Python Seaborn

Visualization is a crucial aspect of data analysis and interpretation, as it allows for easy comprehension of complex data sets.

It helps in identifying patterns, relationships, and trends

Visualization libraries in Python enable users to create intuitive and interactive data visualizations that can effectively communicate insights to a broad audience.

Some of the popular visualization libraries and frameworks in Python include Matplotlib, Plotly, Bokeh, and Seaborn.

Seaborn vs. Matplotlib

One of the main differences between Matplotlib and Seaborn is their focus. Matplotlib is a low-level plotting library that provides a wide range of tools for creating highly customizable visualizations. It is a highly flexible library, allowing users to create almost any type of plot they can imagine.

Seaborn, on the other hand, is a high-level interface for creating statistical graphics. It is built on top of Matplotlib and provides a simpler, more intuitive interface for creating common statistical plots. Seaborn is designed to work with Pandas dataframes, making it easy to create visualizations with minimal code.

Another key difference between Matplotlib and Seaborn is their default styles and color palettes. Matplotlib provides a limited set of default styles and color palettes, requiring users to customize their plots manually to achieve a desired look.

Seaborn, on the other hand, offers a range of default styles and color palettes that are optimized for different types of data and visualizations. This makes it easy for users to create visually appealing plots with minimal customization.

Installing Seaborn

```
# install seaborn with pip

pip install seaborn
```

Seaborn Plot types

Seaborn provides a wide range of plot types that can be used for data visualization and exploratory data analysis. Broadly speaking, any visualization can fall into one of the three categories.

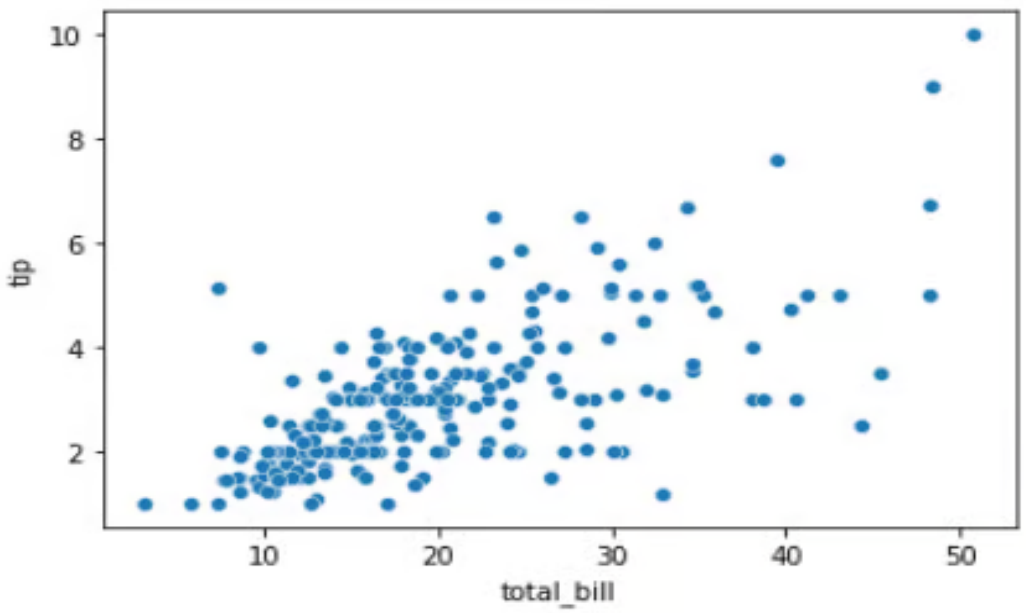
- Univariate – x only (contains only one axis of information)
- Bivariate – x and y (contains two axis of information)
- Trivariate – x, y, z (contains three axis of information)

Seaborn scatter plots

Scatter plots are used to visualize the relationship between two continuous variables. Each point on the plot represents a single data point, and the position of the point on the x and y-axis represents the values of the two variables.

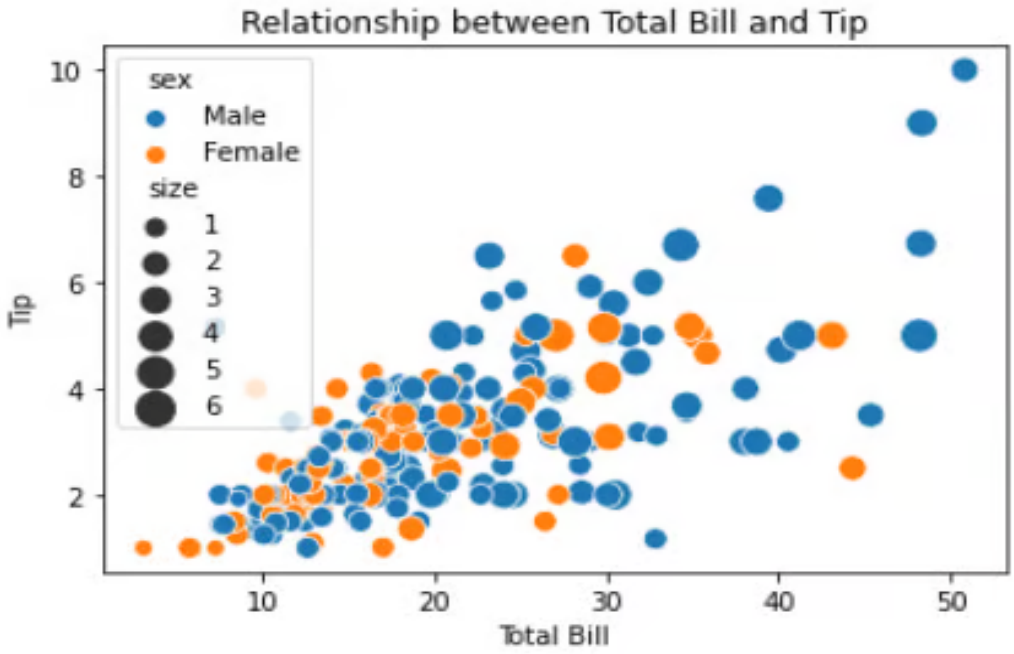
The plot can be customized with different colors and markers to help distinguish different groups of data points. In Seaborn, scatter plots can be created using the `scatterplot()` function.

```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.scatterplot(x="total_bill", y="tip", data=tips)
```



