JGI JAIN DEEMED-TO-BE UNIVERSITY | Powered By Futurense

Revolutionising B.Tech

JGI JAIN DEEMED-TO-BE UNIVERSITY | Powered By Futurense

Module 3: Data Visualization

Course Name:

DATA EXPLORATION AND PREPARATION [22CSE239]

Total Hours : 9

Table of Content

Aim

To equip students in the concepts and mechanisms in the Artificial Intelligence technology and its introduction.

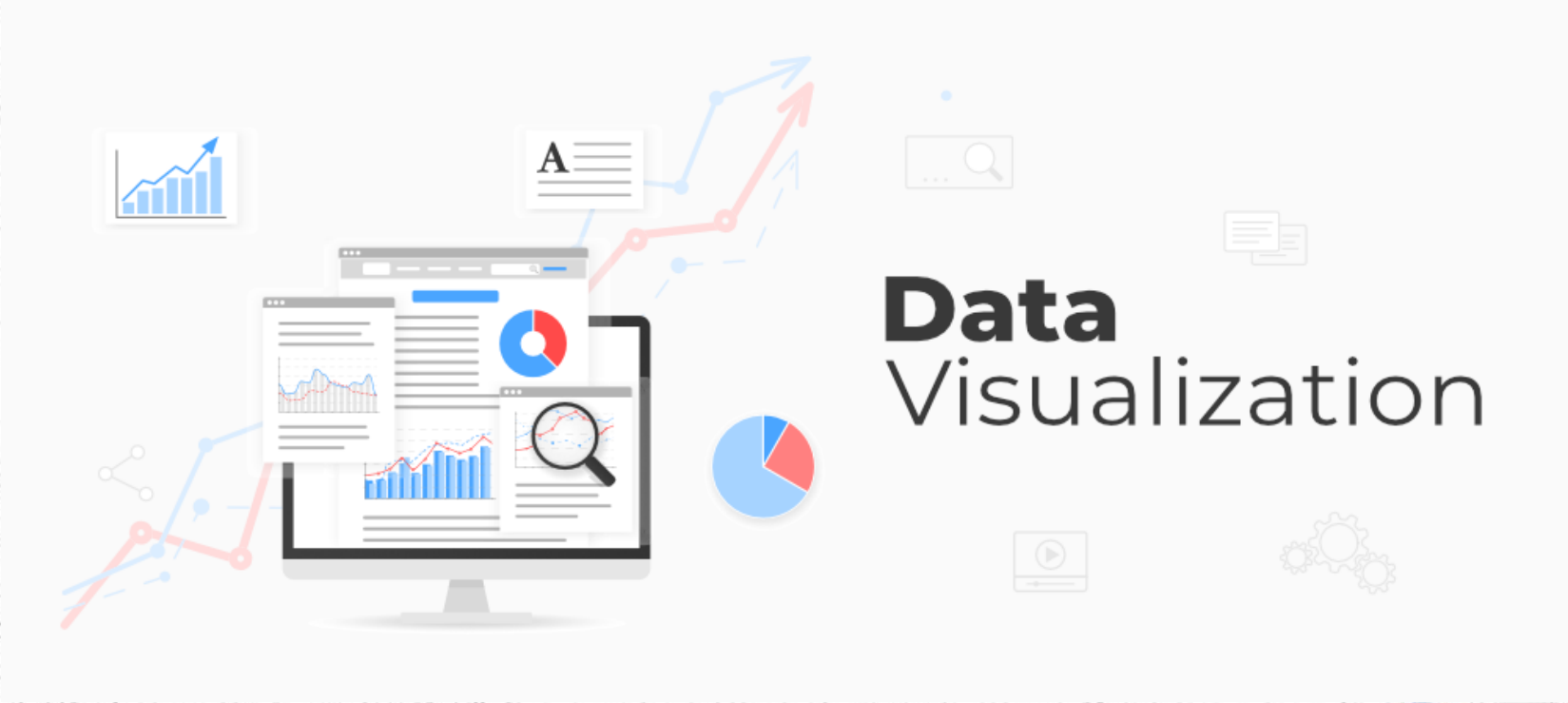
Objective

- Identify key steps in data exploration and preparation such as data cleaning, data wrangling. data transformation and data integration.
- Ability to code in a scalable environment to automate data cleaning and processing tasks.
- Understand the sources of data quality issues. different types of data, how data is stored and accessed.
- Explore data visually and statistically and identify patterns and abnormalities that require further investigation.
- Able to clean and transform data, identify missing or incorrect values, and merge or join datasets as needed.
- Evaluate the effectiveness and efficiency of data exploration and preparation methods in terms of their impact on downstream analysis and modelling.

Data Visualization

•Data visualization is the graphical representation of information and data in a pictorial or graphical format (Visualization of Data could be: charts, graphs, and maps)

• When numeric data is plotted on a graph or converted to charts its easy to identify the patterns and predict the result accurately.



Benefits of Data Visualization

- It simplifies the complex quantitative information
- It helps to analyse and explore big data easily.
- It identifies the area that need attention or improvement
- It identifies the relationship between datapoints and variables
- Explores new patterns and revels hidden pattern in the data

Data Visualization Considerations

- Three major consideration for data visualization are:
- 1. **Clarity**
- Ensure the dataset is complete and relevant. This enables the data scientist to use the patterns yield from the data in the relevant places.
- 2. **Accuracy**
- Ensure using appropriate graphical representation to convey the right message
- 3. **Efficiency**
- Use efficient visualization technique which highlights all the data points

Different Types of Data Visualization

1. Temporal:

Data for these types of visualization should satisfy both conditions: data represented should be linear and should be one-dimensional. These visualization types are represented through lines that might overlap and have a common start and finish data point.

Scatter Plots

Uses dots to represent a data point.

Types..

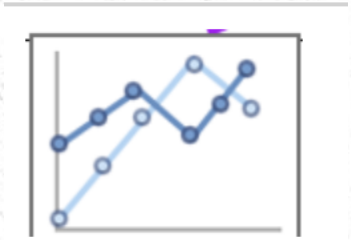
Pie Chart

- This type of visualization includes circular graphics where the arc length signifies the magnitude.

Polar area diagram

Like Pie chart, the Polar area diagram is a circular plot, except the sector angles are equal in length, and the distance of extending from center signifies the magnitude

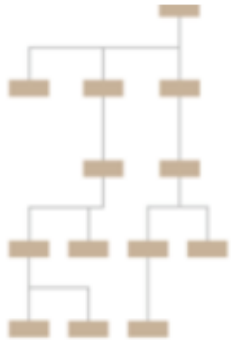
- **Line graphs**
- Like the scatter plot, the data is represented by points, except joined by lines to maintain continuity.



- **Timelines**
- In this way, display a list of data points in chronological order of time.

- **Time series sequences:** In time series, we represent the magnitude of data in a 2-D graph in chronological order of timestamp in data.

- 2. **Hierarchical:**
- These types of visualizations portray ordered groups within a larger group. In simple language, the main intuition behind these visualizations is the clusters can be displayed if the flow of the clusters starts from a single point.
- **Tree Diagram**
- In a tree diagram, the hierarchical flow is represented in the form of a tree, as the name suggests. Few terminologies for this representation are:
- – Root Node: Origination point.
- – Child node: Has a parent above
- – Leaf node: No more child node.



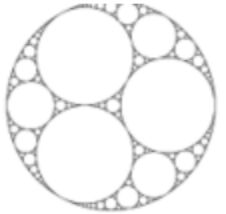
- **Ring Charts / Sunburst Diagram**
- The tree representation in the Tree diagram is converted into a radial basis. This type helps in presenting the tree in a concise size. The innermost circle is the root node. And the area of the child node signifies the % of data.

Tree Map

The tree is represented in the form of rectangles closely packed. The area signifies the quantity contained.

Circle Parking

Similar to a tree map, it uses circular packing instead of rectangles.

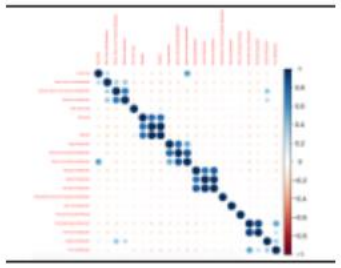


3. Network:

The visualization of these type connects datasets to datasets. These visualizations portray how these datasets relate to one another within a network.

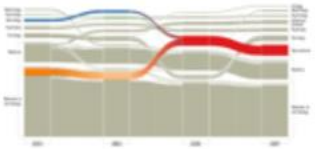
Matrix charts

This type of visualization is widely used to find the connection between different variables within themselves. For example, correlation plot.



Alluvial diagrams

This is a type of flow diagram in which the changes in the flow of the network are represented over intervals as desired by the user.

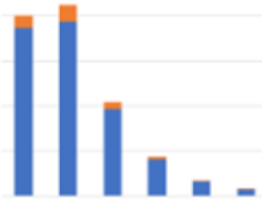


- **Word cloud**
- This is typically used for representing text data. The words are closely packed, and the size of the text signifies the frequency of the word.

