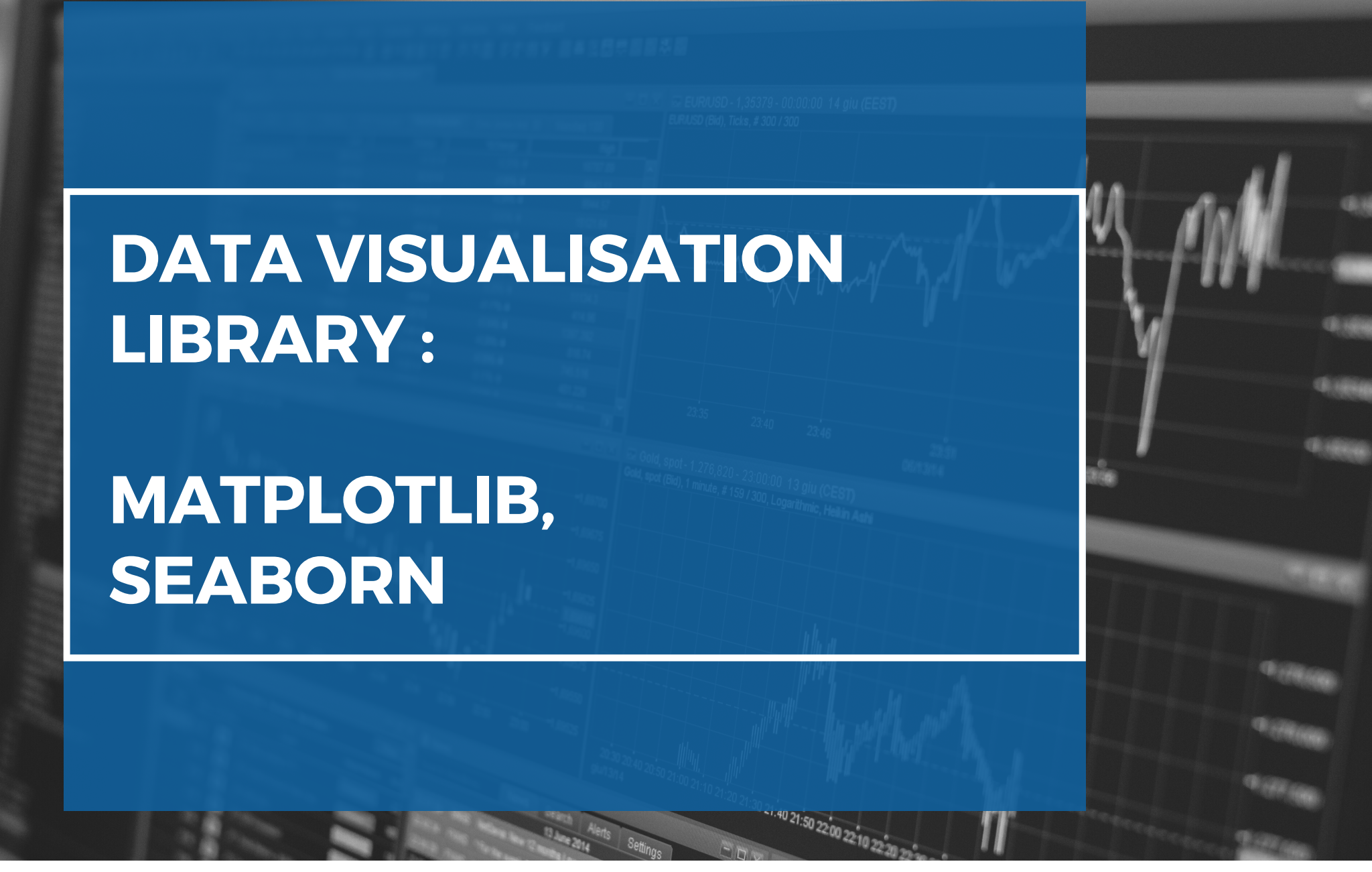


DATA VISUALISATION LIBRARY :