This simple plot can be improved by customizing the `hue` and `size` parameters of the plot. Here’s how:

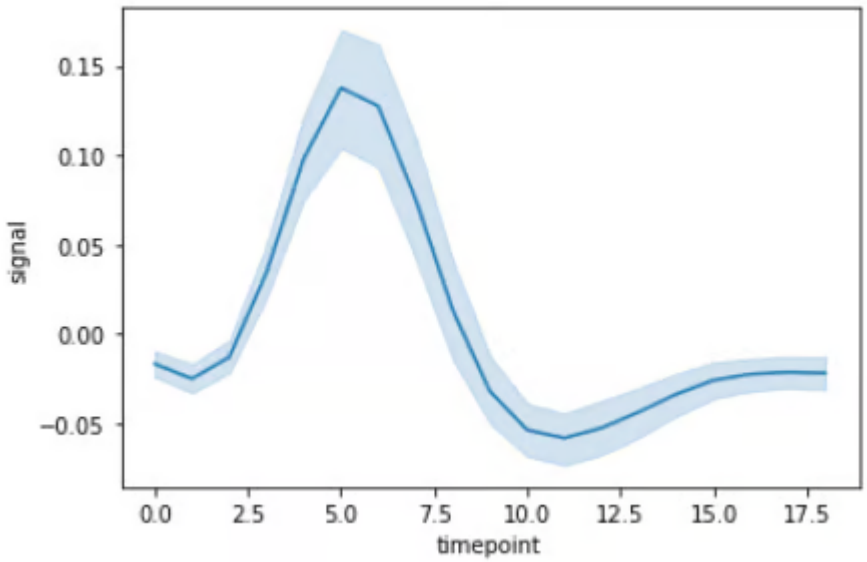
```
import seaborn as sns

import matplotlib.pyplot as plt



tips = sns.load_dataset("tips")

# customize the scatter plot

sns.scatterplot(x="total_bill", y="tip", hue="sex", size="size",
sizes=(50, 200), data=tips)

# add labels and title

plt.xlabel("Total Bill")

plt.ylabel("Tip")

plt.title("Relationship between Total Bill and Tip")



# display the plot

plt.show()
```



Seaborn line plots

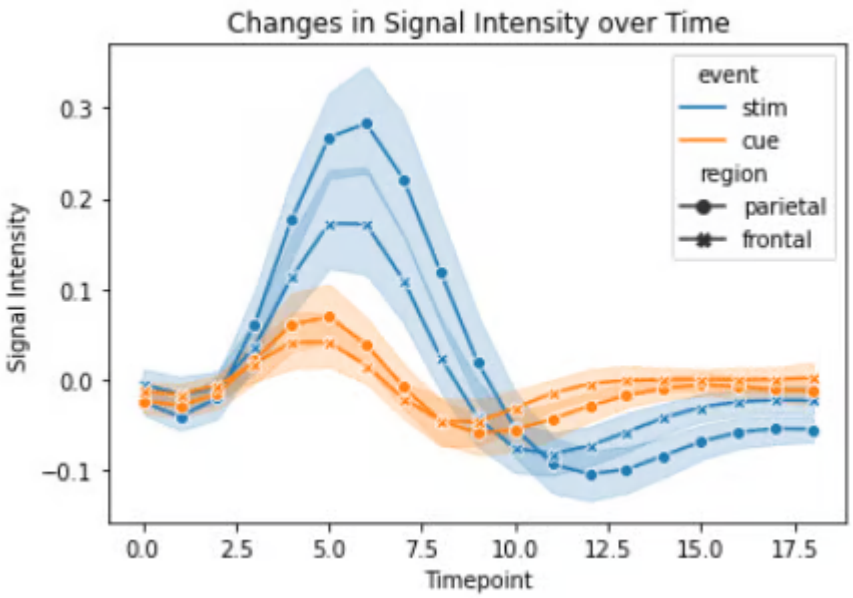
Line plots are used to visualize trends in data over time or other continuous variables. In a line plot, each data point is connected by a line, creating a smooth curve. In Seaborn, line plots can be created using the lineplot() function.

```
import seaborn as sns

fmri = sns.load_dataset("fmri")

sns.lineplot(x="timepoint", y="signal", data=fmri)
```



We can very easily customize this by using `event` and `region` columns from the dataset.