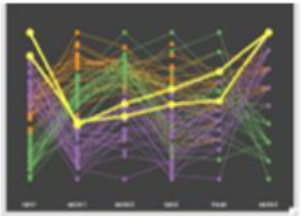
- **Node-link diagrams**
- Here the nodes are represented as dots, and the connection between nodes is presented.

- **4. Multidimensional:**
- In contrast to the temporal type of visualization, these types can have multiple dimensions. In this, we can use 2 or more features to create a 3-D visualization through concurrent layers. These will enable the user to present key takeaways by breaking a lot of non-useful data.
- **Scatter plots**
- In multi-dimensional data, we select any 2 features and then plot them in a 2-D scatter plot. By doing this we would have nC2 = n(n-1)/2 graphs.

- **Stacked bar graphs**
- The representation segment bars on top of each other. It can be either a 100% Stacked Bar graph where the segregation is represented in % or a simple stacked bar graph, which denotes the actual magnitude.



- **Parallel Co-ordinate plot**
- In this representation, a backdrop is drawn, and n parallel lines are drawn (for n-dimensional data).



5. Geospatial: These visualizations relates to present real-life physical location by crossing it over with maps (It may be a geospatial or spatial map). The intuition behind these visualizations is to create a holistic view of performance.

Flow map

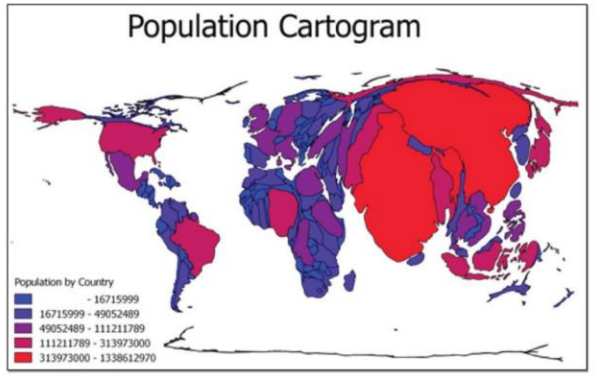
The movement of information or objects from one location to another is presented where the size of the arrow signifies the amount.

Choropleth Map

The geospatial map is colored on the basis of a particular data variable.

Cartogram

This type of representation uses the thematic variable for mapping. A cartogram is a type of map where the areas of regions or countries are distorted to represent a specific data value, like population or economic size.



Cartograms are useful for showing how many people are affected by a variable, while choropleths are useful for showing how much geographic area is affected by a variable

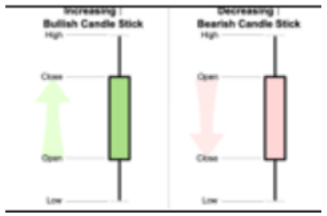
Heat Map

- These are very similar to **Choropleth** in the geospatial genre but can be used in areas apart from geospatial as well. Graphical representation of data using colors that represent different values.

6. **Miscellaneous:** These visualizations can't be generalized in a particularly large group. So instead of forming smaller groups for the individual type. Few examples are below:

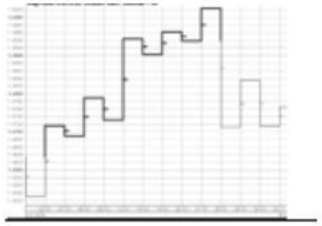
Open-High-Low-Close chart

This type of graphs is typically used for stock price representation. The increasing trend is called as **Bullish** and decreasing as **Bearish.**

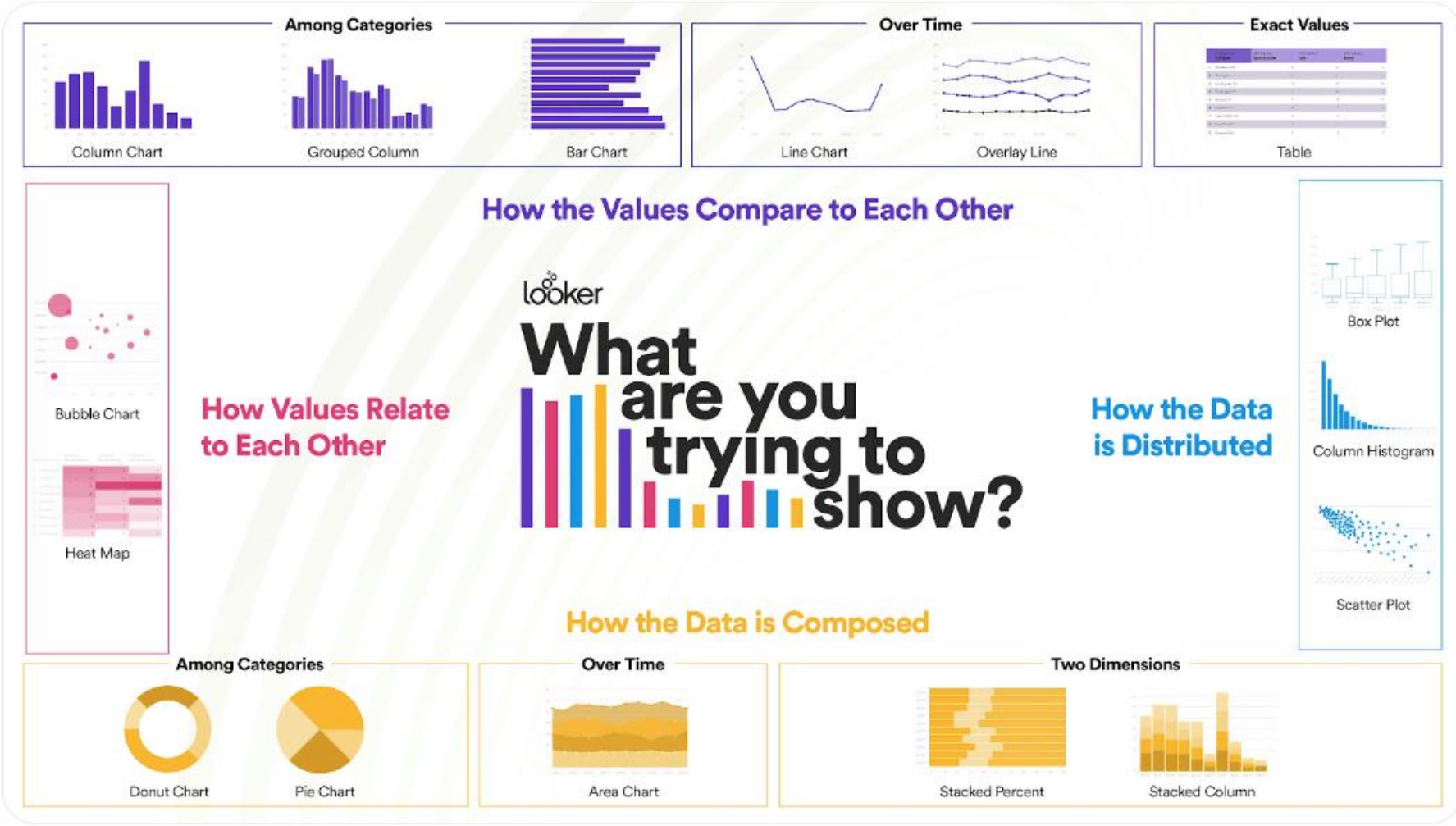


Kagi-Chart

Typically the demand-supply of an asset is represented using this chart.



Choosing right chart for data



- **Comparison**
- Comparison charts are used to compare one or more datasets. They can compare items or show differences over time. These visual aids are valuable tools for gaining insights into patterns, trends, and variations within the data sets, facilitating a comprehensive understanding of the information at hand.
- **Relationship**
- Relationship charts are used to show a connection or correlation between two or more variables. By visually representing the interdependencies and associations, relationship charts provide a clear and concise way to comprehend the intricate dynamics between the variables being examined.
- **Composition**
- Composition charts are used to display parts of a whole and change over time. These charts provide a visual representation of how different components contribute to the overall composition and how their proportions evolve over a specified period.
- **Distribution**
- Distribution charts are used to show how variables are distributed over time, helping identify outliers and trends. By illustrating the spread and pattern of data points, distribution charts enable analysts to discern patterns, anomalies, and the overall shape of the data distribution.

Comparison

- Comparison questions ask how different values or attributes within the data compare to each other.
- *Tables help you compare exact values to one another. Column and bar charts showcase comparisons across different categories, while line charts excel at showing trends over time.*

Chart type	Ideal use
Column Chart	Comparisons with **fewer categories and short names** (for display and readability purposes).
Grouped Column	Direct comparisons of **multiple data series per category**. Keep in mind that in a grouped column chart, it becomes more difficult to compare a single series across categories.
Bar Chart	Comparisons with **a large number of categories**, since stacking category names vertically makes them easier to read. Bar charts are also great for **displaying negative numbers**.
Line Chart	Showing trends in continuous data over time.
Overlay Line	Showing trends in continuous data over time with **multiple dimensions**.
Table	Displaying **exact values**; not ideal for finding trends or comparing data sets.

