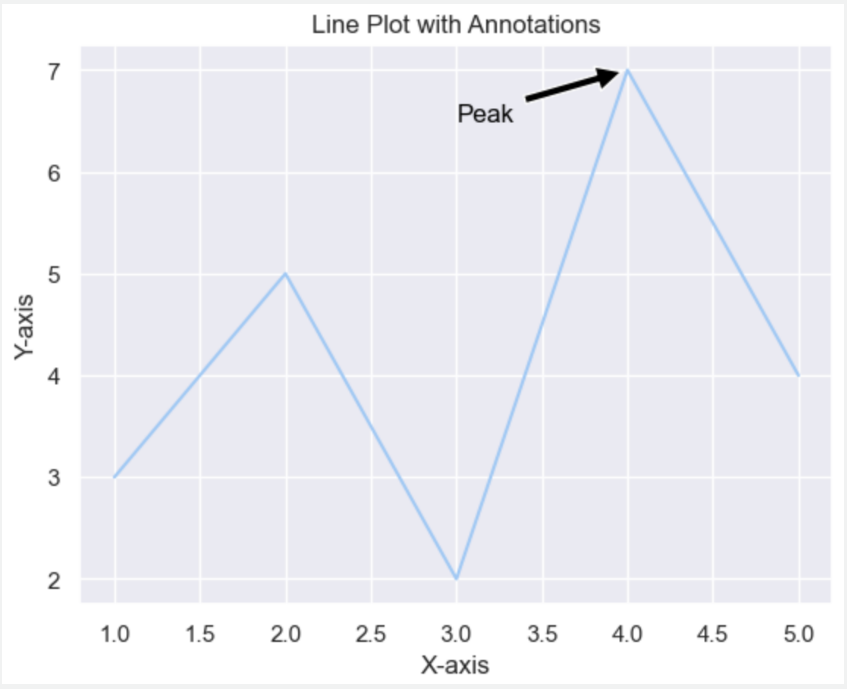
4.3 Adding Annotations

Annotations help in adding additional information to plots, such as labels, text, and arrows. Seaborn allows you to easily include annotations using the **annotate()** function.

EXAMPLE:

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

# Sample data
length = [1, 2, 3, 4, 5]
width = [3, 5, 2, 7, 4]

# Create a line plot
sns.lineplot(x=length, y=width)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot with Annotations')

# Add annotation
plt.annotate('Peak', xy=(4, 7), xytext=(3, 6.5),
             arrowprops=dict(facecolor='black', shrink=0.05),
             fontsize=12, color='black')
plt.show()
```

OUTPUT:



@RAMCHANDRAPADWAL

CHAPTER N.5

Plotting with Categorical Data

A Step-by-Step Guide

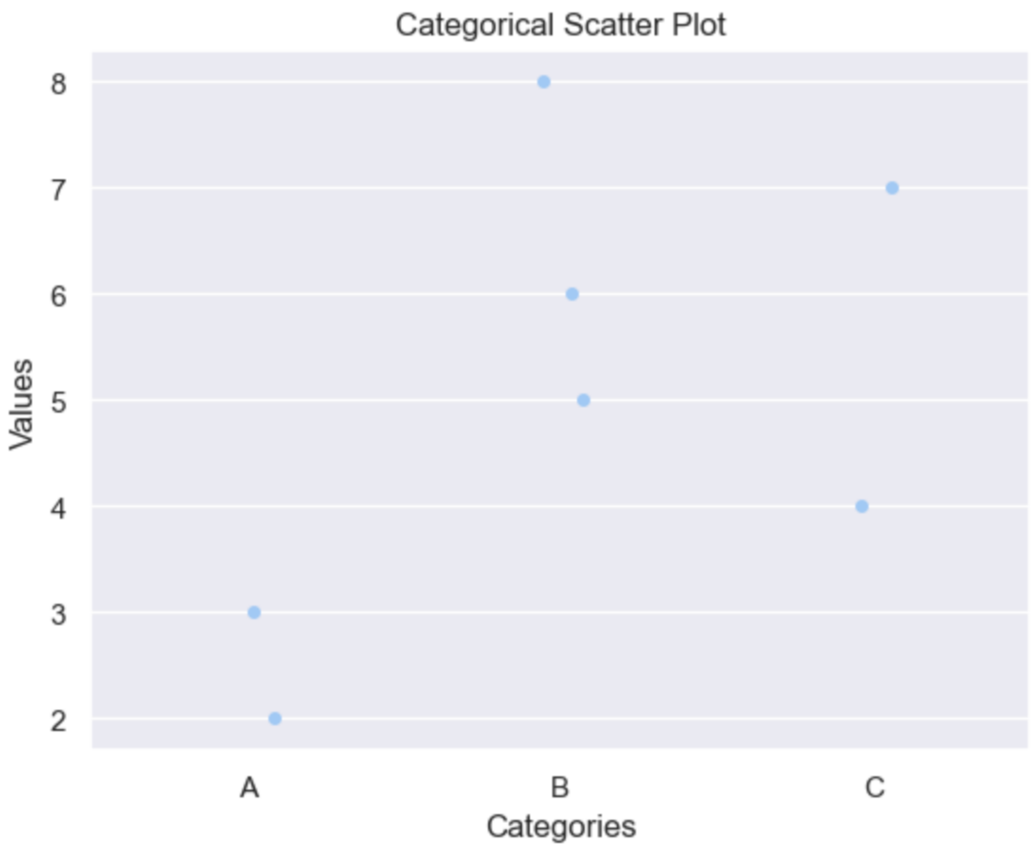
5.1 Categorical Scatter Plots

When dealing with categorical data, Seaborn's **stripplot()** and **swarmplot()** functions are useful for visualizing individual data points along with their distribution.

EXAMPLE:

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

# Sample data
category = ['A', 'B', 'A', 'C', 'C', 'B', 'B']
value = [3, 5, 2, 7, 4, 6, 8]

# Create a categorical scatter plot (strip plot)
sns.stripplot(x=category, y=value)
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Categorical Scatter Plot')
plt.show()
```

OUTPUT:



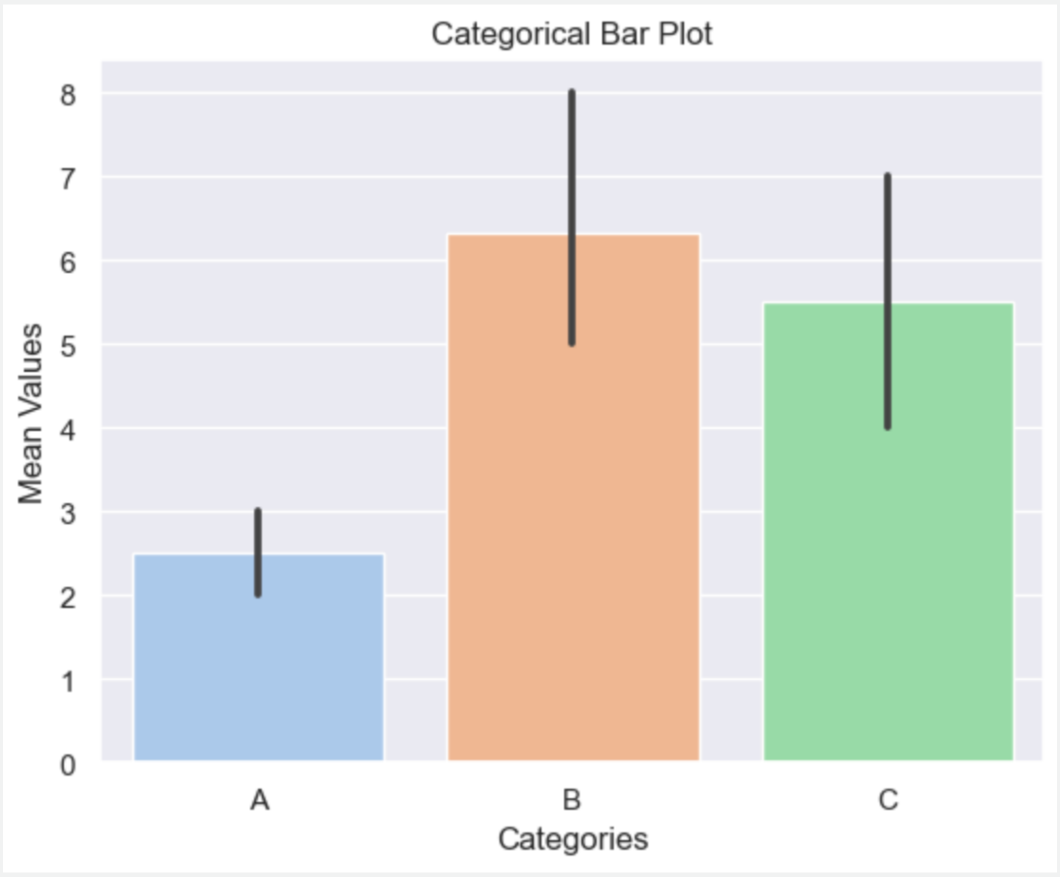
5.2 Categorical Bar Plots

Seaborn's **barplot()** can be used to create bar plots with categorical data, summarizing the central tendency and variability.