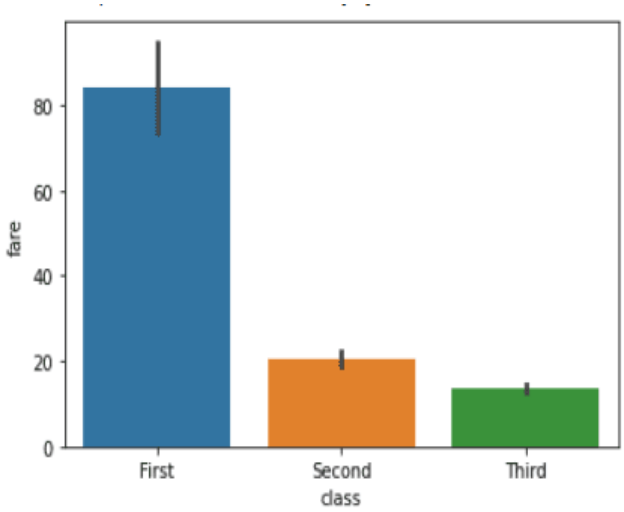
```
import seaborn as sns

import matplotlib.pyplot as plt

fmri = sns.load_dataset("fmri")

# customize the line plot



sns.lineplot(x="timepoint", y="signal", hue="event", style="region",
markers=True, dashes=False, data=fmri)

# add labels and title

plt.xlabel("Timepoint")

plt.ylabel("Signal Intensity")

plt.title("Changes in Signal Intensity over Time")

# display the plot

plt.show()
```



Seaborn bar plots

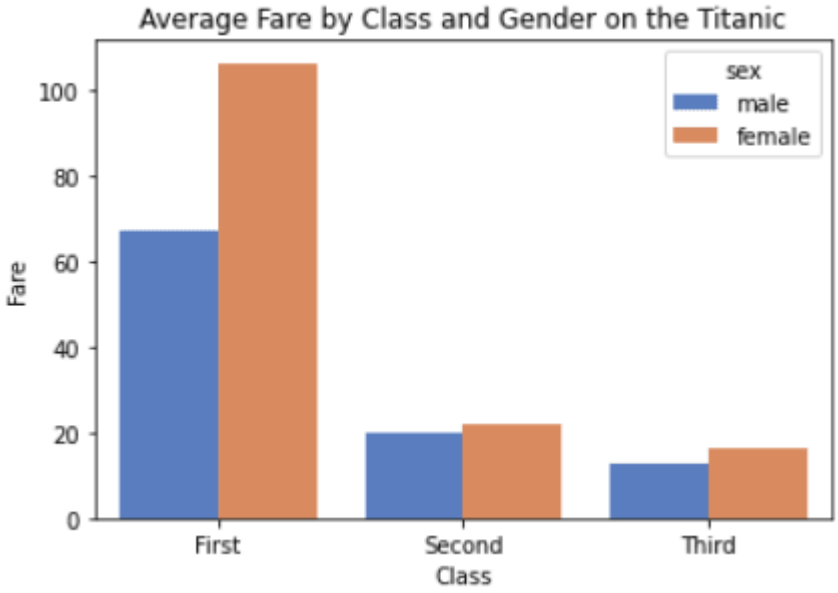
Bar plots are used to visualize the relationship between a categorical variable and a continuous variable. In a bar plot, each bar represents the mean or median (or any aggregation) of the continuous variable for each category. In Seaborn, bar plots can be created using the barplot() function.

```
import seaborn as sns

titanic = sns.load_dataset("titanic")

sns.barplot(x="class", y="fare", data=titanic)
```



Let’s customize this plot by including `sex` column from the dataset.

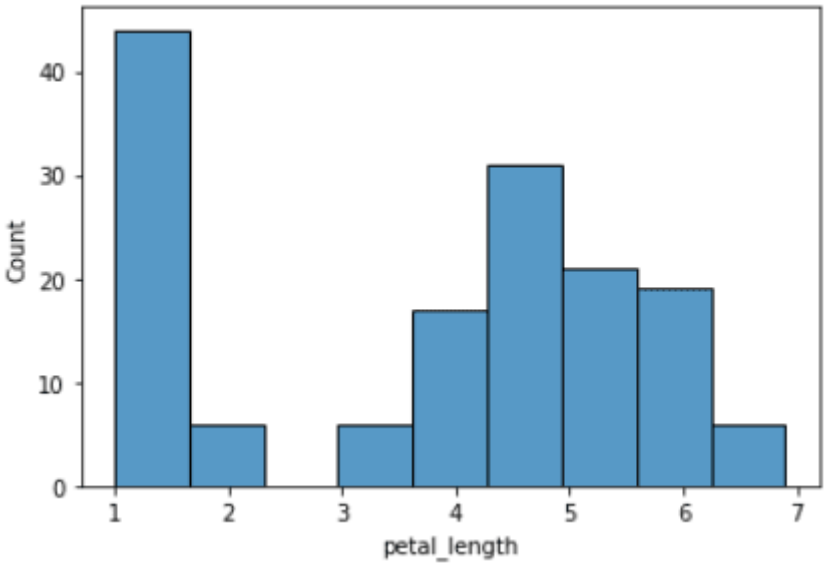
```
import seaborn as sns



import matplotlib.pyplot as plt

titanic = sns.load_dataset("titanic")

# customize the bar plot

sns.barplot(x="class", y="fare", hue="sex", ci=None,
palette="muted", data=titanic)

# add labels and title

plt.xlabel("Class")

plt.ylabel("Fare")

plt.title("Average Fare by Class and Gender on the Titanic")

# display the plot

plt.show()
```