MATPLOTLIB, SEABORN

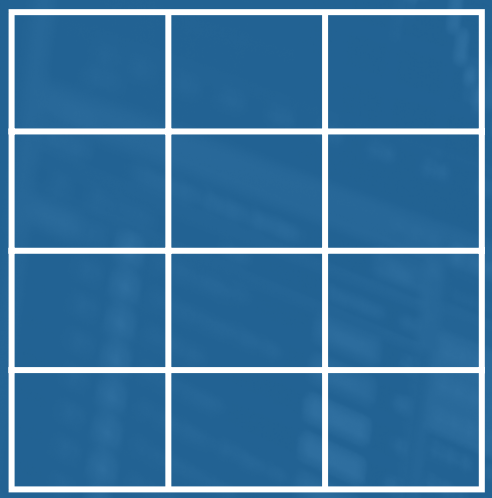
Univariate, Bivariate & Multivariate

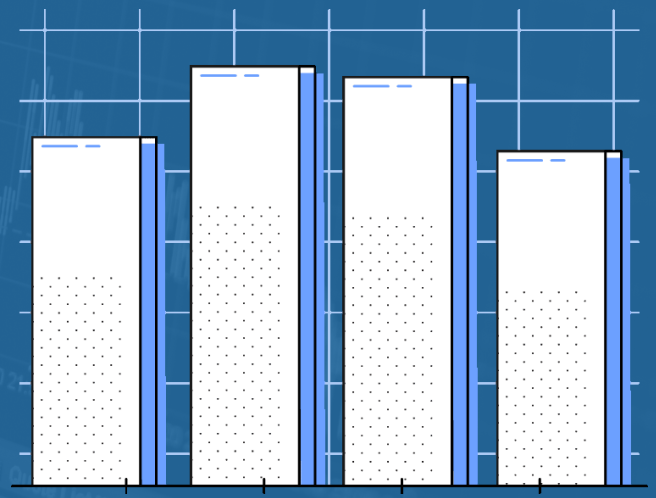
What is Data Visualization?

Data visualization is a way to show complex data in a form that is graphs and charts .

- Bar Graph,
- Pie Chart,
- Line Graph,
- Histogram Chart,
- Area Graph,
- Dot Graph,
- Scatter Plot,
- Bubble Chart , etc.

A PICTURE IS WORTH A THOUSAND WORDS.





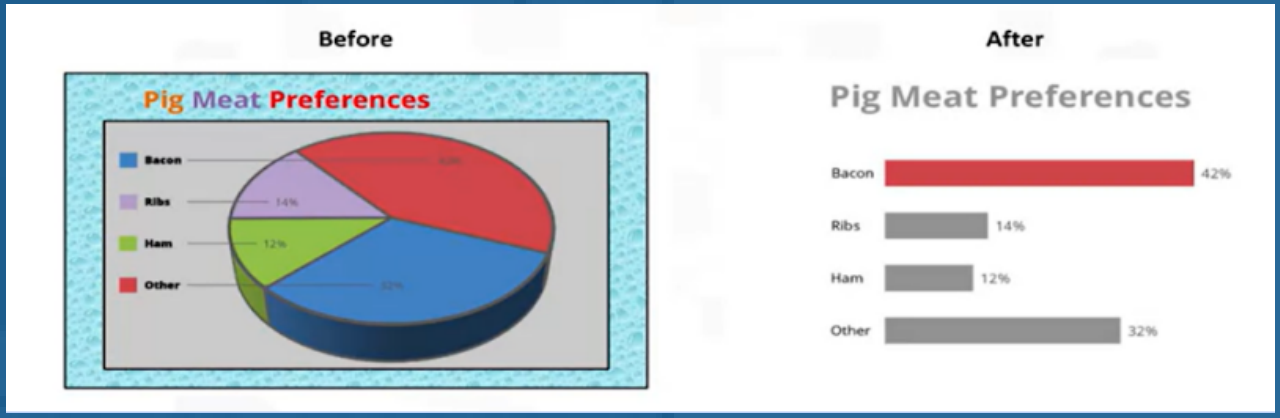
WHY NEED VISUALS?

- Human is a visual creature.
- For exploratory analysis
- Better data communication
- Shared unbiased representation of Data