EXAMPLE:

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

# Sample data
category = ['A', 'B', 'A', 'C', 'C', 'B', 'B']
value = [3, 5, 2, 7, 4, 6, 8]

# Create a categorical bar plot
sns.barplot(x=category, y=value)
plt.xlabel('Categories')
plt.ylabel('Mean Values')
plt.title('Categorical Bar Plot')
plt.show()
```

OUTPUT:



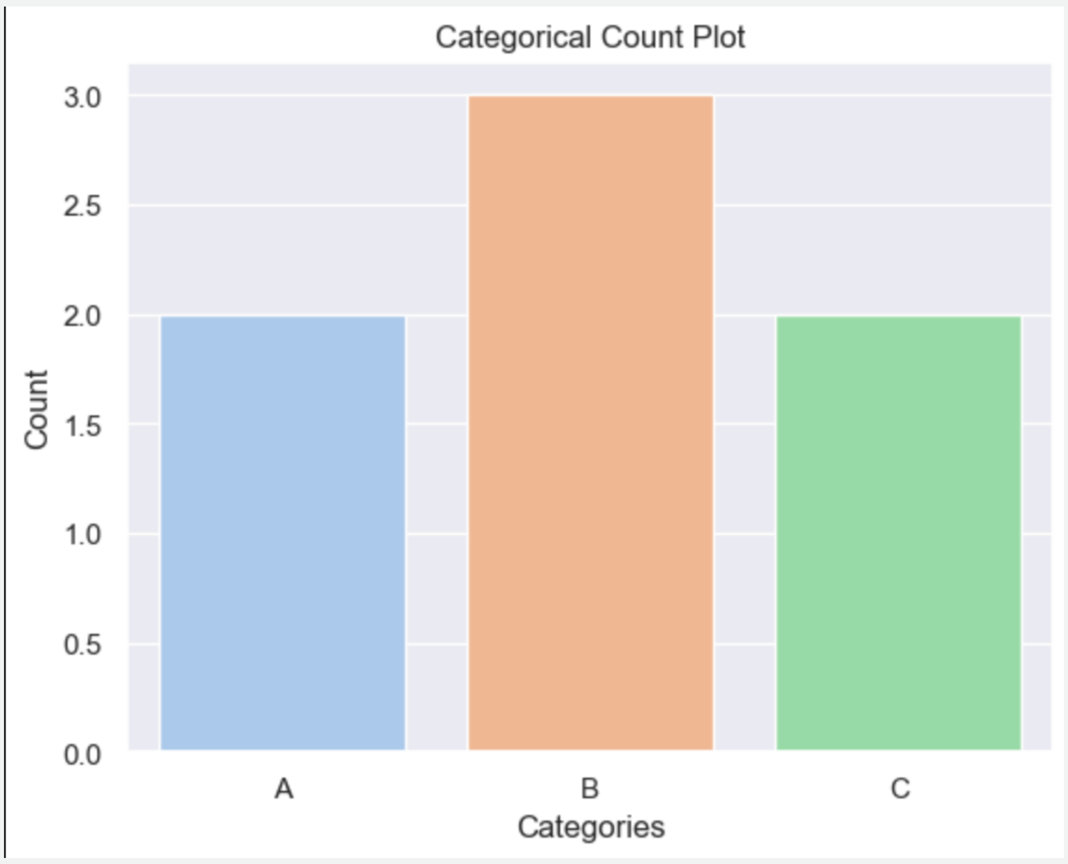
5.3 Categorical Count Plots

To visualize the count of categorical variables, Seaborn's **countplot()** function is useful.

EXAMPLE:

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

# Sample data
category = ['A', 'B', 'A', 'C', 'C', 'B', 'B']

# Create a count plot
sns.countplot(x=category)
plt.xlabel('Categories')
plt.ylabel('Count')
plt.title('Categorical Count Plot')
plt.show()
```

OUTPUT:



CHAPTER N.6

Visualizing Relationships

A Step-by-Step Guide

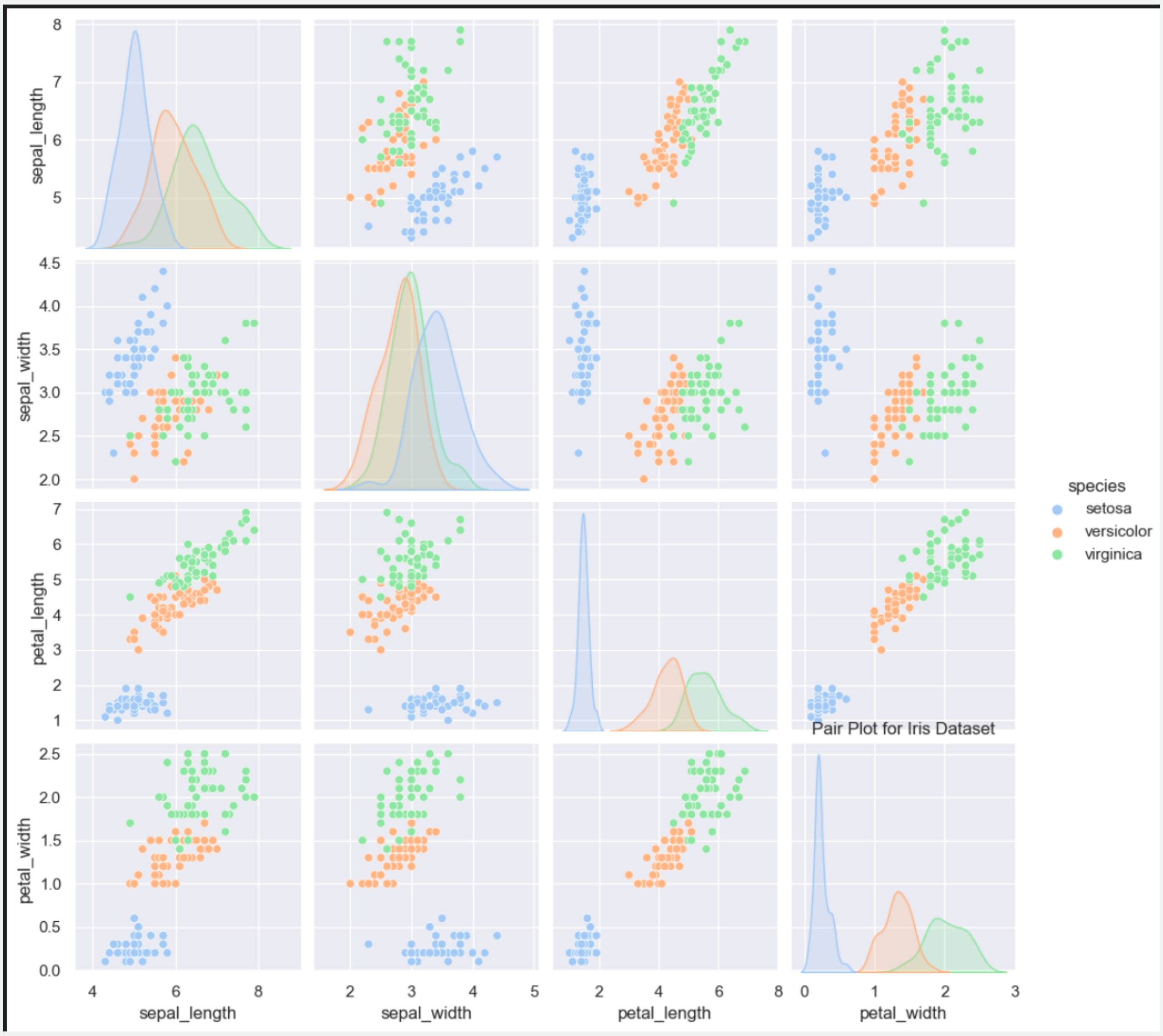
6.1 Pair Plots

Pair plots are used to visualize the relationships between multiple variables. Seaborn's **pairplot()** function creates a grid of scatter plots.