Seaborn histograms

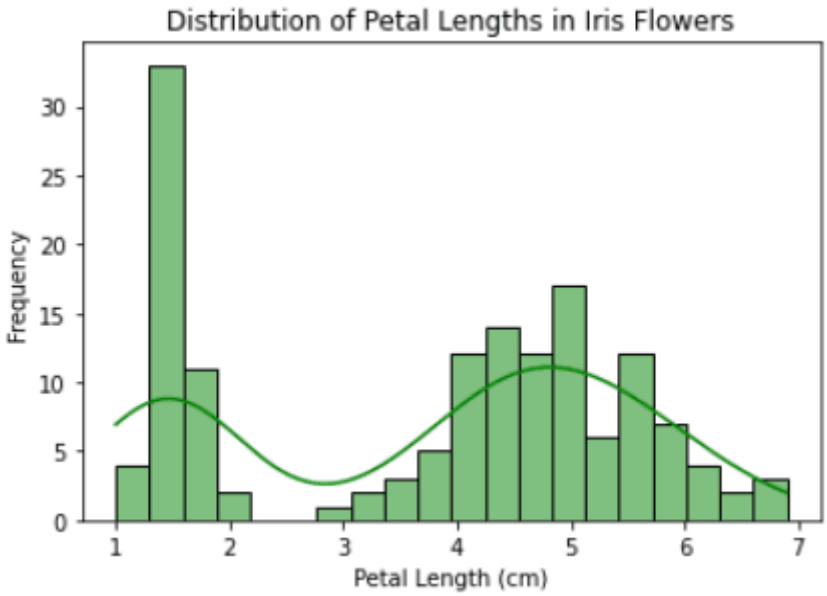
Histograms visualize the distribution of a continuous variable. In a histogram, the data is divided into bins and the height of each bin represents the frequency or count of data points within that bin. In Seaborn, histograms can be created using the `histplot()` function.

```
import seaborn as sns

iris = sns.load_dataset("iris")

sns.histplot(x="petal_length", data=iris)
```



Customizing a histogram

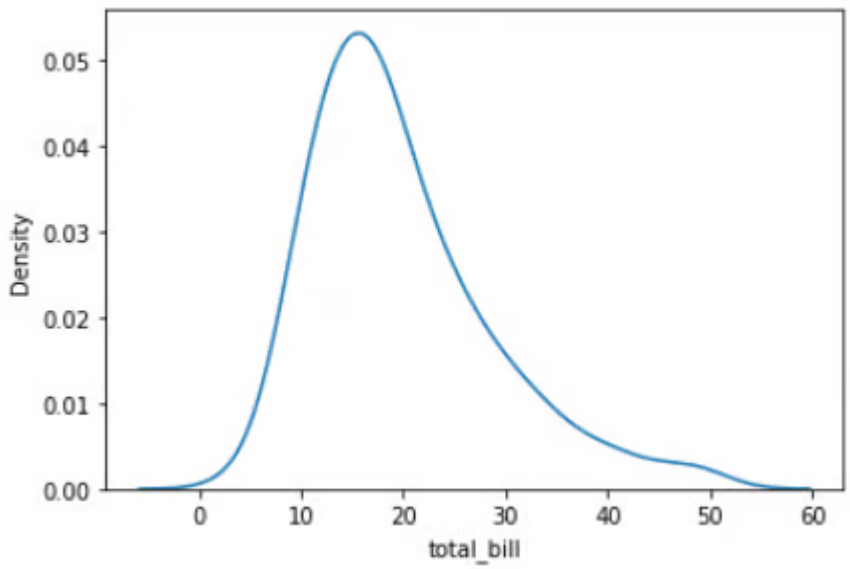
```
import seaborn as sns

import matplotlib.pyplot as plt

iris = sns.load_dataset("iris")

# customize the histogram



sns.histplot(data=iris, x="petal_length", bins=20, kde=True,
color="green")

# add labels and title

plt.xlabel("Petal Length (cm)")

plt.ylabel("Frequency")

plt.title("Distribution of Petal Lengths in Iris Flowers")

# display the plot

plt.show()
```



Seaborn density plots

Density plots, also known as kernel density plots, are a type of data visualization that display the distribution of a continuous variable. They are similar to histograms, but instead of representing the data as bars, density plots use a smooth curve to estimate the density of the data. In Seaborn, density plots can be created using the kdeplot() function.