Relationship

- Questions in this category ask how values and attributes relate to each other.
- *Bubble charts and heat maps can help you quickly identify relationships between data points.*

Chart type	Ideal use
Bubble Chart	Showing the relationship between data points with three variables.
Heat Map	Quickly relating information across a larger data set of exact values.

Composition

- Composition questions ask what general features are present in the data set.
- *Donut and pie charts are great choices to show composition when simple proportions are useful. Area charts put the composition of data within the context of trends over time. Stacked bar, percent, and column charts show an overview of the data's composition.*

Chart type	Ideal use
Donut Chart	Examining part-to-whole relationships when **simple proportions** provide meaningful information and **pivots for multiple categories** are needed.
Pie Chart	Examining part-to-whole relationships when **simple proportions** provide meaningful information.
Area Chart	Showing trends in continuous data over time in the context of part-to-whole relationships**. Area charts are not ideal for distinguishing the trends of individual data sets over time because the overlapping colors can prevent the trend from being easily seen.
Stacked Bar	Showing an **overview of the composition of data** among many categories, or when working with **time**.
Stacked Percent	Times when **the pure composition of data is the message you want to deliver** and exact values aren't necessary. This chart also works well when comparing the proportional contributions of different categories.
Stacked Column	Showing **an overview of the composition of data**. Avoid having too many segments in a column. If you want to compare the same segment across bars, use a plain column chart instead.

Distribution

- Distribution questions seek to understand how individual data points are distributed within the broader data set.
- *Box plots show distribution based on a statistical summary, while column histograms are great for finding the frequency of an occurrence. Scatter plots are best for showing distribution in large data sets.*

Chart type	Ideal use
Box Plot	Showing how data is distributed based on a five-number **statistical summary**. A small "box" indicates that most of the data falls within a consistent range, while a larger box indicates the data is more widely distributed.
Column Histogram	Showing distribution of variables, plotting quantitative data, and identifying the frequency of something occurring within a bucketed range of values. This differs from a column chart because **histograms have quantitative data on both axes**, rather than relating information about categories.
Scatter Plot	Showing **how data is distributed with two variables**, especially good for large data sets; quickly identifying specific data points that are **outliers**.

Best practices for Effective Visualization.

- Visualizing data is an important aspect of presenting insights clearly.
- **1. Keep it simple**
- Data visualization is all about communicating data insights in a way that is both accurate and easy to understand. When creating visuals, less is almost always more. Data overload can quickly lead to confusion, so it's important to only include the most important information.
- **2. Use intuitive visual cues**
- The best data visualizations make use of visual cues that are immediately intuitive to the viewer. For example, using different colors to represent different data points can help the viewer more easily understand the relationships between those points.
- **3. Avoid distracting elements**
- In addition to keeping the visualization itself simple, it's also important to avoid any extra elements that could serve as distractions. This includes things like excessive use of color, busy background patterns, and cluttered layouts.
- **4. Tell a story**
- Data visualization is an opportunity to tell a story with the data. An effective visualization will effectively communicate the key insights from the data in a way that is both accurate and easy to understand.

- **5. Use data points carefully**
- When using data points in a visualization, it's important to carefully consider which points are most important to include. Data points should be chosen based on their ability to accurately represent the underlying data and tell the story you want to communicate.
- **6. Use visual hierarchy effectively**
- An effective data visualization makes use of visual hierarchy to direct the viewer's attention to the most important information. This can be accomplished through the use of things like size, color, and position.
- **7. Use effective labels**
- Labels are an important part of any data visualization. They should be clear and concise, and they should accurately describe the data that is being represented.
- **8. Test data visualizations with users**
- Before finalizing a data visualization, it's important to test it with actual users. This will help ensure that the visualization is effective in communicating the story you want to tell.
- **9. Continuously iterate and improve**
- Data visualization is an iterative process, and it's important to continually strive for improvement. As new data becomes available, or as new insights are gained, be sure to update the visualization accordingly.

Data Visualization Libraries:

Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, analyze.

Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs.

- Matplotlib
- Seaborn
- Bokeh
- Plotly

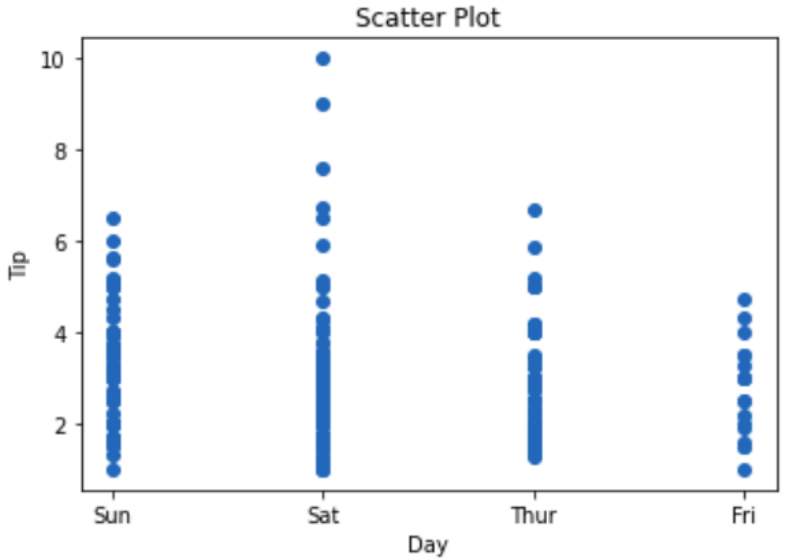
Matplotlib

- Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.
- To install this type the below command in the terminal.
- **pip install matplotlib**

- **Scatter Plot**
- Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The **scatter()** method in the matplotlib library is used to draw a scatter plot.

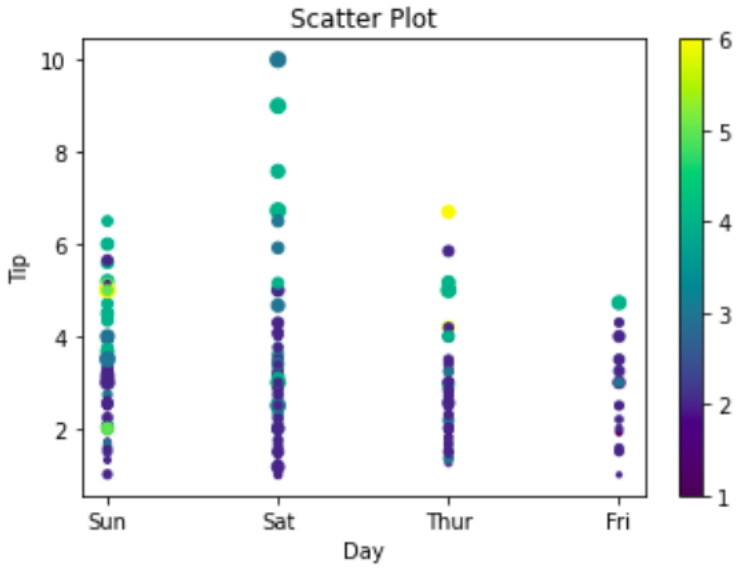
```python
import pandas as pd
import matplotlib.pyplot as plt


# reading the database
data = pd.read_csv("tips.csv")

# Scatter plot with day against tip
plt.scatter(data['day'], data['tip'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.show()
```

Output:



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


# reading the database
data = pd.read_csv("tips.csv")

# Scatter plot with day against tip
plt.scatter(data['day'], data['tip'], c=data['size'],
            s=data['total_bill'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.colorbar()

plt.show()
```

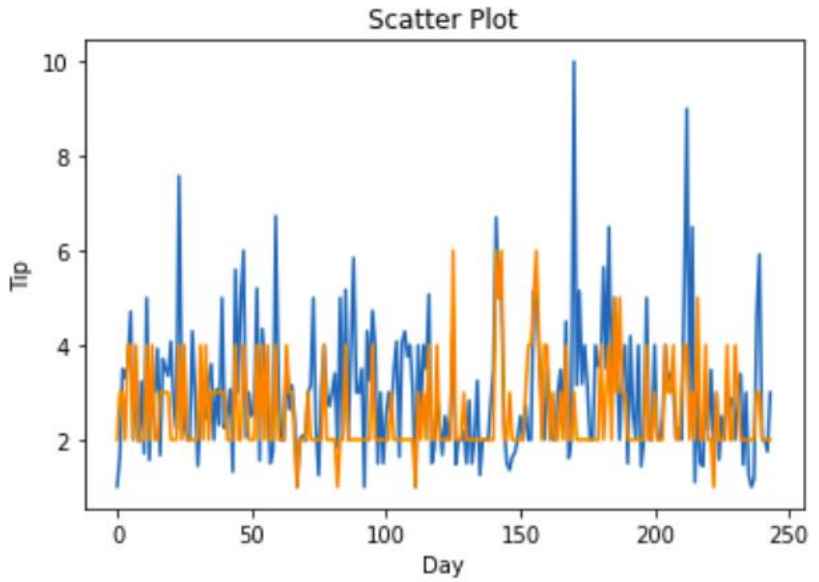
Output:



- **Line Chart**
- Line Chart is used to represent a relationship between two data X and Y on a different axis. It is plotted using the **plot()** function.

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


# reading the database
data = pd.read_csv("tips.csv")

# Scatter plot with day against tip
plt.plot(data['tip'])
plt.plot(data['size'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.show()
```