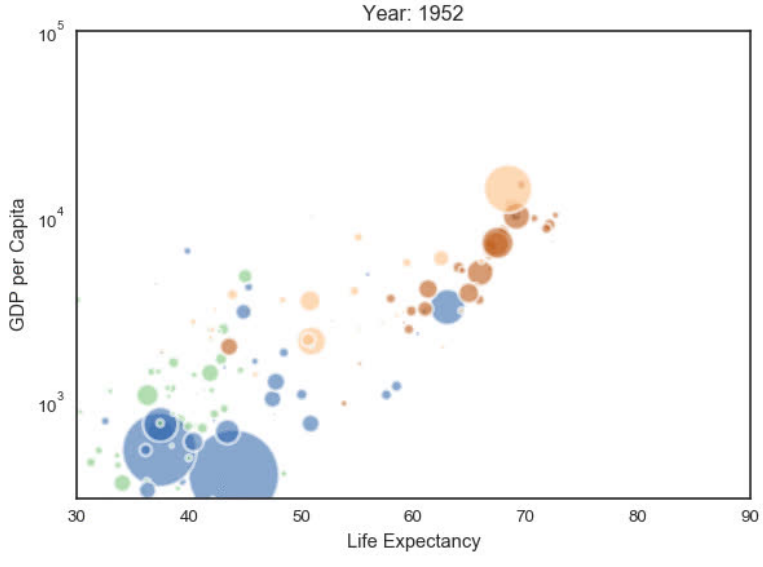
Best Practice

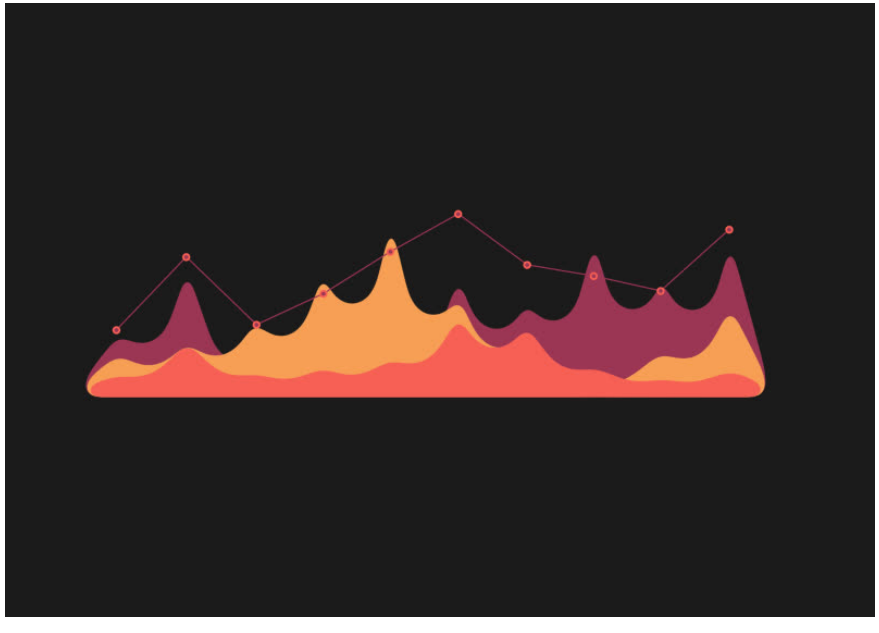
While creating a visuals , always remember -

1. Less is more effective
2. Less is more attractive
3. Less is more impactive



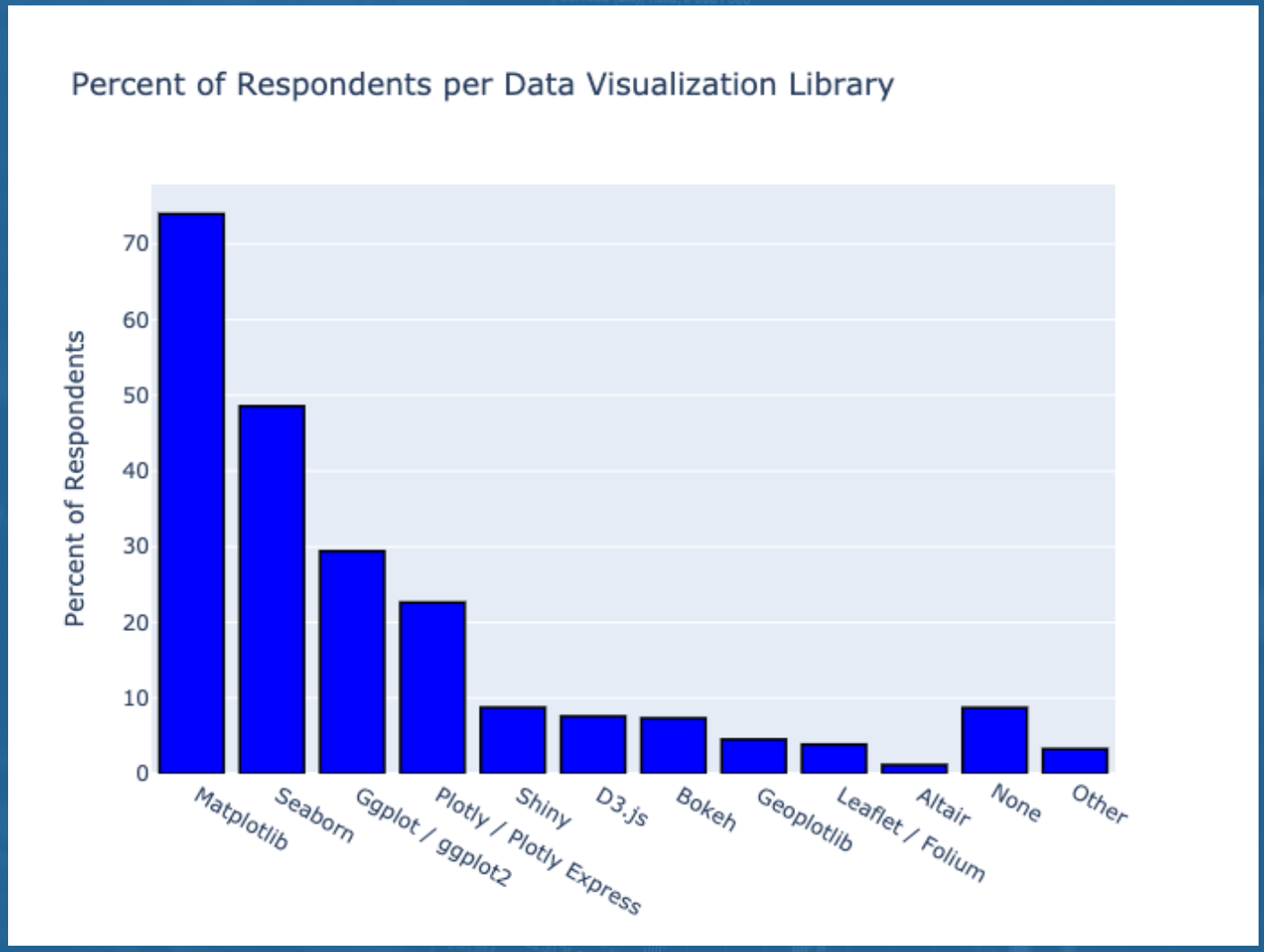
With the increasing size of typical 2D and 3D data, efficient computational methods are becoming increasingly crucial for achieving desired levels of interactivity.