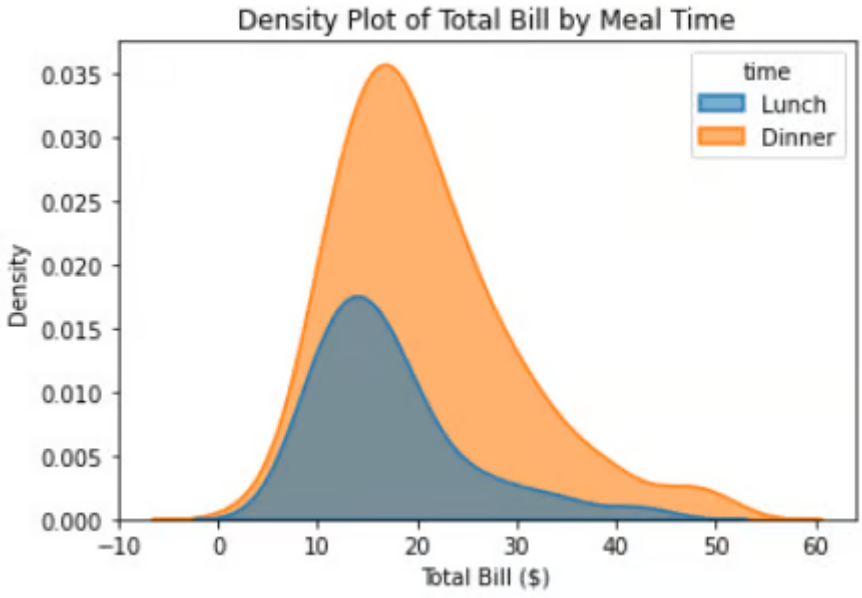
```
import seaborn as sns

tips = sns.load_dataset("tips")

 sns.kdeplot(data=tips, x="total_bill")
```



Let’s improve the plot by customizing it.

```
import seaborn as sns

import matplotlib.pyplot as plt

 Load the "tips" dataset from Seaborn

tips = sns.load_dataset("tips")



# Create a density plot of the "total_bill" column from the "tips" dataset
```

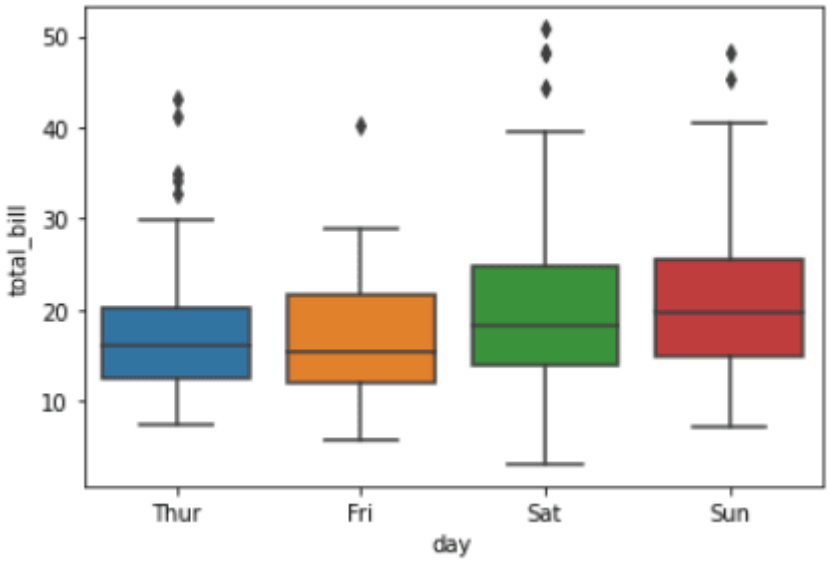
```
# We use the "hue" parameter to differentiate between "lunch" and "dinner" meal times

# We use the "fill" parameter to fill the area under the curve

# We adjust the "alpha" and "linewidth" parameters to make the plot more visually appealing

sns.kdeplot(data=tips, x="total_bill", hue="time", fill=True, alpha=0.6, linewidth=1.5)




# Add a title and labels to the plot using Matplotlib

plt.title("Density Plot of Total Bill by Meal Time")

plt.xlabel("Total Bill ($)")

plt.ylabel("Density")

# Show the plot

plt.show()
```



Seaborn box plots

Box plots are a type of visualization that shows the distribution of a dataset. They are commonly used to compare the distribution of one or more variables across different categories.

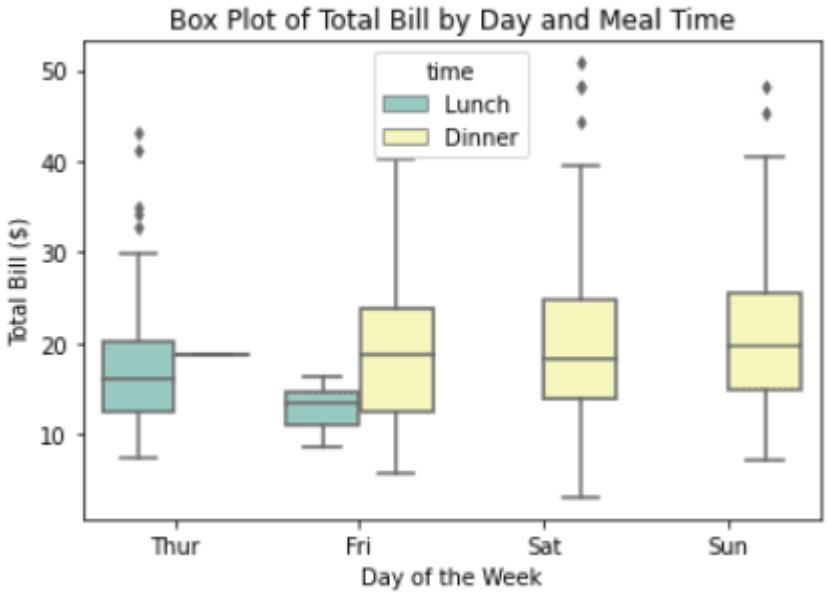
```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.boxplot(x="day", y="total_bill", data=tips)
```