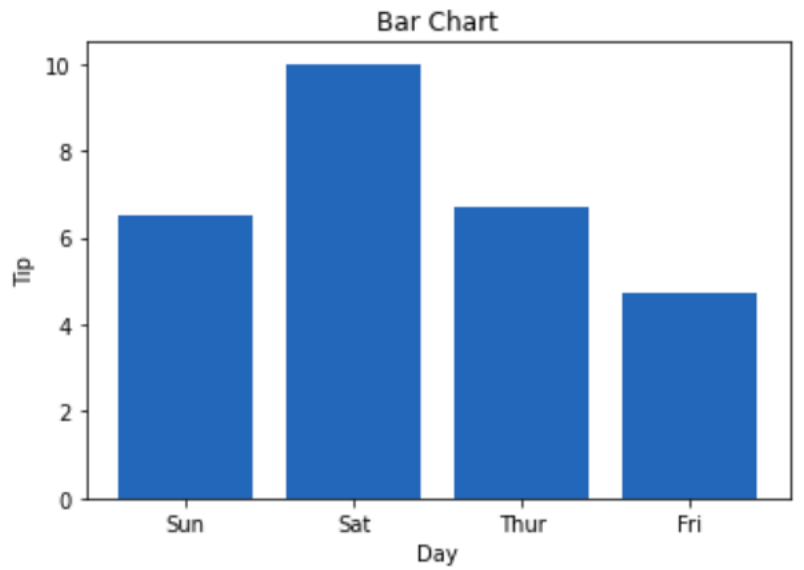
Output:



- **Bar Chart**
- A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the **bar()** method.

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


# reading the database
data = pd.read_csv("tips.csv")

# Bar chart with day against tip
plt.bar(data['day'], data['tip'])

plt.title("Bar Chart")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

# Adding the legends
plt.show()
```

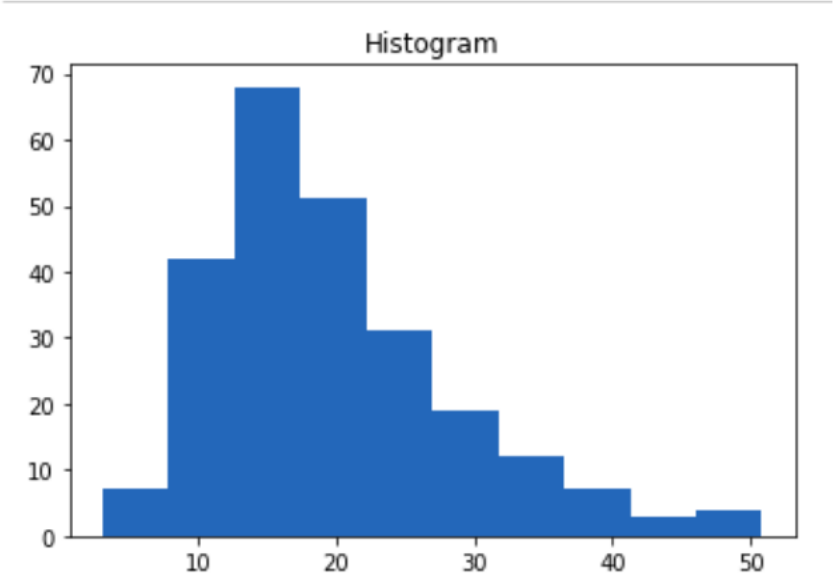
Output:



- **Histogram**
- A histogram is basically used to represent data in the form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The **hist()** function is used to compute and create a histogram. In histogram, if we pass categorical data then it will automatically compute the frequency of that data i.e. how often each value occurred.

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


# reading the database
data = pd.read_csv("tips.csv")

# histogram of total_bills
plt.hist(data['total_bill'])

plt.title("Histogram")

# Adding the legends
plt.show()
```

Output:

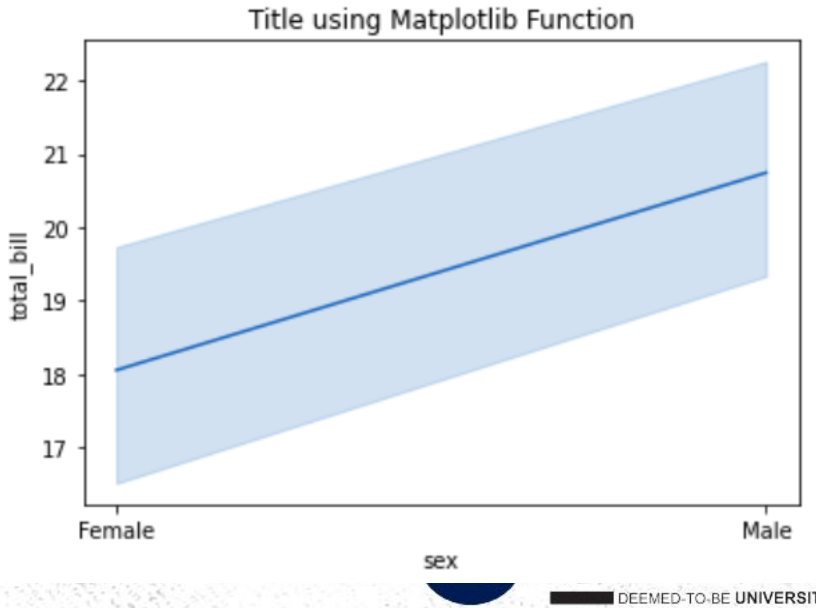


Seaborn Library for Statistical Analysis

- **Seaborn** is a high-level interface built on top of the Matplotlib. It provides beautiful design styles and color palettes to make more attractive graphs.
- To install seaborn
- pip install seaborn
- Seaborn is built on the top of Matplotlib, therefore it can be used with the Matplotlib as well. Using both Matplotlib and Seaborn together is a very simple process.

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


# reading the database
data = pd.read_csv("tips.csv")

# draw lineplot
sns.lineplot(x="sex", y="total_bill", data=data)

# setting the title using Matplotlib
plt.title('Title using Matplotlib Function')

plt.show()
```

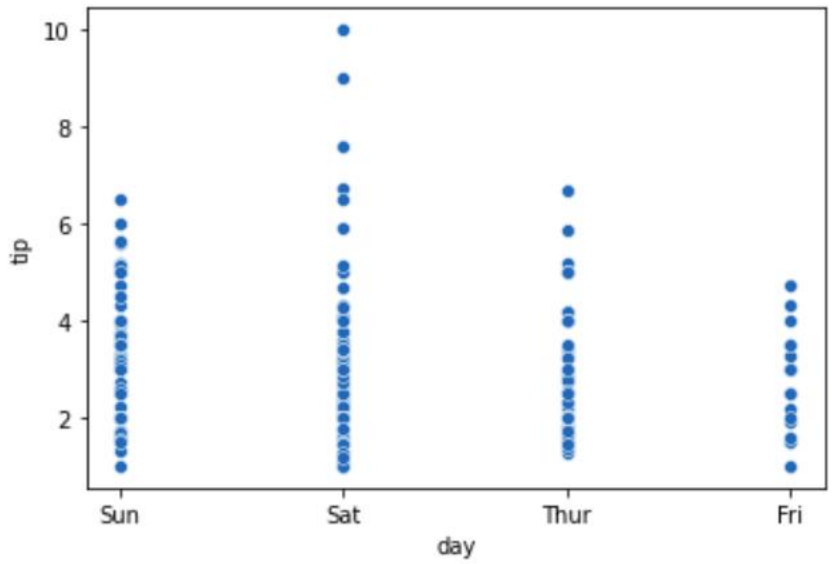
Output:



- **Scatter Plot**
- Scatter plot is plotted using the **scatterplot()** method. This is similar to Matplotlib, but additional argument data is required.

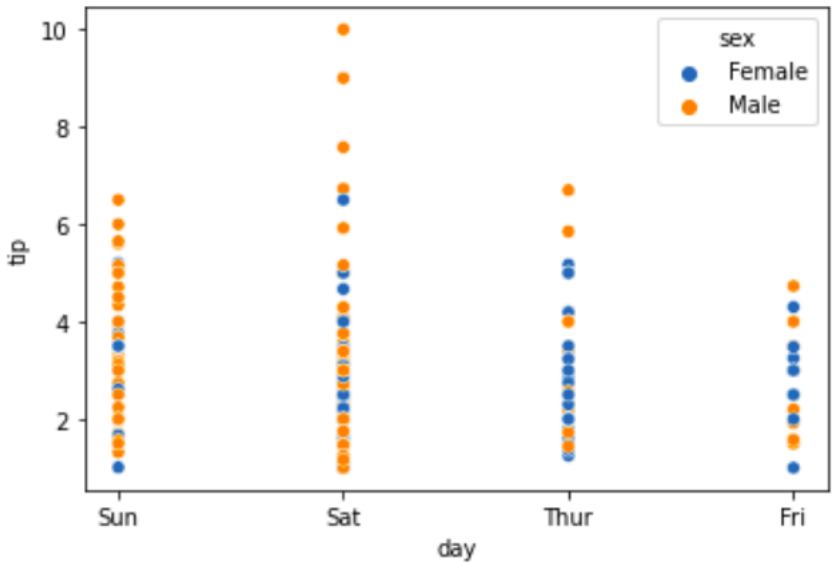
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

sns.scatterplot(x='day', y='tip', data=data,)
plt.show()
```