Python Common Famous Libraries for Visualization

- Matplotlib
- Plotly
- Seaborn
- GGplot
- Altair
- Bokeh
- Pygal
- Geoplotlib

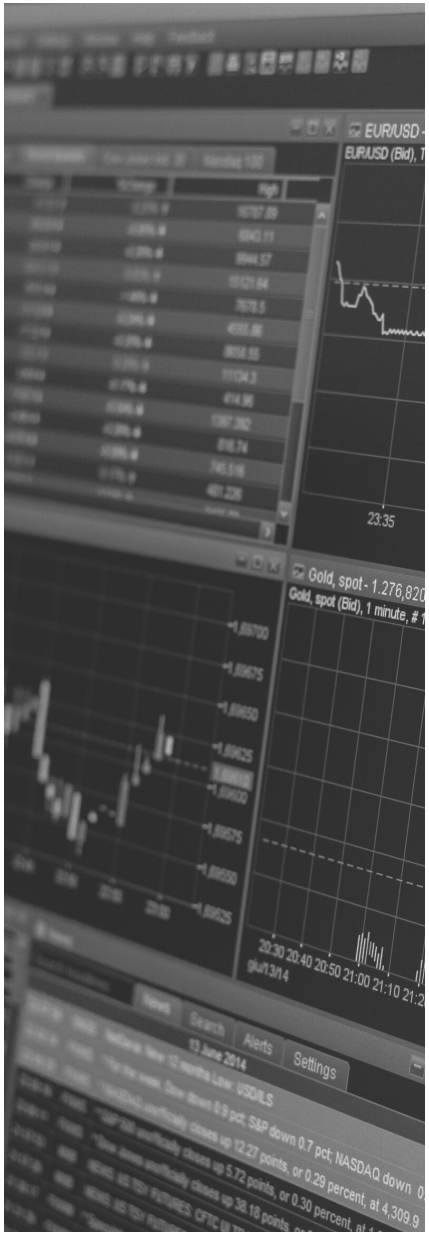


FAMOUS ONES -MATPLOTLIB &SEABORN

Before knowing how to plot , we need to know about DATA

Exploratory Data Analysis

Exploratory data analysis is a way to better understand your data which helps in further Data preprocessing and data visualization is key, making the exploratory data analysis process streamline and easily analyzing data using wonderful plots and charts.



EDA USAGE

1. Creating Hypotheses, testing various business assumptions while dealing with any Machine learning problem statement is very important and this is what EDA helps to accomplish.

1. To discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

EXPLORATORY DATA ANALYSIS IS CROSS-CLASSIFIED IN TWO DIFFERENT WAYS WHERE EACH METHOD IS EITHER GRAPHICAL OR NON-GRAPHICAL. AND THEN, EACH METHOD IS EITHER UNIVARIATE, BIVARIATE OR MULTIVARIATE.

- ## Univariates

  When we use a single feature to analyze its properties. Eg- heights of students of a class.