Customize the box plot by including `time` column from the dataset.

```
import seaborn as sns

import matplotlib.pyplot as plt



# load the tips dataset from Seaborn

tips = sns.load_dataset("tips")

# create a box plot of total bill by day and meal time, using the "hue" parameter to differentiate between lunch and dinner
```

```
# customize the color scheme using the "palette" parameter

# adjust the linewidth and fliersize parameters to make the plot more visually appealing

sns.boxplot(x="day", y="total_bill", hue="time", data=tips, palette="Set3", linewidth=1.5, fliersize=4)

# add a title, xlabel, and ylabel to the plot using Matplotlib functions

plt.title("Box Plot of Total Bill by Day and Meal Time")



plt.xlabel("Day of the Week")

plt.ylabel("Total Bill ($)")

# display the plot

plt.show()
```



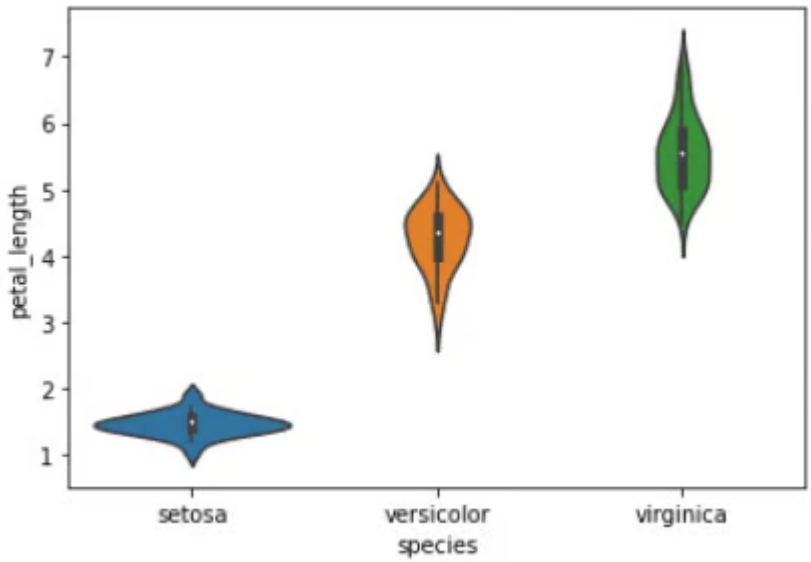
Seaborn violin plots

A violin plot is a type of data visualization that combines aspects of both box plots and density plots. It displays a density estimate of the data, usually

smoothed by a kernel density estimator, along with the interquartile range (IQR) and median in a box plot-like form.

The width of the violin represents the density estimate, with wider parts indicating higher density, and the IQR and median are shown as a white dot and line within the violin.

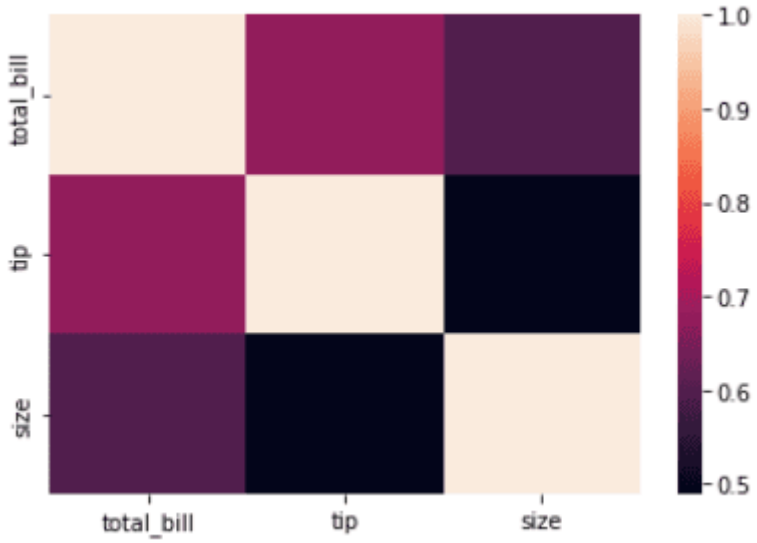
```
import seaborn as sns

# load the iris dataset from Seaborn

iris = sns.load_dataset("iris")



# create a violin plot of petal length by species

sns.violinplot(x="species", y="petal_length", data=iris)

# display the plot

plt.show()
```



Seaborn heatmaps

A heatmap is a graphical representation of data that uses colors to depict the value of a variable in a two-dimensional space. Heatmaps are commonly used to visualize the correlation between different variables in a dataset.