Output:



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

# reading the database
data = pd.read_csv("tips.csv")

sns.scatterplot(x='day', y='tip', data=data,
                hue='sex')
plt.show()
```

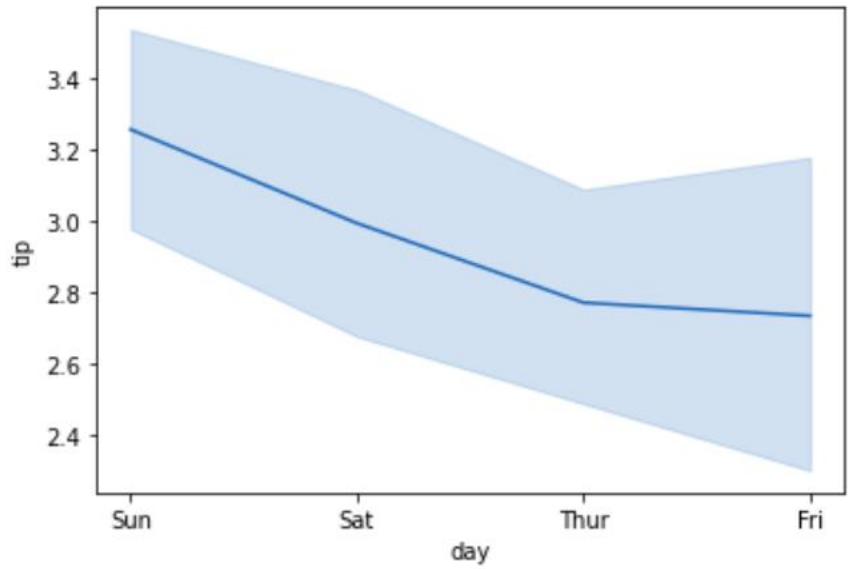
Output:



- **Line Plot**
- Line Plot in Seaborn plotted using the **lineplot()** method.

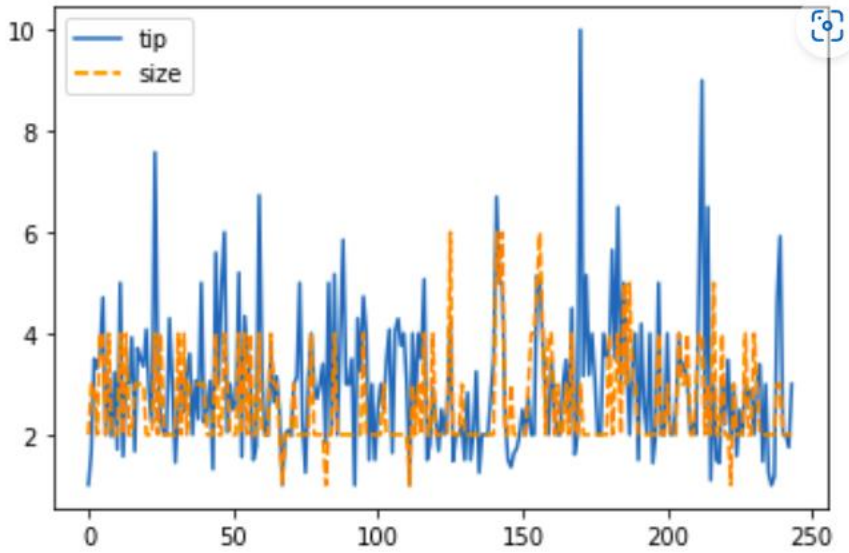
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd


# reading the database
data = pd.read_csv("tips.csv")

sns.lineplot(x='day', y='tip', data=data)
plt.show()
```

Output:



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


# reading the database
data = pd.read_csv("tips.csv")

# using only data attribute
sns.lineplot(data=data.drop(['total_bill'], axis=1))
plt.show()
```

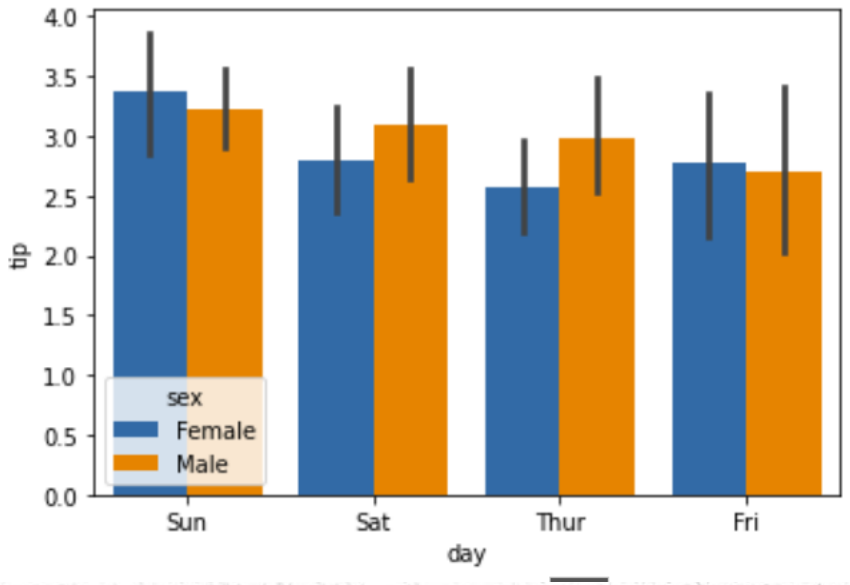
Output:



- **Bar Plot**
- Bar Plot in Seaborn can be created using the **barplot()** method.

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


# reading the database
data = pd.read_csv("tips.csv")

sns.barplot(x='day',y='tip', data=data,
            hue='sex')

plt.show()
```

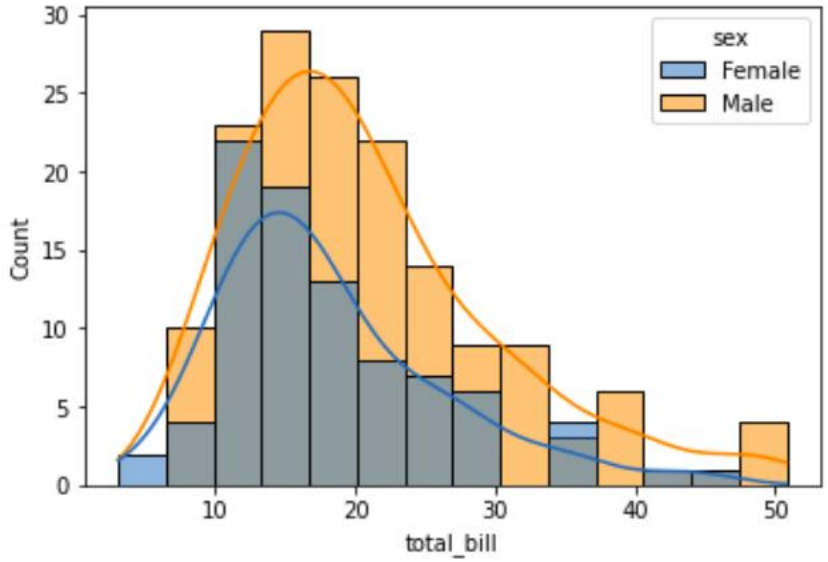
Output:



- **Histogram**
- The histogram in Seaborn can be plotted using the **histplot()** function.