EXAMPLE:

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

# Sample data
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = sns.load_dataset('iris')

# Create a pair plot
sns.pairplot(data, hue='species')
plt.title('Pair Plot for Iris Dataset')
plt.show()
```

OUTPUT:



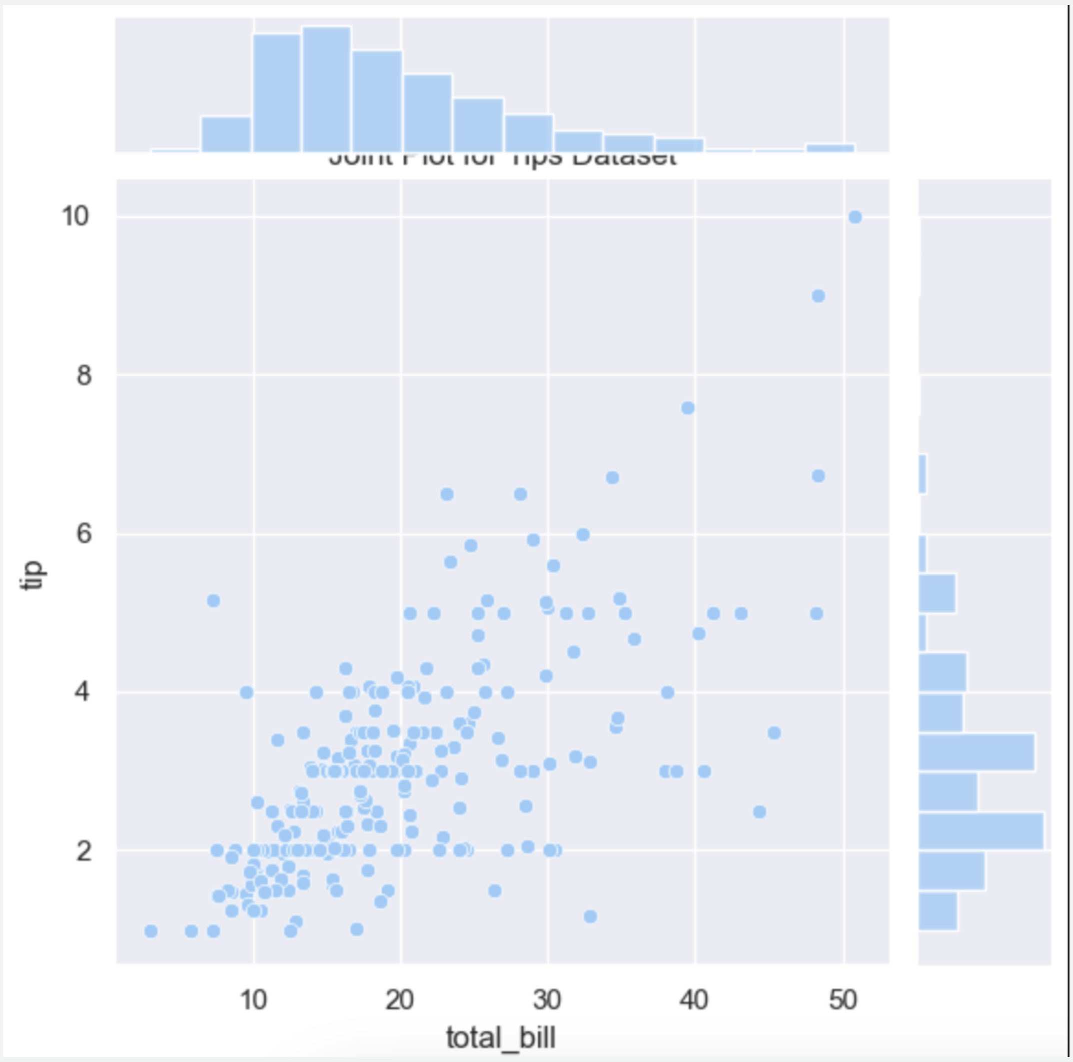
6.2 Joint Plots

Joint plots combine two different plots to visualize the relationship between two variables and their univariate distributions.

EXAMPLE:

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

# Sample data
data = sns.load_dataset('tips')

# Create a joint plot
sns.jointplot(x='total_bill', y='tip', data=data)
plt.title('Joint Plot for Tips Dataset')
plt.show()
```

OUTPUT:



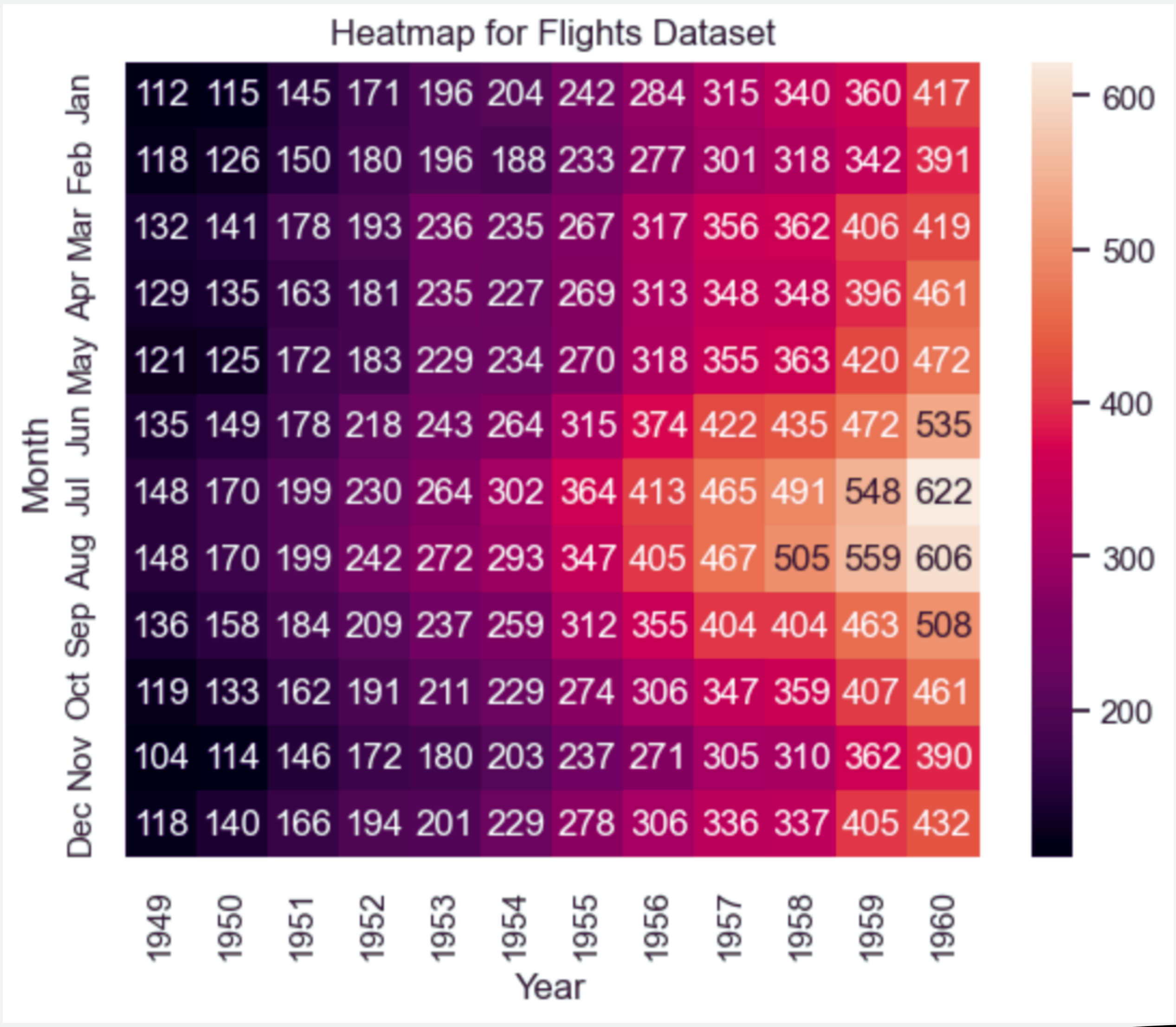
6.3 Heatmaps

Heatmaps are useful for visualizing the correlation between variables in a dataset.

EXAMPLE:

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

# Sample data
data = sns.load_dataset('flights')
data_pivot = data.pivot('month', 'year', 'passengers')

# Create a heatmap
sns.heatmap(data_pivot, annot=True, fmt='d')
plt.xlabel('Year')
plt.ylabel('Month')
plt.title('Heatmap for Flights Dataset')
plt.show()
```

OUTPUT:



@RAMCHANDRAPADWAL

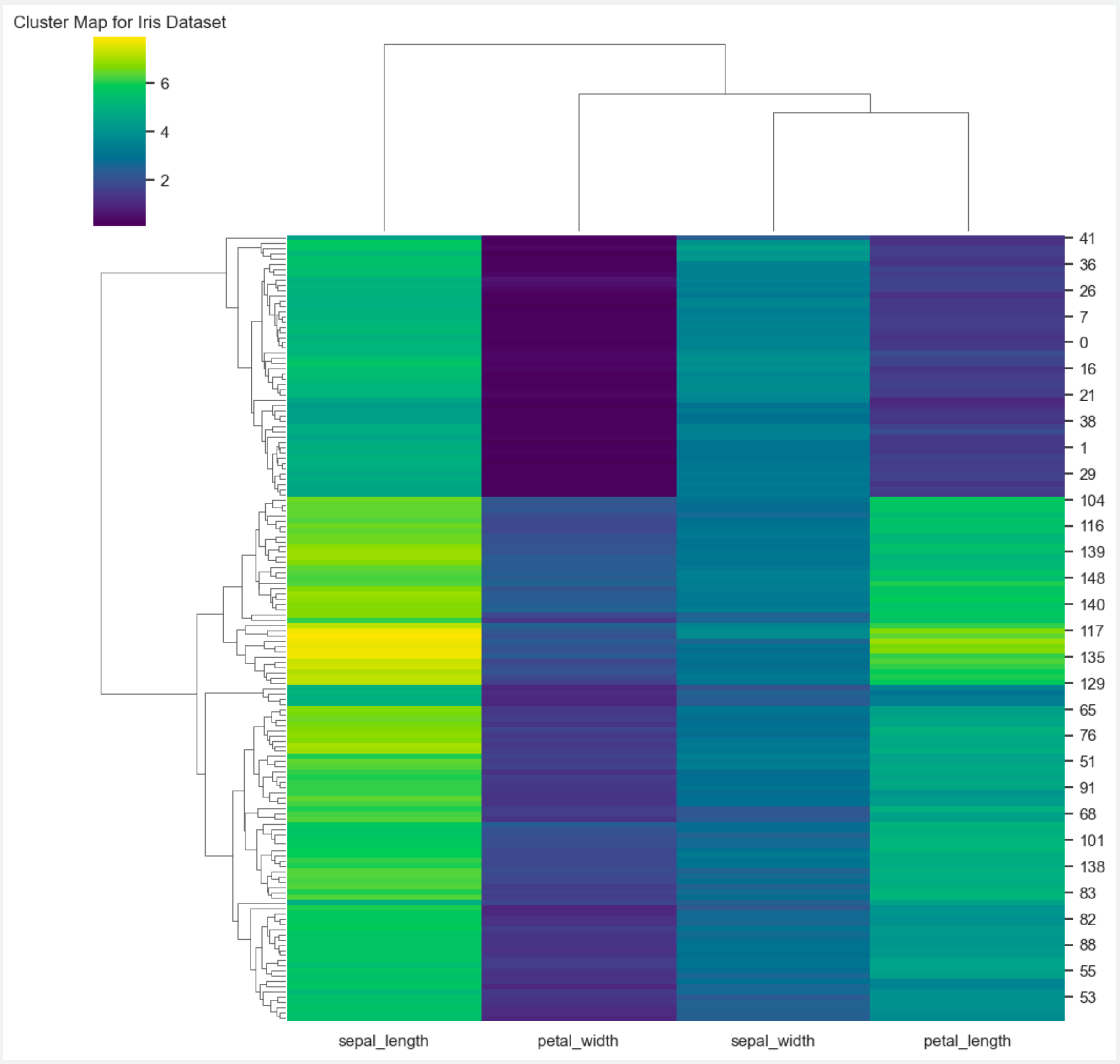
6.4 Cluster Maps

Cluster maps are used to group similar data together based on their correlation.

EXAMPLE:

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

# Sample data
data = sns.load_dataset('iris')
data = data.drop(columns='species')

# Create a cluster map
sns.clustermap(data, cmap='viridis')
plt.title('Cluster Map for Iris Dataset')
plt.show()
```

OUTPUT:



@RAMCHANDRAPADWAL

CHAPTER N.7

Time Series Visualization

A Step-by-Step Guide

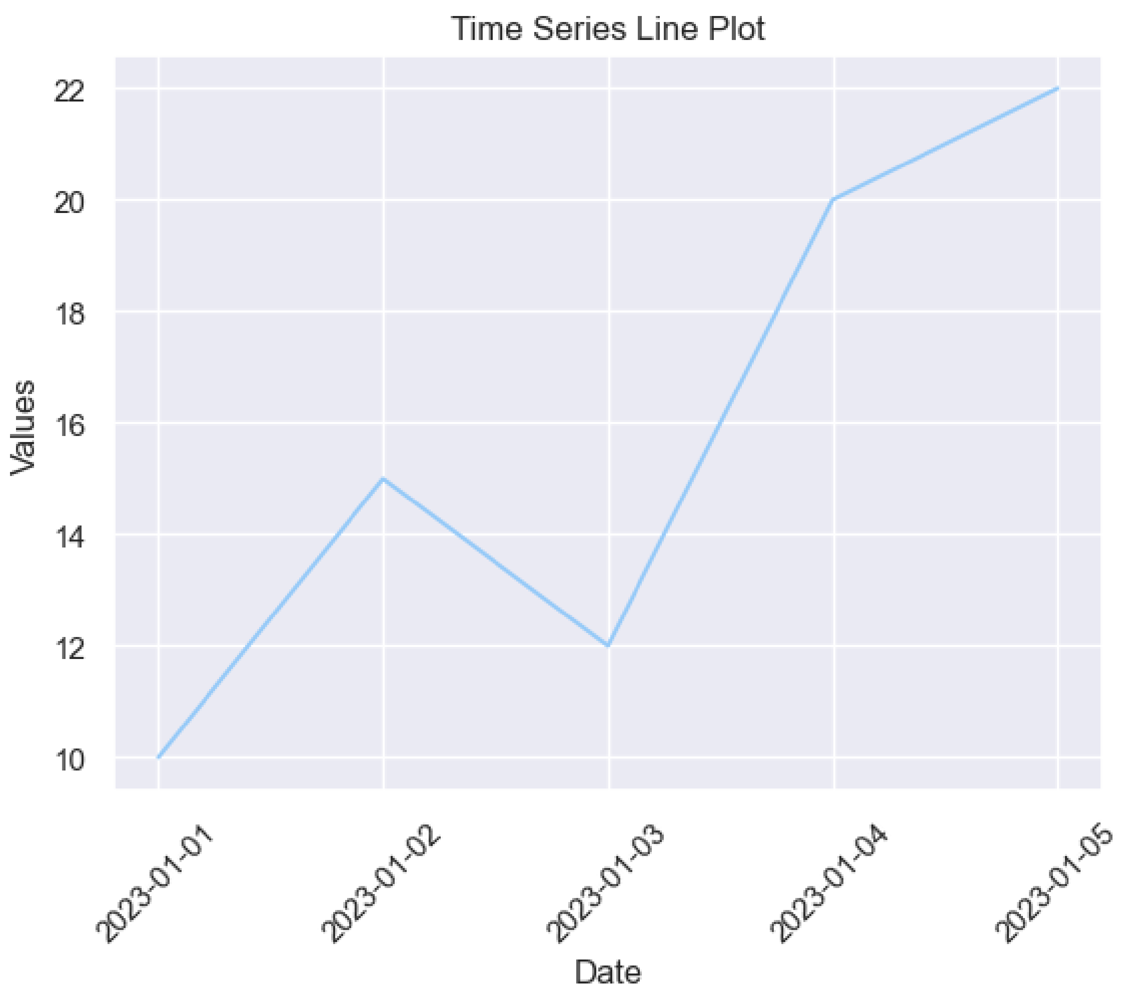
7.1 Line Plots with Time Series Data

Time series data can be visualized using line plots to observe trends and patterns over time.

EXAMPLE:

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

# Sample data
import pandas as pd

dates = ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05']
values = [10, 15, 12, 20, 22]

# Create a time series line plot
sns.lineplot(x=dates, y=values)
plt.xlabel('Date')
plt.ylabel('Values')
plt.title('Time Series Line Plot')
plt.xticks(rotation=45)
plt.show()
```

OUTPUT:



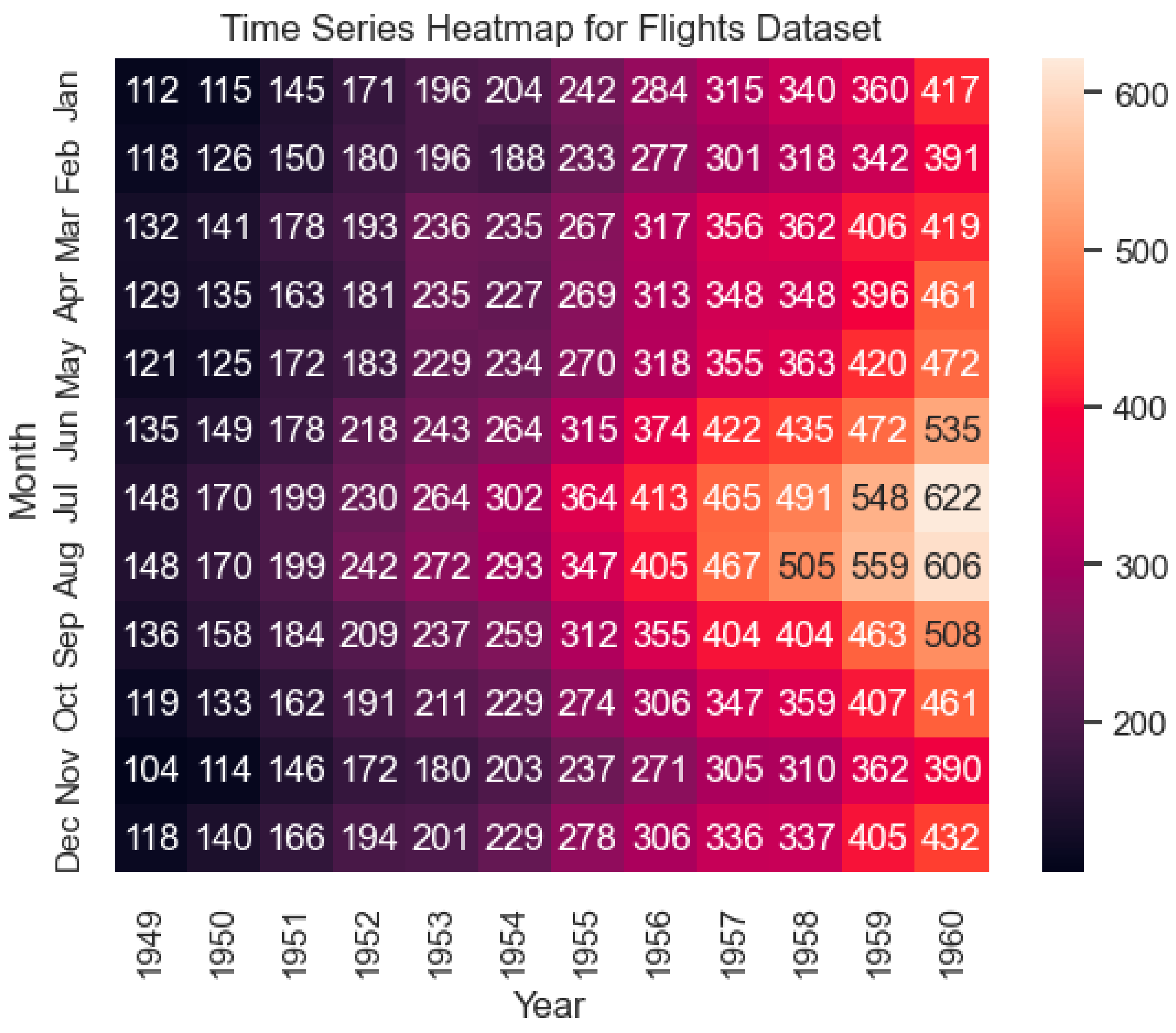
7.2 Time Series Heatmaps

Time series heatmaps provide an overview of data variations across both time and another variable.

EXAMPLE:

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

# Sample data
data = sns.load_dataset('flights')
data_pivot = data.pivot('month', 'year', 'passengers')

# Create a time series heatmap
sns.heatmap(data_pivot, annot=True, fmt='d')
plt.xlabel('Year')
plt.ylabel('Month')
plt.title('Time Series Heatmap for Flights Dataset')
plt.show()
```