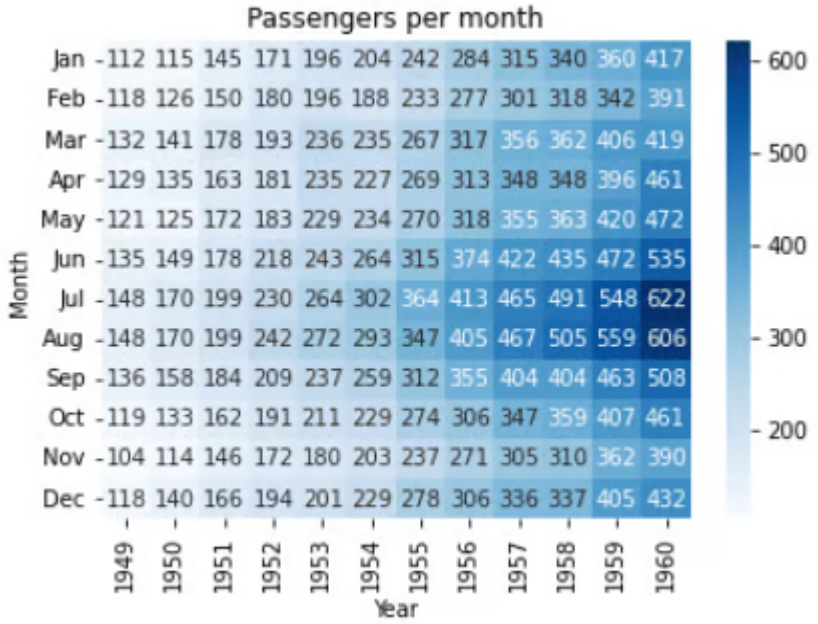
```
import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset

tips = sns.load_dataset('tips')

# Create a heatmap of the correlation between variables

corr = tips.corr()

sns.heatmap(corr)

# Show the plot

plt.show()
```



Another example of a heatmap using the `flights` dataset.

```
import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset

flights = sns.load_dataset('flights')

# Pivot the data

flights = flights.pivot('month', 'year', 'passengers')

# Create a heatmap

sns.heatmap(flights, cmap='Blues', annot=True, fmt='d')

# Set the title and axis labels
```

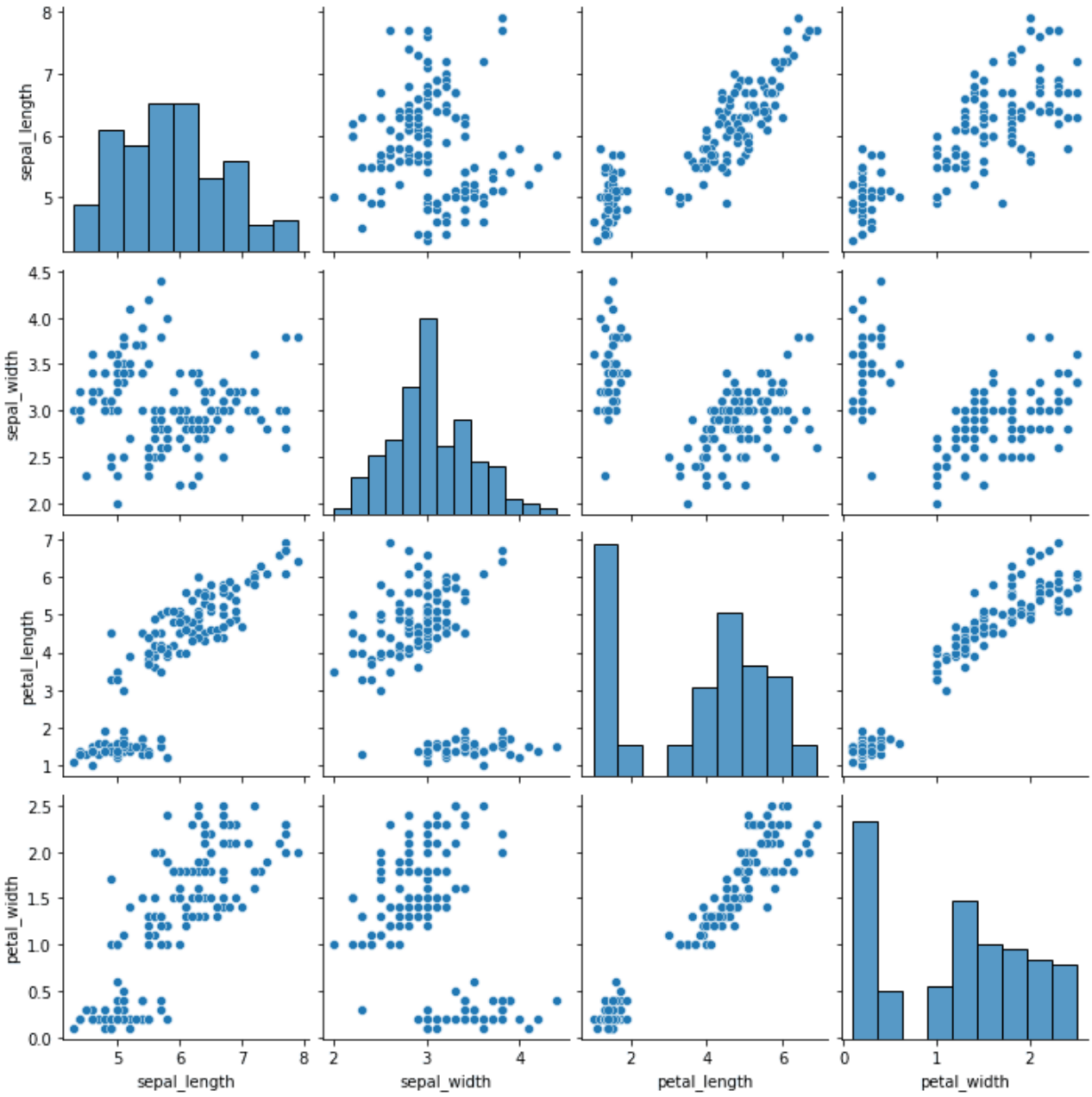
```
plt.title('Passengers per month')

plt.xlabel('Year')

plt.ylabel('Month')

# Show the plot

plt.show()
```