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


# reading the database
data = pd.read_csv("tips.csv")

sns.histplot(x='total_bill', data=data, kde=True, hue='sex')

plt.show()
```

Output:



Plotly Express Library.

- **Plotly** library in Python is an open-source library that can be used for data visualization and understanding data simply and easily.
- Plotly has hover tool capabilities that allow us to detect any outliers or anomalies in a large number of data points.
- It is visually attractive and can be accepted by a wide range of audiences.
- Plotly generally allows us endless customization of our graphs and makes our plot more meaningful and understandable for others.
- To Install
- pip install plotly

Package Structure of Plotly

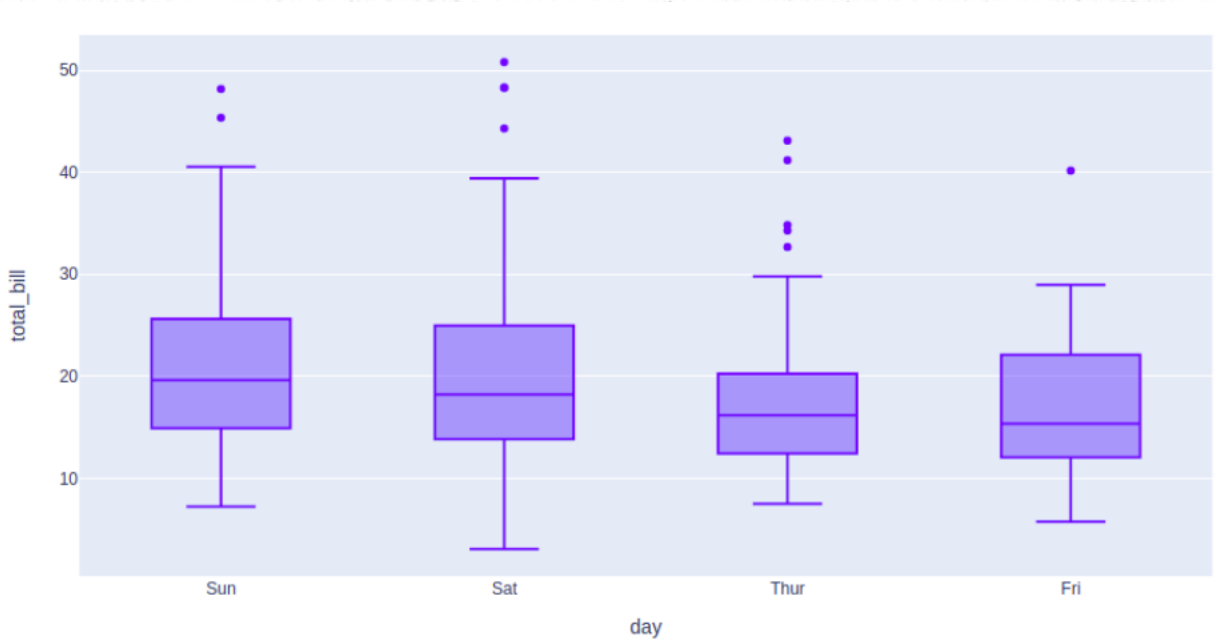
- There are three main modules in Plotly. They are:
- plotly.plotly
- plotly.graph.objects
- plotly.tools
- **plotly.plotly** acts as the interface between the local machine and Plotly. It contains functions that require a response from Plotly's server.
- **plotly.graph_objects** module contains the objects (Figure, layout, data, and the definition of the plots like scatter plot, line chart) that are responsible for creating the plots. The Figure can be represented either as dict or instances of **plotly.graph_objects.Figure** and these are serialized as JSON before it gets passed to plotly.js. Consider the below example for better understanding.

Creating Different Types of Charts

1. Line Chart
2. Bar Chart
3. Histograms
4. Scatter Plot and Bubble charts
5. Pie Charts
6. Box Plots
7. Violin plots
8. Gantt Charts
9. Contour Plots
10. Heatmaps
11. Error Bars
12. 3D Line Plots
13. 3D Scatter Plot Plotly
14. 3D Surface Plots

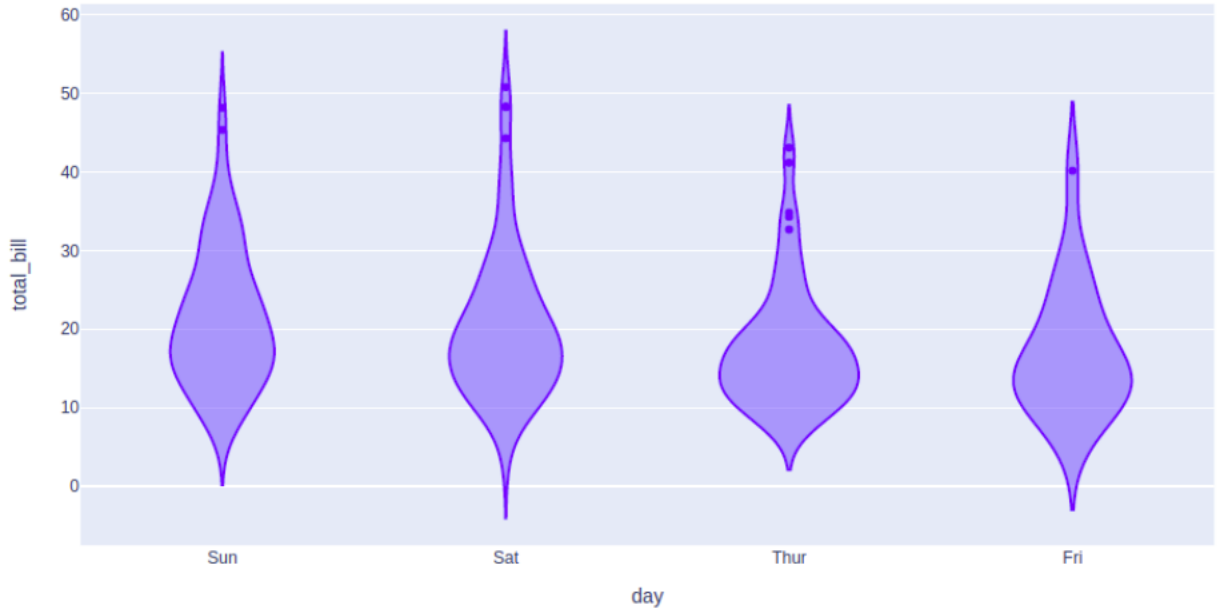
Box Plots

- A **Box Plot** is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum.
- In the box plot, a box is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution.



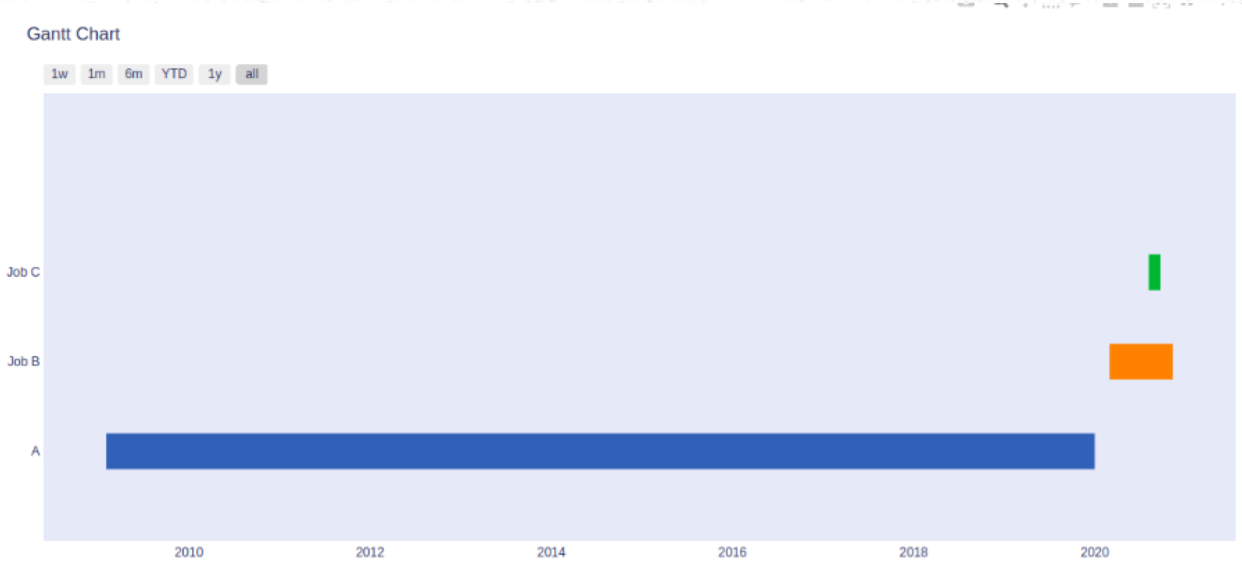
Violin plots

- **A Violin Plot** is a method to visualize the distribution of numerical data of different variables. It is similar to Box Plot but with a rotated plot on each side, giving more information about the density estimate on the y-axis. The density is mirrored and flipped over and the resulting shape is filled in, creating an image resembling a violin.
- The advantage of a violin plot is that it can show nuances in the distribution that aren't perceptible in a boxplot. On the other hand, the boxplot more clearly shows the outliers in the data.