- ## Bivariates

  When we compare the data between exactly 2 features . Eg- find out average SAT score and age of students.

- ## Multivariates

  Comparing more than 2 variables. Eg- A doctor collecting cholesterol, blood pressure,and weight.

Matplotlib is **a low level plotting library available for the Python programming language as a component of NumPy, a big data numerical handling resource**. Matplotlib uses an object oriented API to embed plots in Python applications.

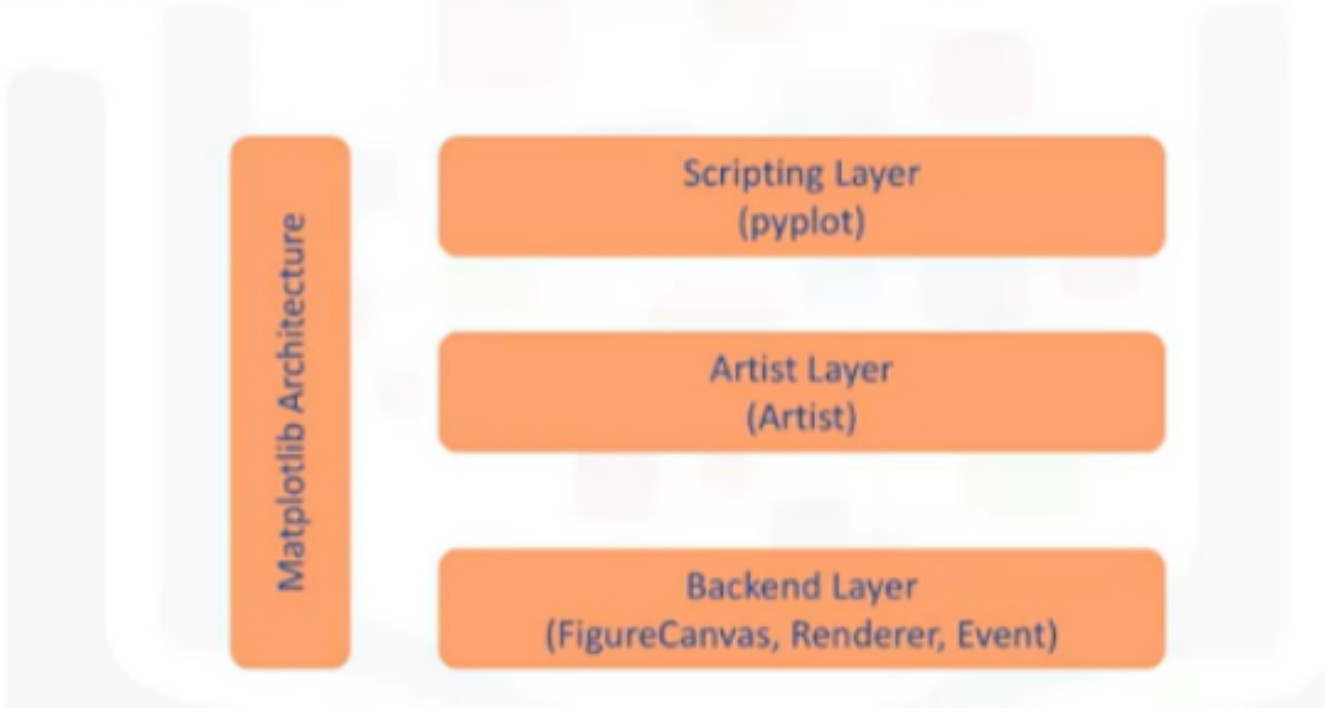
MATPLOTLIB WAS CREATED BY JOHN D. HUNTER.

IT WAS ORIGINALLY DEVELOPED AS AN ELECTROCORTICOGRAPHY (ECOG)VISUALIZATION TOOL.

Univariate methods	Bivariate methods	Multivariate methods
Frequency distributions	Cross tabulations	Conditional tables
	Scattergrams	Partial rank order correlation
	Regression	Multiple and partial correlation
	Correlation	Multiple and partial regression
	Comparison of means	Path analysis

MATPLOTLIB

Matplotlib Architecture



1. Backend Layer

Back-end layer of matplotlib provides the implementations of **three abstract** interface classes.

FigureCanvas:
Provides area onto which figure is drawn.
Example: for drawing in real world, we need paper onto which we can draw.
FigureCanvas is similar to paper.

matplotlib.backend_bases.FigureCanvas

RENDERER:

IT DOES DRAWING ON FIGURECANVAS.INSTANCE OF RENDERER KNOWS HOW TO DRAW ON THE FIGURECANVAS.