Seaborn pair plots

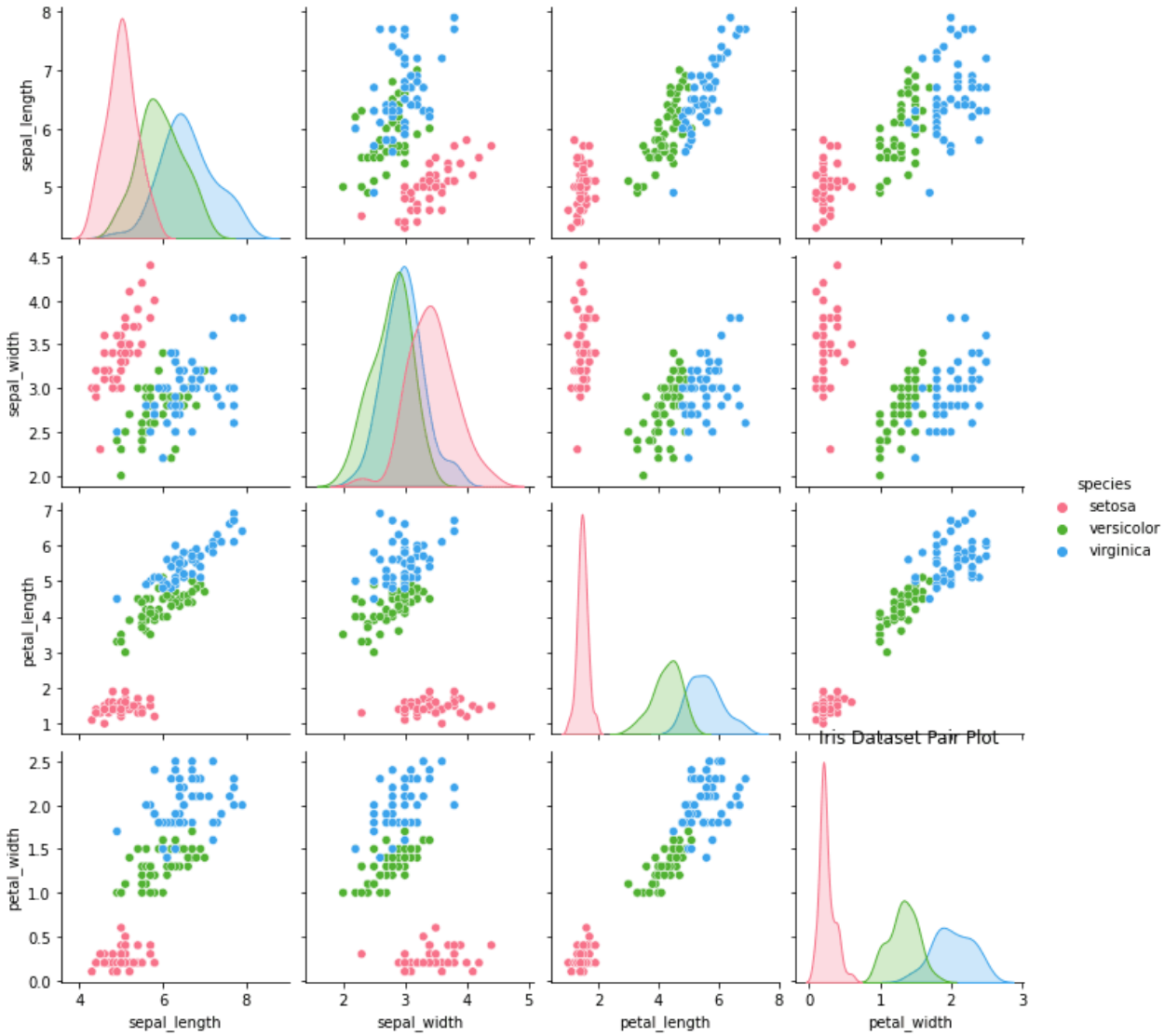
Pair plots are a type of visualization in which multiple pairwise scatter plots are displayed in a matrix format. Each scatter plot shows the relationship between two variables, while the diagonal plots show the distribution of the individual variables.

```
import seaborn as sns

# Load iris dataset

iris = sns.load_dataset("iris")

# Create pair plot

sns.pairplot(data=iris)

# Show plot

plt.show()
```


sepal_length
sepal_width
petal_length
petal_width
sepal_length
sepal_width
petal_length
petal_width

We can customize this plot by using `hue` and `diag_kind` parameter.

```
import seaborn as sns

import matplotlib.pyplot as plt

# Load iris dataset

iris = sns.load_dataset("iris")

# Create pair plot with custom settings

sns.pairplot(data=iris, hue="species", diag_kind="kde",
palette="husl")

# Set title

plt.title("Iris Dataset Pair Plot")

# Show plot

plt.show()
```


Iris Dataset Pair Plot
species
setosa
versicolor
virginica
sepal_length
sepal_width
petal_length
petal_width