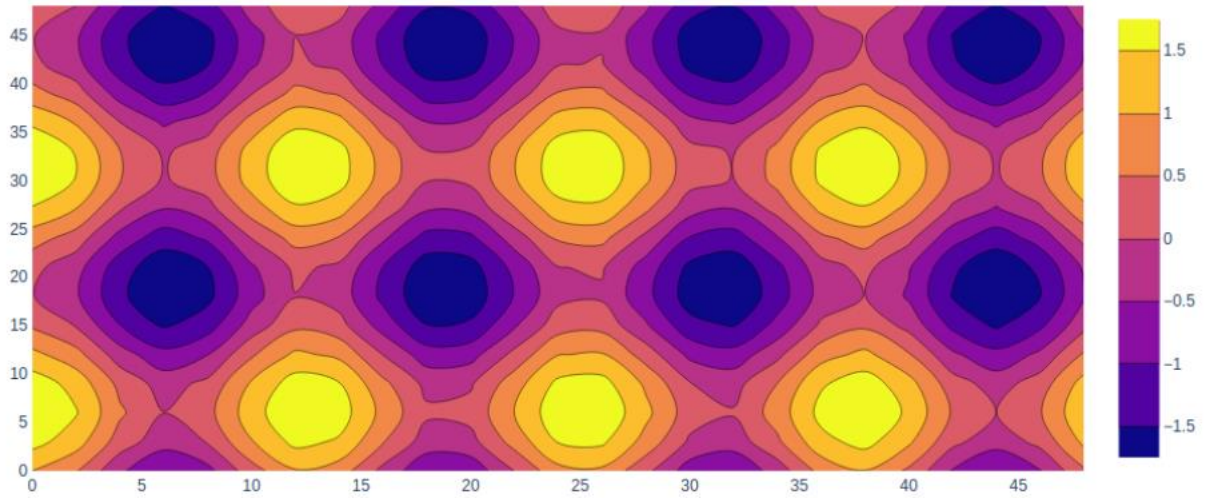
Gantt Charts

- **Generalized Activity Normalization Time Table (GANTT) chart** is type of chart in which series of horizontal lines are present that show the amount of work done or production completed in given period of time in relation to amount planned for those projects.



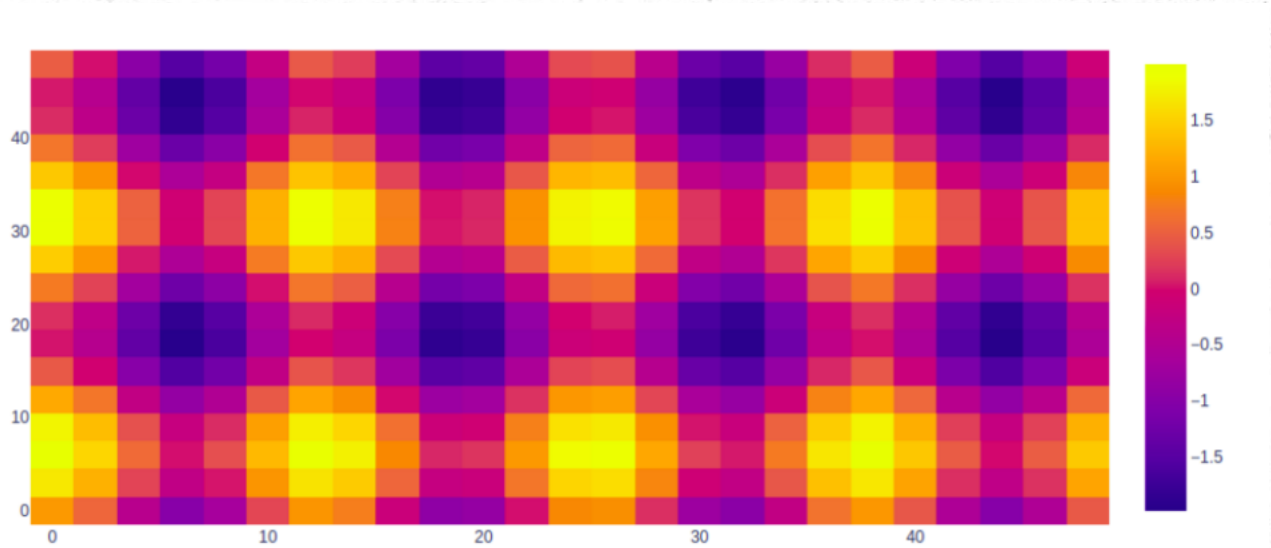
Contour Plots

- A **Contour plots** also called level plots are a tool for doing multivariate analysis and visualizing 3-D plots in 2-D space. If we consider X and Y as our variables we want to plot then the response Z will be plotted as slices on the X-Y plane due to which contours are sometimes referred as Z-slices or iso-response.



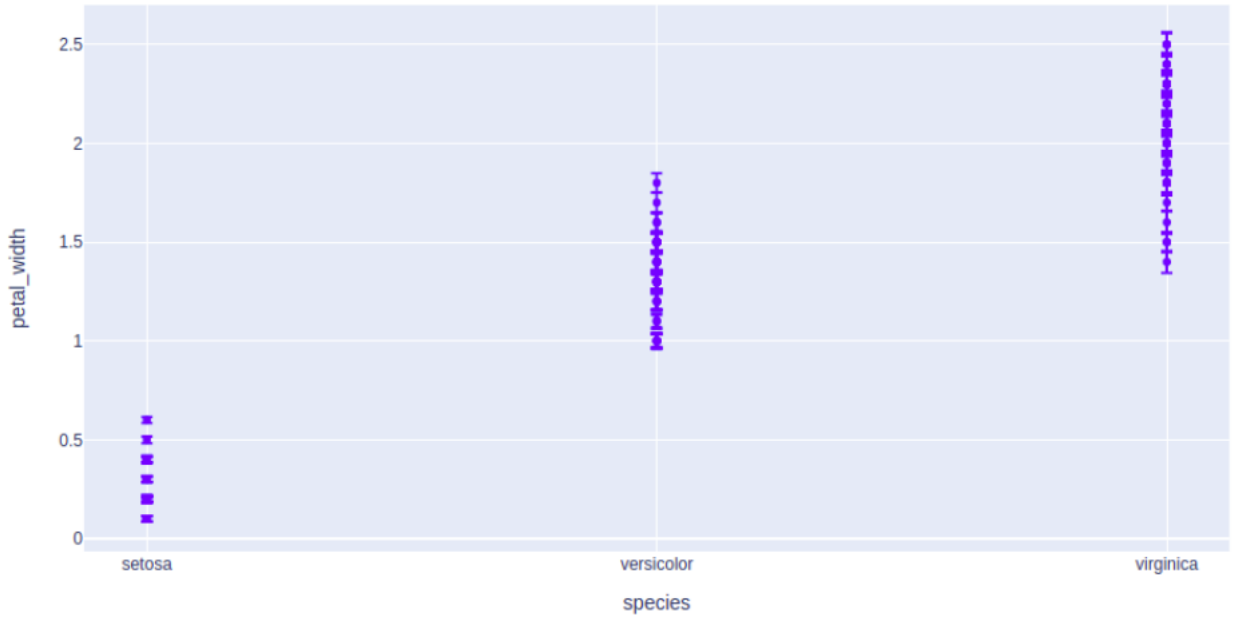
Heatmap

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix.



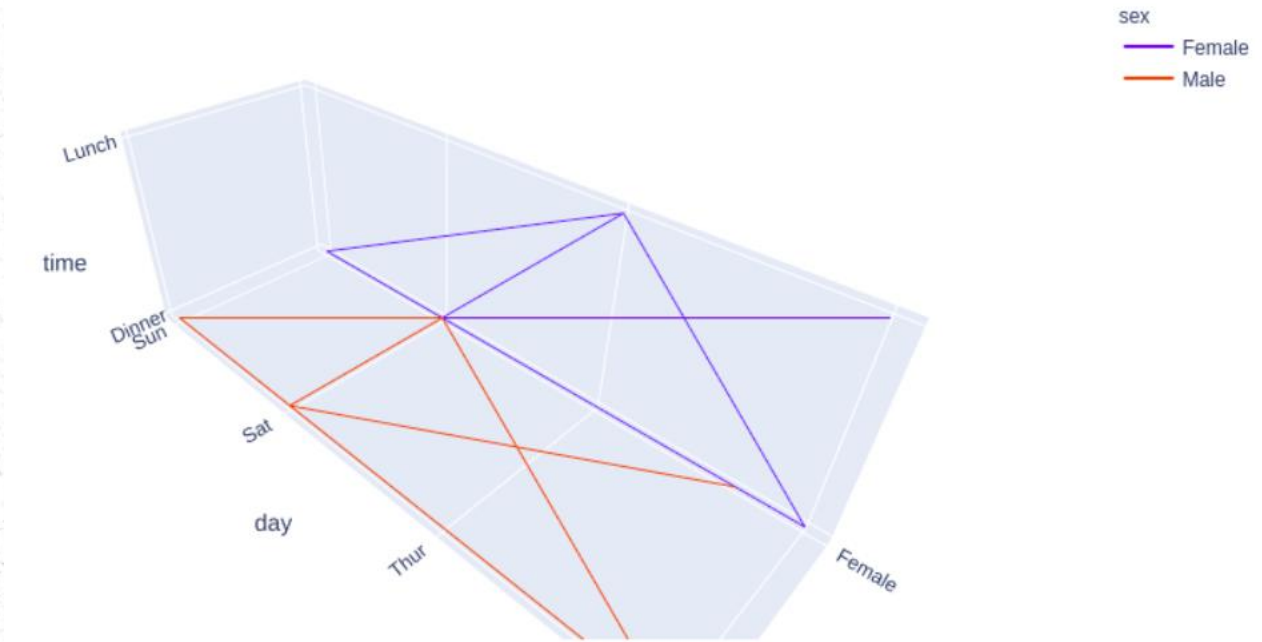
Error Bars

- For functions representing 2D data points such as px.scatter, px.line, px.bar, etc., **error bars** are given as a column name which is the value of the error_x (for the error on x position) and error_y (for the error on y position). Error bars are the graphical presentation alternation of data and used on graphs to imply the error or uncertainty in a reported capacity.



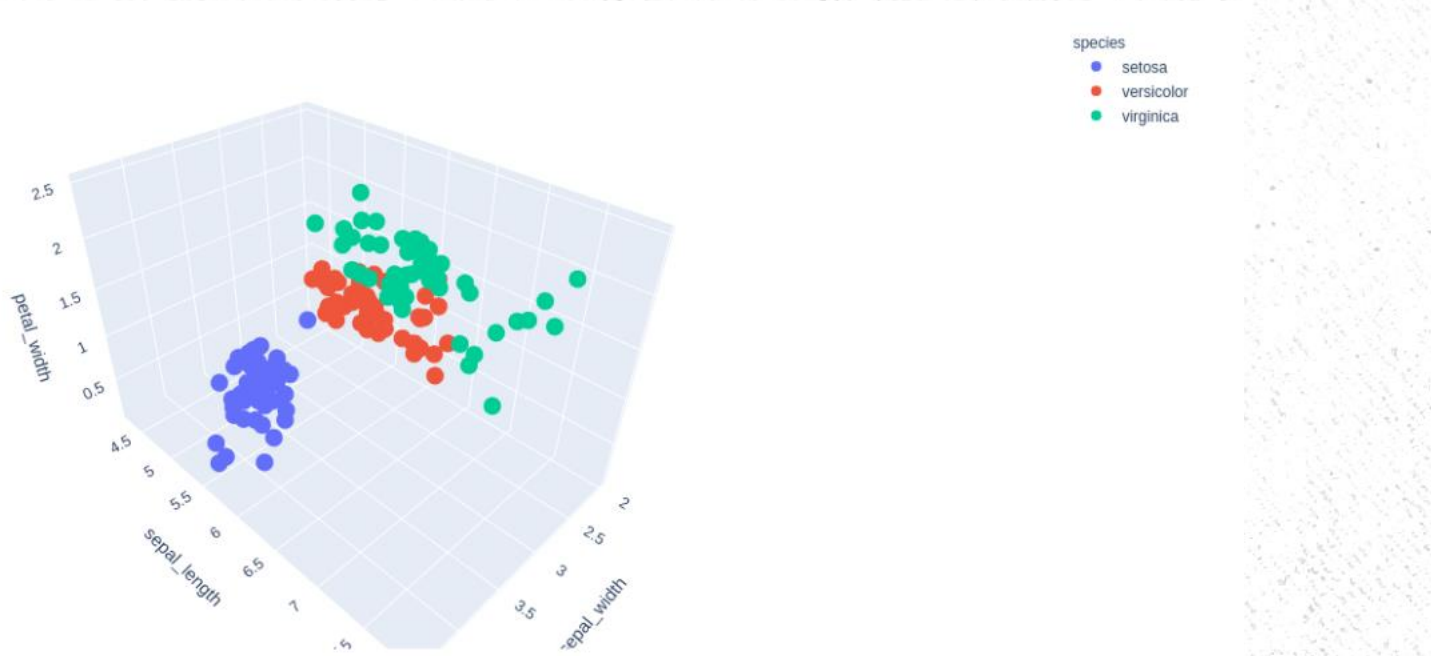
3D Line Plots

- Line plot in plotly is much accessible and illustrious annexation to plotly which manage a variety of types of data and assemble easy-to-style statistic. With **px.line_3d** each data position is represented as a vertex (which location is given by the x, y and z columns) of a polyline mark in 3D space.



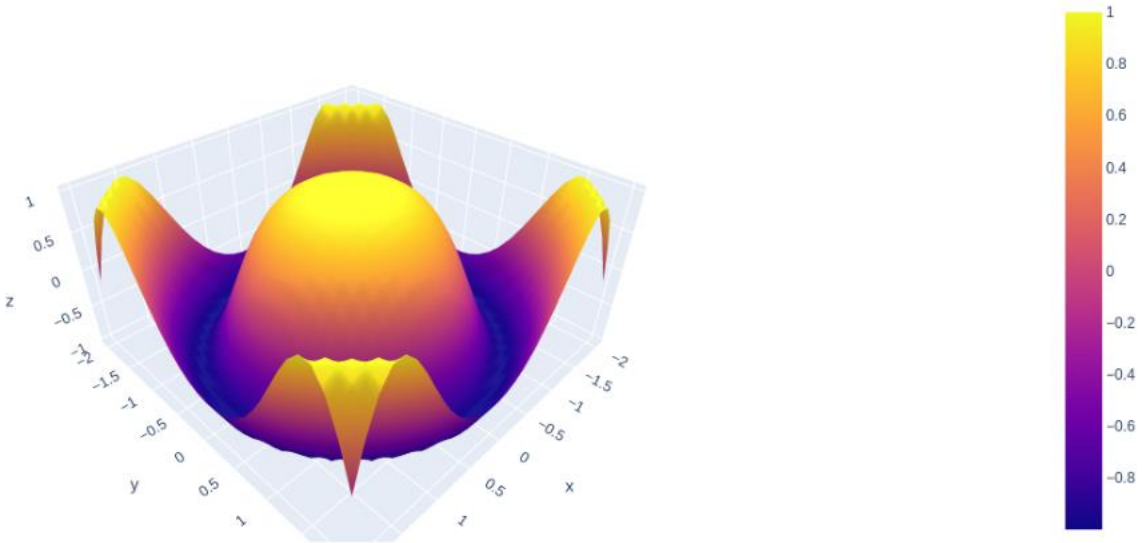
3D Scatter Plot Plotly

- **3D Scatter Plot** can plot two-dimensional graphics that can be enhanced by mapping up to three additional variables while using the semantics of hue, size, and style parameters. All the parameter control visual semantic which are used to identify the different subsets. Using redundant semantics can be helpful for making graphics more accessible. It can be created using the scatter_3d function of plotly.express class.



3D Surface Plots

- **Surface plot** is those plot which has three-dimensions data which is X, Y, and Z. Rather than showing individual data points, the surface plot has a functional relationship between dependent variable Y and have two independent variables X and Z. This plot is used to distinguish between dependent and independent variables.



Interacting with the Plots