OUTPUT:



	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

CHAPTER N.8

Additional Tips and Tricks

A Step-by-Step Guide

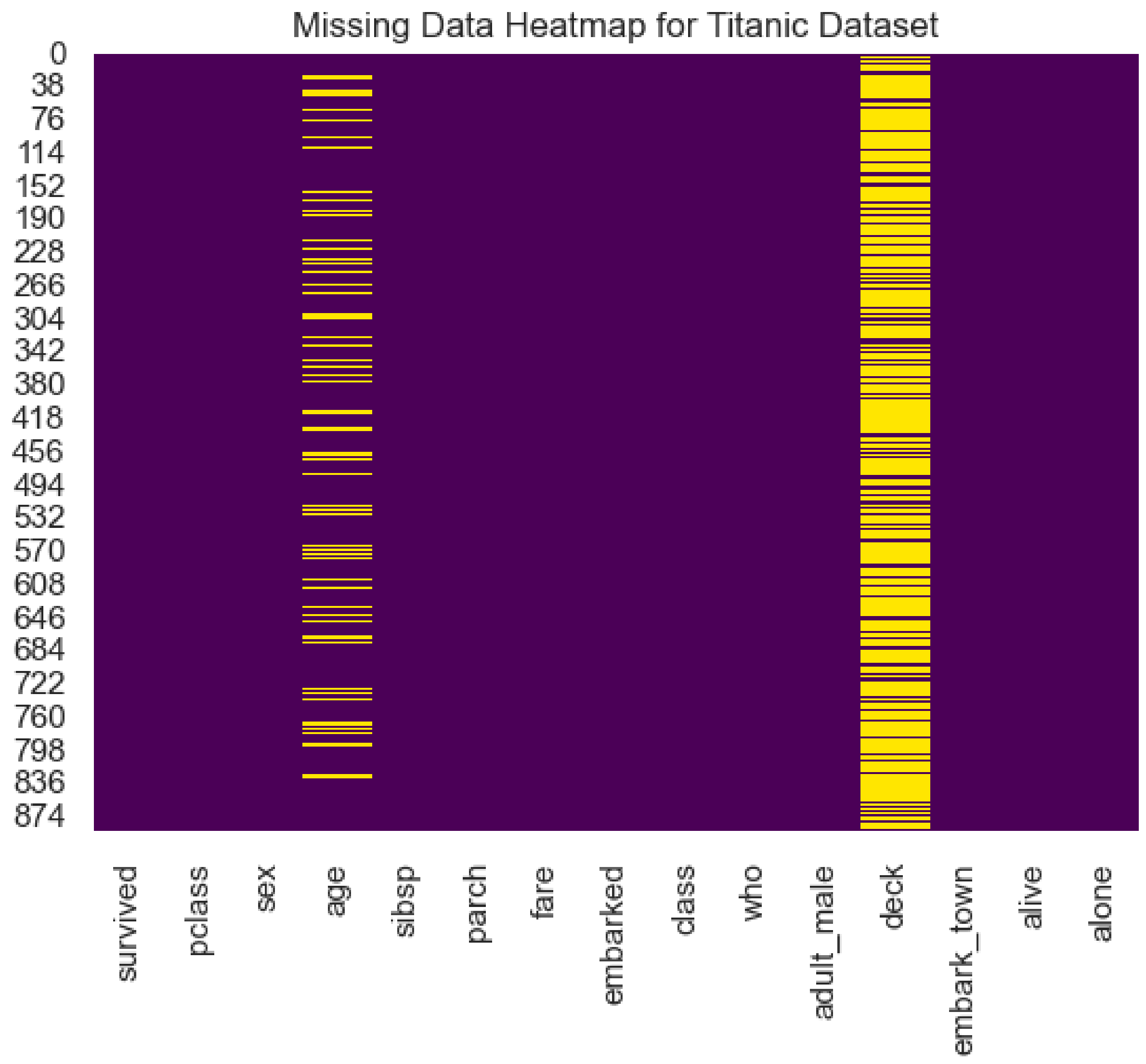
8.1 Dealing with Missing Data

Seaborn provides options to handle missing data while visualizing the dataset.

EXAMPLE:

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

# Sample data with missing values
data = sns.load_dataset('titanic')

# Create a heatmap to visualize missing data
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Heatmap for Titanic Dataset')
plt.show()
```

OUTPUT:



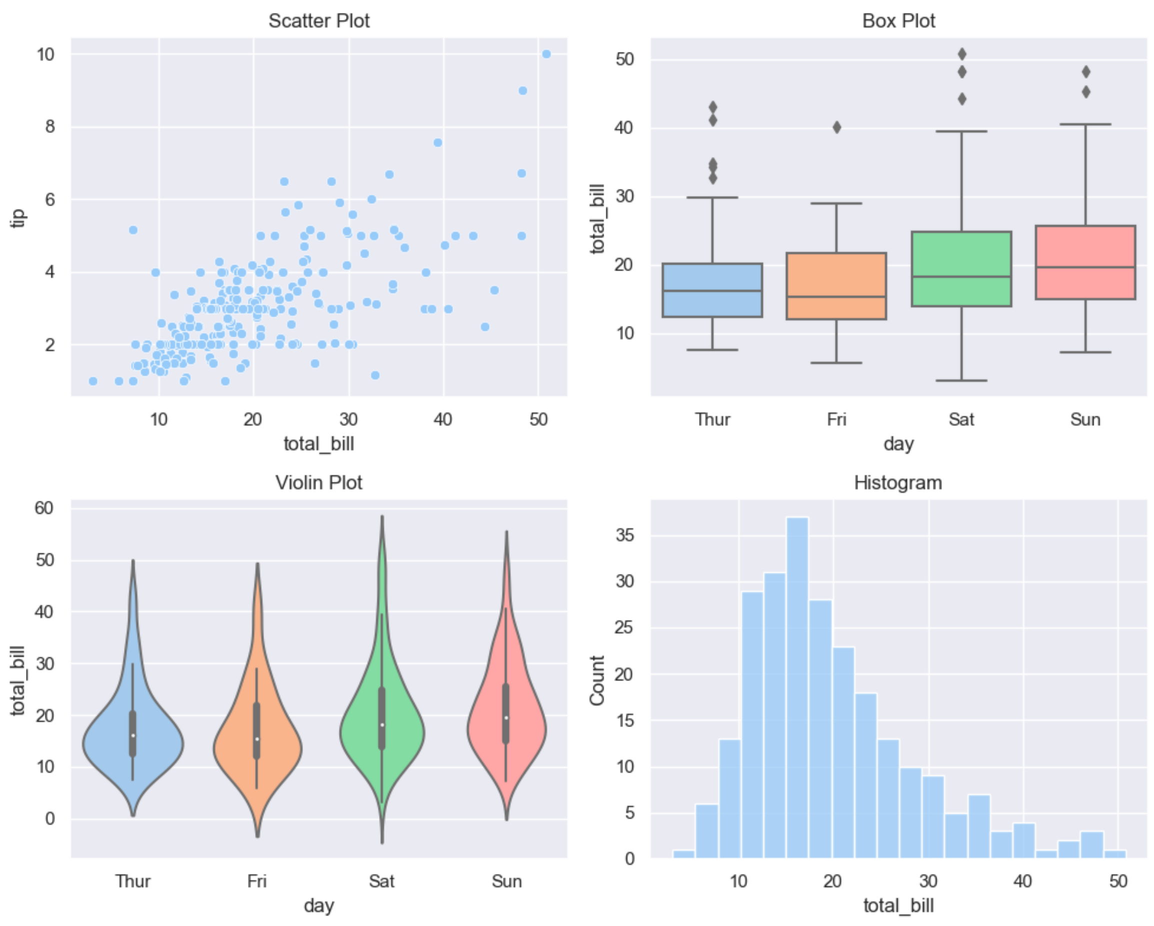
8.2 Working with Subplots

Seaborn can be used in combination with Matplotlib to create subplots for visualizing multiple plots simultaneously.

EXAMPLE:

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

# Sample data
data = sns.load_dataset('tips')

# Create subplots
fig, axes = plt.subplots(2, 2, figsize=(10, 8))

# Subplot 1: Scatter plot
sns.scatterplot(x='total_bill', y='tip', data=data, ax=axes[0, 0])
axes[0, 0].set_title('Scatter Plot')

# Subplot 2: Box plot
sns.boxplot(x='day', y='total_bill', data=data, ax=axes[0, 1])
axes[0, 1].set_title('Box Plot')

# Subplot 3: Violin plot
sns.violinplot(x='day', y='total_bill', data=data, ax=axes[1, 0])
axes[1, 0].set_title('Violin Plot')

# Subplot 4: Histogram
sns.histplot(data['total_bill'], ax=axes[1, 1], bins=20)
axes[1, 1].set_title('Histogram')

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

OUTPUT:



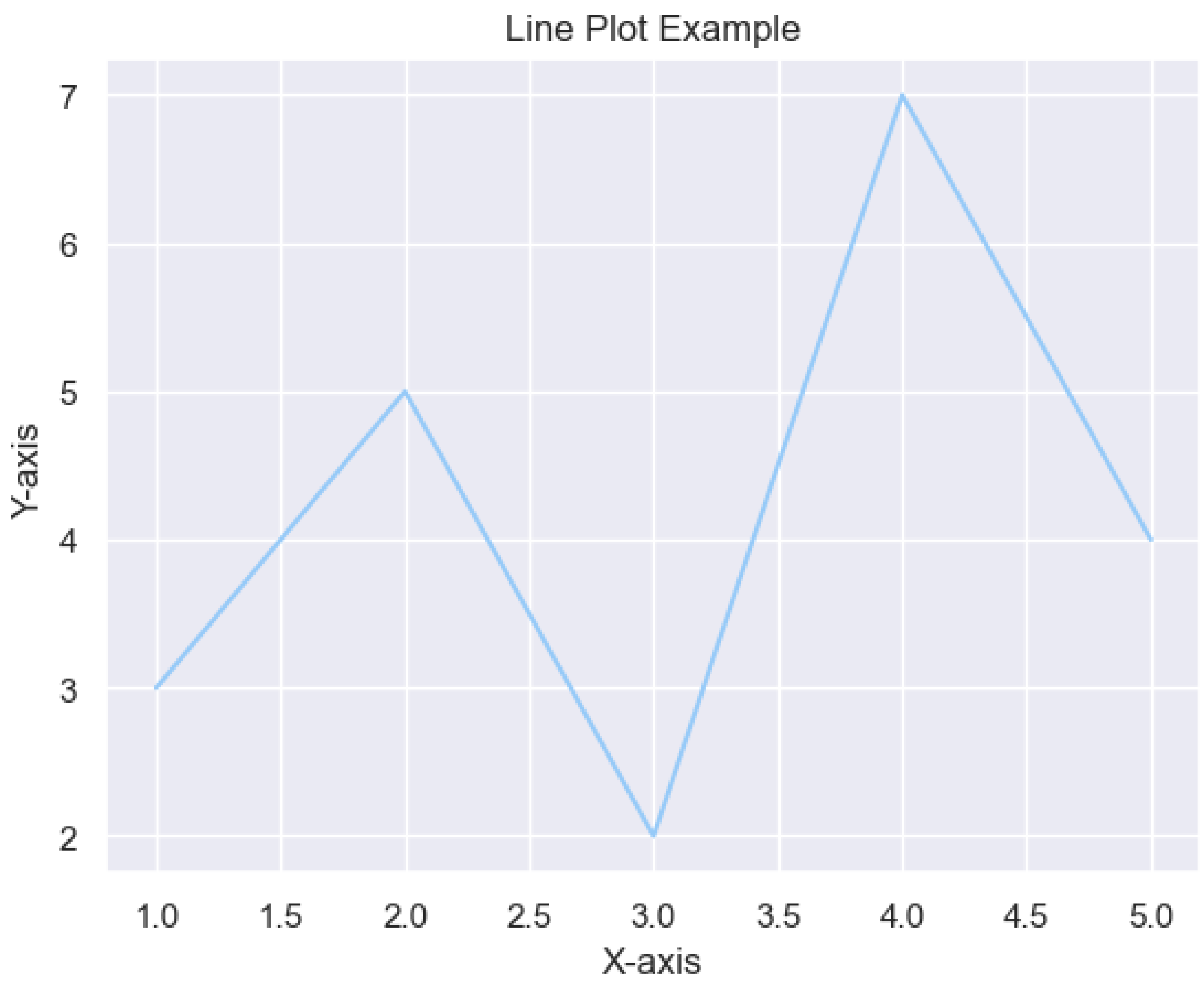
8.3 Saving and Exporting Plots

You can save Seaborn plots as image files for further use or sharing.

EXAMPLE:

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

# Sample data
length = [1, 2, 3, 4, 5]
width = [3, 5, 2, 7, 4]

# Create a line plot
sns.lineplot(x=length, y=width)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot Example')
plt.savefig('line_plot.png')
plt.show()
```

OUTPUT:



@RAMCHANDRAPADWAL

CHAPTER N.9

Real-World Data Analysis Example

A Step-by-Step Guide

9.1 Data Preparation

Let's work with a real-world dataset to demonstrate Seaborn's capabilities for data analysis. We'll use the famous "Titanic" dataset to visualize and explore various aspects of the passengers on the Titanic.

EXAMPLE:

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

# Load the Titanic dataset
data = sns.load_dataset('titanic')

# Display the first few rows of the dataset
print(data.head())
```

OUTPUT

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

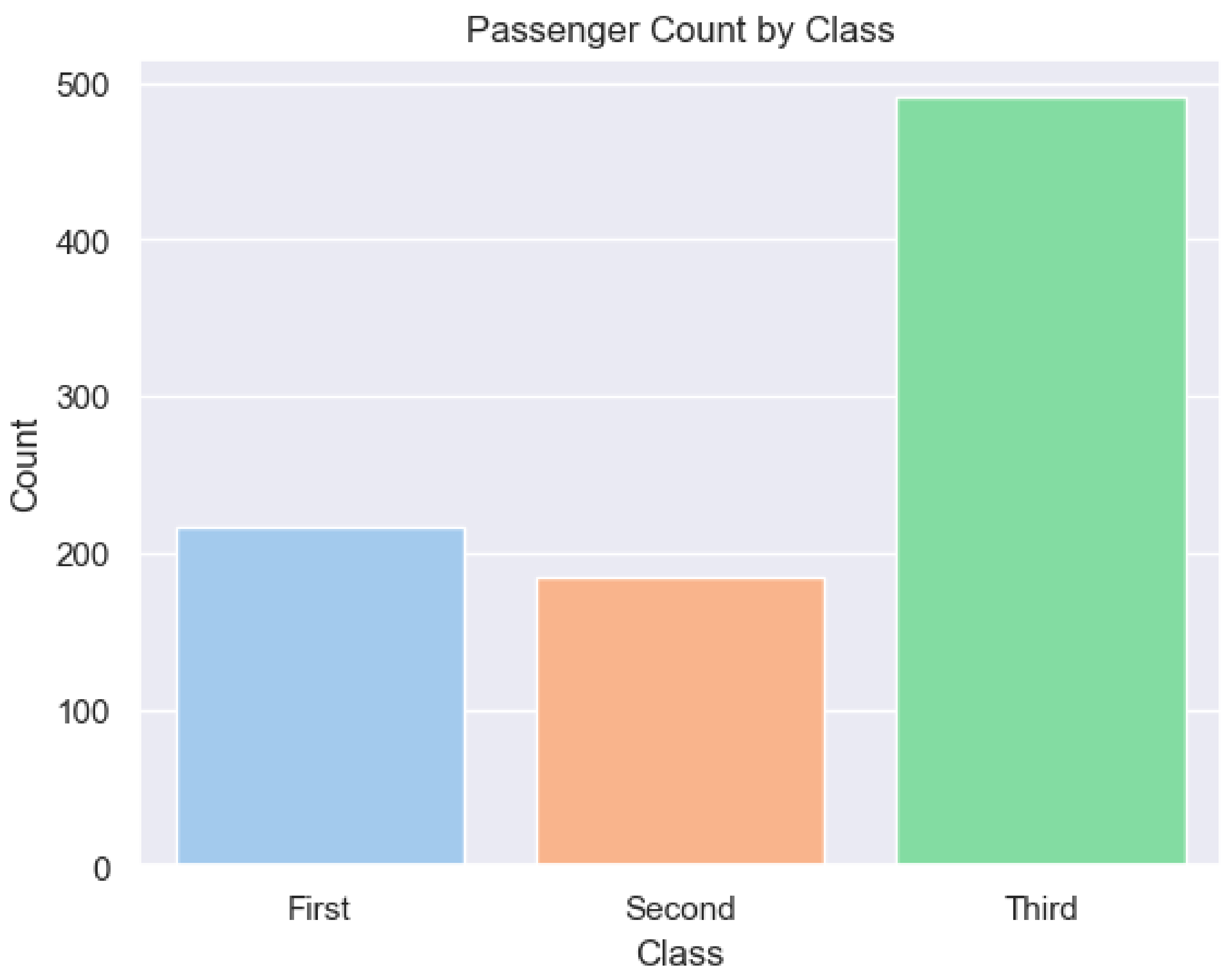
9.2 Exploratory Data Analysis

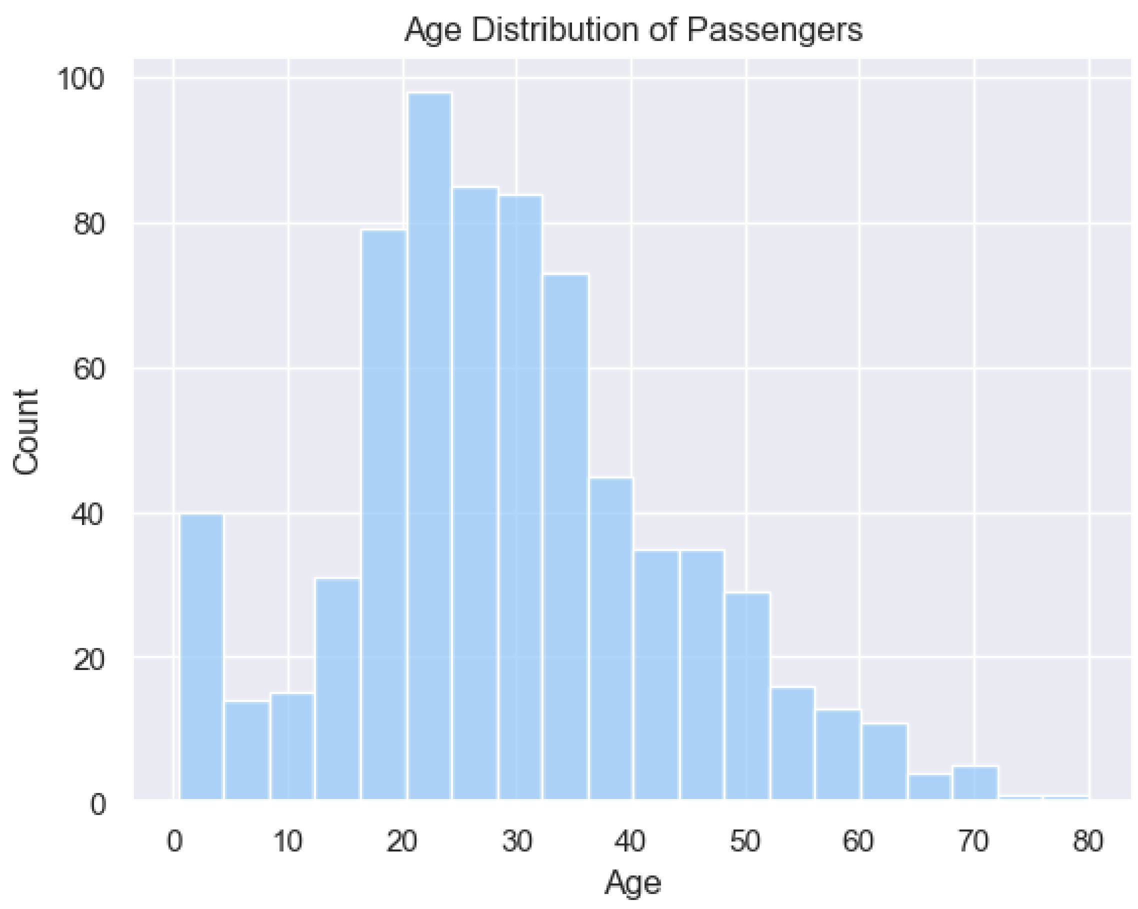
We'll begin by exploring the basic statistics and distributions of the data using Seaborn's visualization functions.

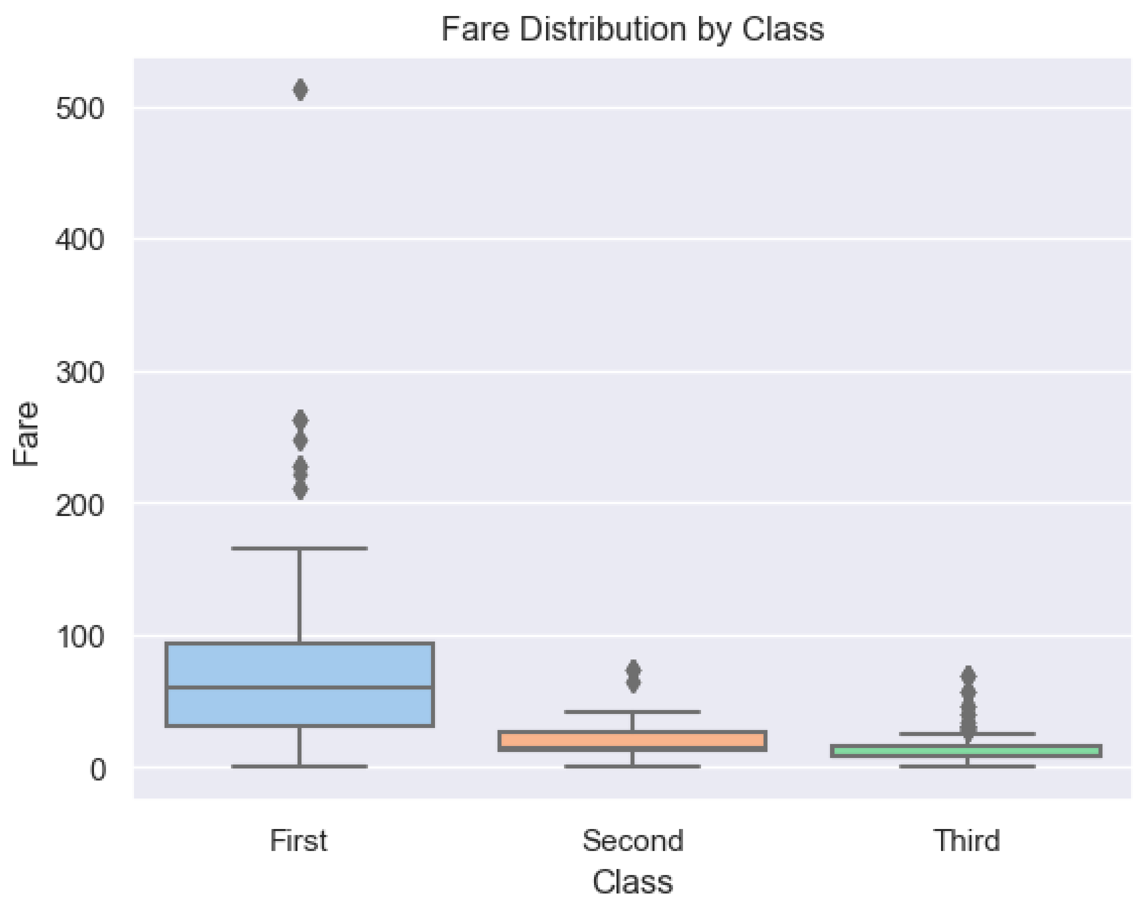
EXAMPLE:

```
# Count plot for the number of passengers in each class
sns.countplot(x='class', data=data)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Passenger Count by Class')
plt.show()

# Histogram for the age distribution of passengers
sns.histplot(data['age'].dropna(), bins=20)
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution of Passengers')
plt.show()