EXAMPLE: JUST LIKE WE NEED PAINTBRUSH , PEN OR PENCIL TO DRAW ON PAPER , WE USE RENDERER OBJECT TO DRAW ON FIGURECANVAS.

MATPLOTLIB.BACKEND_BASES.RENDERER

EVENT:

INSTANCE OF EVENT HANDLES USER INPUT LIKE KEYSTROKES, MOUSE CLICKS.

MATPLOTLIB.BACKEND_BASES.EVENT

2. Artist Layer

- Artist layer comprised of one object known as Artist.
- FigureCanvas from back-end layer is paper, artist object knows how to use Renderer object to draw on canvas.
- Everything we see on matplotlib figure is an instance of Artist.
- Example: title,lines,labels,images etc.

There are two types of Artist object:

1. **PRIMITIVE :** LINE2D, RECTANGLE, CIRCLE , TEXT.
2. **COMPOSITE:** AXIS,AXES,TICK, AND FIGURE.

EACH COMPOSITE ARTIST MAY CONTAIN OTHER COMPOSITE ARTIST AS WELL AS PRIMITIVE ARTIST.

ARTIST LAYER IS BEST FOR THE PROGRAMMERS AND APPROPRIATE PARADIGM WHEN WRITING A WEB APPLICATION SERVER, UI APPLICATION.

Scripting layer:
Artist layer is syntactically heavy for the everyday purposes, especially for exploratory work . Matplotlib provides lighter scripting interface to simplify common task .

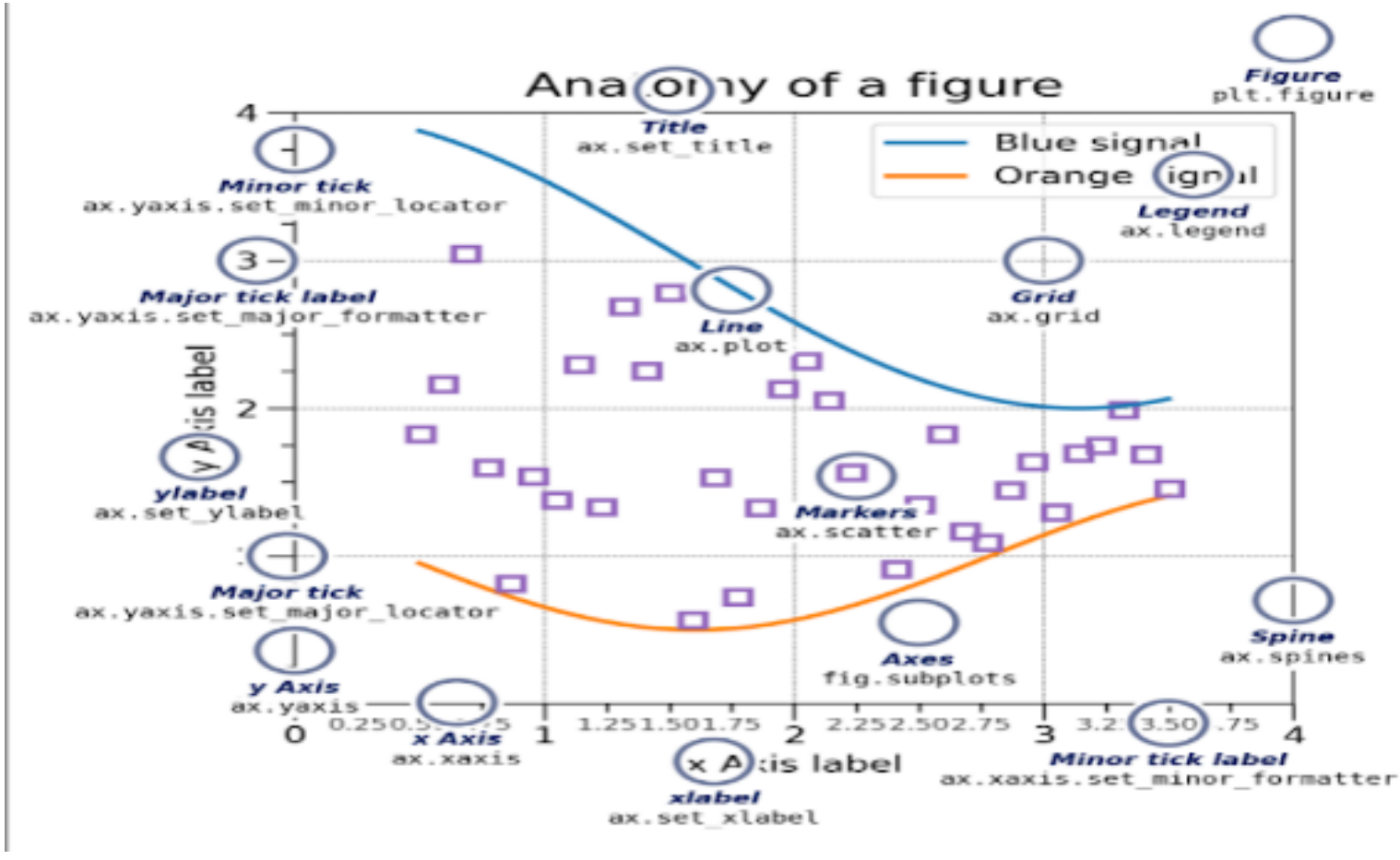
Scripting layer composed mainly of pyplot, a lighter interface than Artist layer .