- Plotly provides various tools for interacting with the plots such as adding dropdowns, buttons, sliders, etc. These can be created using the update menu attribute of the plot layout.
- **Creating Dropdown Menu in Plotly**
- A **drop-down menu** is a part of the menu-button which is displayed on a screen all the time. Every menu button is associated with a Menu widget that can display the choices for that menu button when clicked on it. In plotly, there are 4 possible methods to modify the charts by using update menu method.
- **restyle:** modify data or data attributes
- **relayout:** modify layout attributes
- **update:** modify data and layout attributes
- **animate:** start or pause an animation

Adding Buttons to the Plot

- In plotly, **actions custom Buttons** are used to quickly make actions directly from a record. Custom Buttons can be added to page layouts in CRM, Marketing, and Custom Apps. There are also 4 possible methods that can be applied in custom buttons:
- **restyle:** modify data or data attributes
- **relayout:** modify layout attributes
- **update:** modify data and layout attributes
- **animate:** start or pause an animation

Creating Sliders and Selectors to the Plot

- In plotly, the **range slider** is a custom range-type input control. It allows selecting a value or a range of values between a specified minimum and maximum range. And the range selector is a tool for selecting ranges to display within the chart. It provides buttons to select pre-configured ranges in the chart. It also provides input boxes where the minimum and maximum dates can be manually input.

Reference Links

- Type of Data Visualization | 6 Awesome Types of Data Visualization (educba.com)
- How to choose the best chart or graph for your data | Google Cloud Blog
- 9 Best Practices and Tips For Effective Data Visualization (thoughtspot.com)
- Data Visualization with Python – GeeksforGeeks
- Plotly tutorial – GeeksforGeeks

Reference Material:

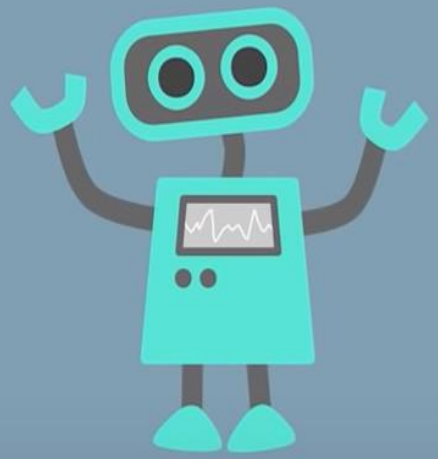
- 1. Python for Data Analysis, Data Wrangling with Numpy, Pandas and Jupyter, Third Edition, O'Reilly.
- 2. Hands on Exploratory Data Analysis using Python, Packt.

Self Assessments and Activities

Here are some assessments you can use to evaluate students understanding of Data cleaning process

- Quizzes
- In class Participation
- Programming assessments

Did You Know?



A statistician and data scientist known for creating several widely used R packages for data analysis and visualization

HADLEY WICKHAM

Summary

Outcomes: students able to

a. Able to implement data cleaning process

Thank you