# Box plot for fare distribution by class
sns.boxplot(x='class', y='fare', data=data)
plt.xlabel('Class')
plt.ylabel('Fare')
plt.title('Fare Distribution by Class')
plt.show()
```

OUTPUT







@RAMCHANDRAPADWAL

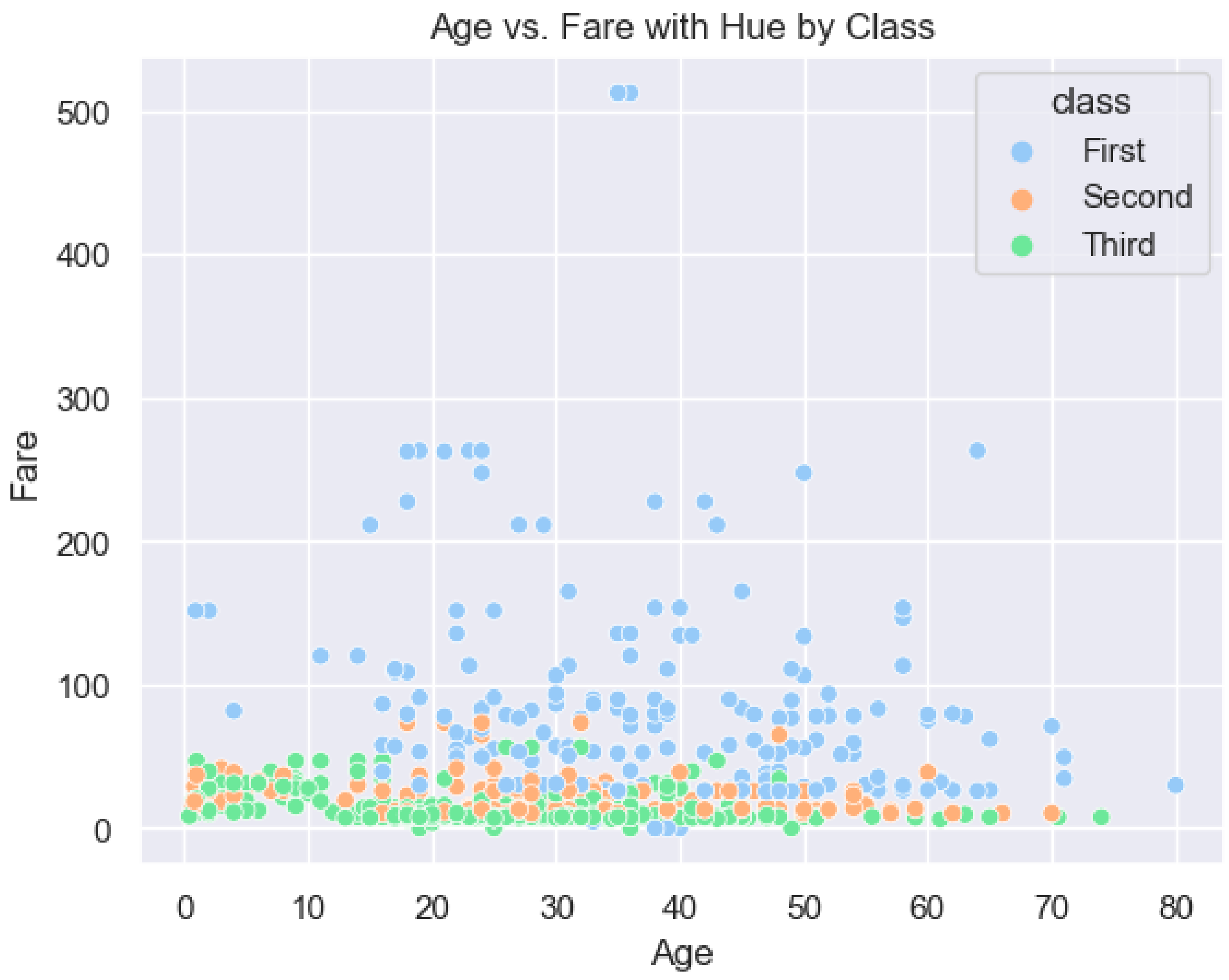
9.3 Advanced Visualizations

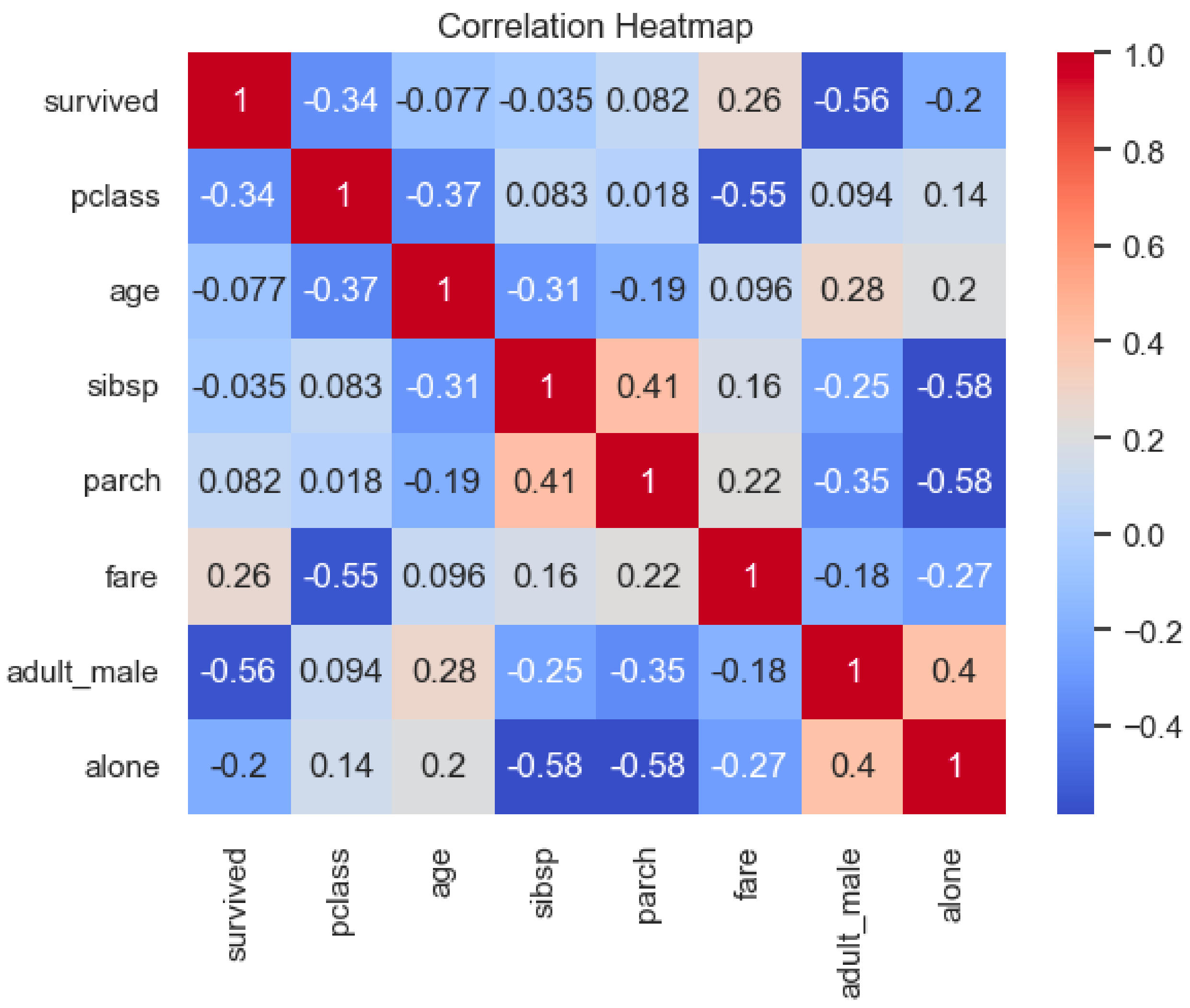
Next, we can create more advanced visualizations to understand relationships between variables.

EXAMPLE:

```python
# Scatter plot for age vs. fare with hue by class
sns.scatterplot(x='age', y='fare', data=data, hue='class')
plt.xlabel('Age')
plt.ylabel('Fare')
plt.title('Age vs. Fare with Hue by Class')
plt.show()

# Heatmap to visualize correlations between numeric variables
corr_matrix = data.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

OUTPUT




Correlation Heatmap
survived 1 -0.34 -0.077 -0.035 0.082 0.26 -0.56 -0.2
pclass -0.34 1 -0.37 0.083 0.018 -0.55 0.094 0.14
age -0.077 -0.37 1 -0.31 -0.19 0.096 0.28 0.2
sibsp -0.035 0.083 -0.31 1 0.41 0.16 -0.25 -0.58
parch 0.082 0.018 -0.19 0.41 1 0.22 -0.35 -0.58
fare 0.26 -0.55 0.096 0.16 0.22 1 -0.18 -0.27
adult_male -0.56 0.094 0.28 -0.25 -0.35 -0.18 1 0.4
alone -0.2 0.14 0.2 -0.58 -0.58 -0.27 0.4 1
survived pclass age sibsp parch fare adult_male alone
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

Conclusion

Seaborn is a versatile and powerful data visualization library that simplifies the creation of informative and aesthetically pleasing plots. This practical guide has covered various aspects of Seaborn, including basic plots, customizations, plotting with categorical data, visualizing relationships, time series visualization, and additional tips and tricks. By mastering Seaborn, you can effectively explore and communicate insights from your data in data science and machine learning projects.