matplotlib.pyplot

3. Scripting Layer

Parts of FIgure :

The components of a Matplotlib Figure



Types of Matplotlib in PYTHON

- BAR GRAPH,
- HISTOGRAM,
- SCATTERPLOT,
- AREA PLOT,
- BOX PLOT,
- PIE PLOT, ETC.

BAR GRAPH

Used in **data comparison** where we can measure the changes over a period of time. It can be represented ***horizontally or vertically***.

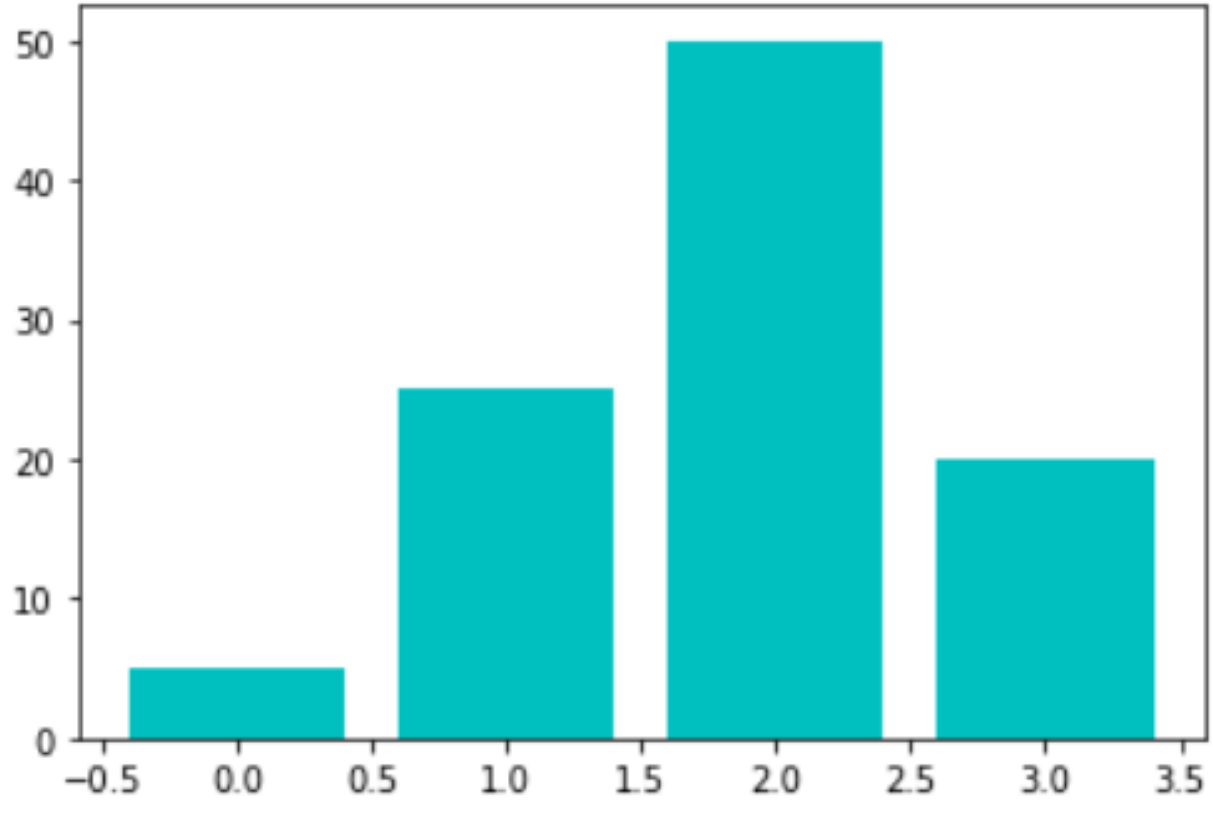
Longer the bar it has the greater the value it contains.

plt.bar(x,height,width,bottom,align)

x: representing the coordinates of the x-axis
height: the height of the bars
width: width of the bars. Its default value is 0.8
bottom: It's optional. It is a y-coordinate of the bar its default value is None
Align: center, edge its default value is center

[n [1]:

```
import matplotlib.pyplot
import matplotlib.pyplot as plt
data= [5. , 25. , 50. , 20.]
plt.bar(range(len(data)), data,color='c')
plt.show()
```



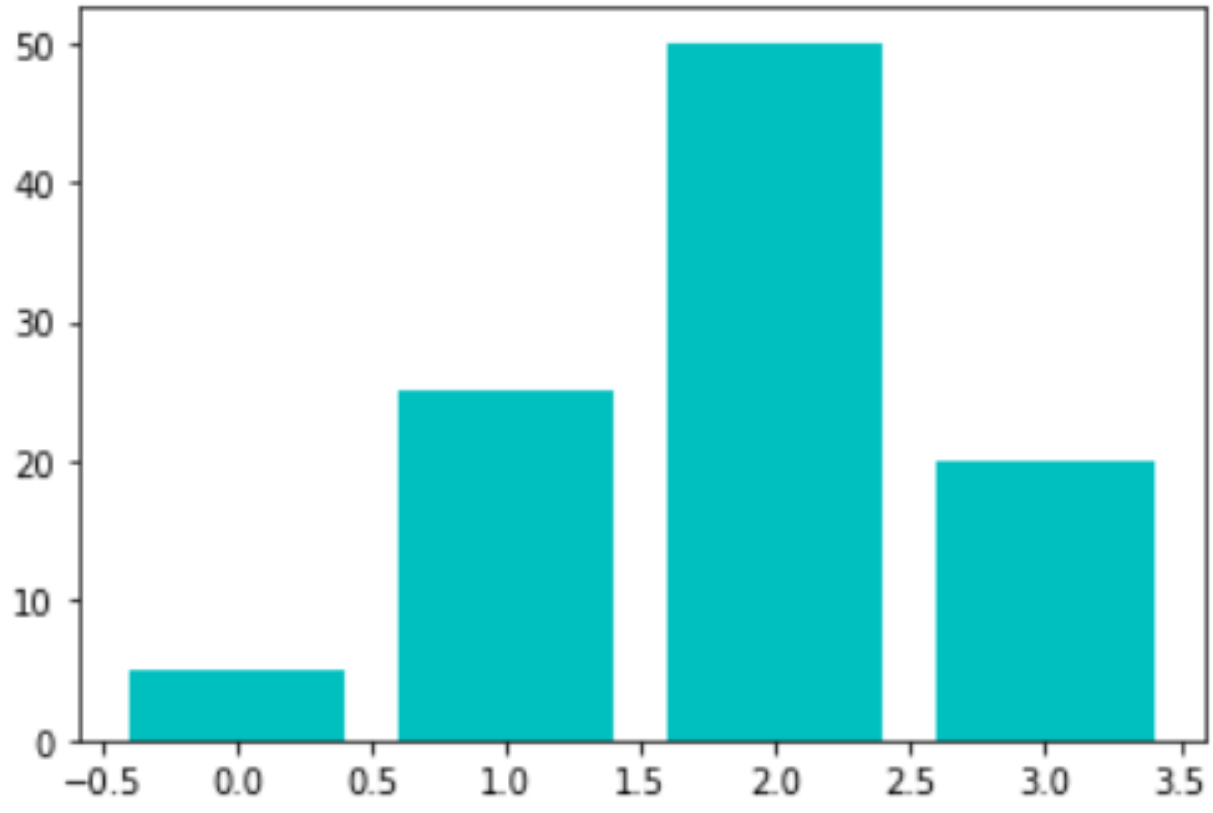
HISTOGRAM GRAPH

THE MOST COMMON GRAPH FOR DISPLAYING FREQUENCY DISTRIBUTIONS IS A HISTOGRAM. TO CREATE A HISTOGRAM THE FIRST STEP IS TO CREATE A BIN OF RANGES , THEN DISTRIBUTE THE WHOLE RANGE OF VALUE INTO SERIES OF INTERVALS, AND COUNT THE VALUE WHICH WILL FALL IN THE GIVEN INTERVAL.

x: x-coordinate or sequence of the array
bins: integer value for the number of bins wanted in the graph
range: the lower and upper range of bins
density: optional parameter that contains boolean values
histtype: optional parameter used to create different types of histograms like :-bar, bar stacked, step, step filled and the default is a bar

plt.hist() function used for plotting the histograms which will take various arguments like data, bins, color, etc.

In [1]:
```
import matplotlib.pyplot
import matplotlib.pyplot as plt
data= [5. , 25. , 50. , 20.]
plt.bar(range(len(data)), data,color='c')
plt.show()
```



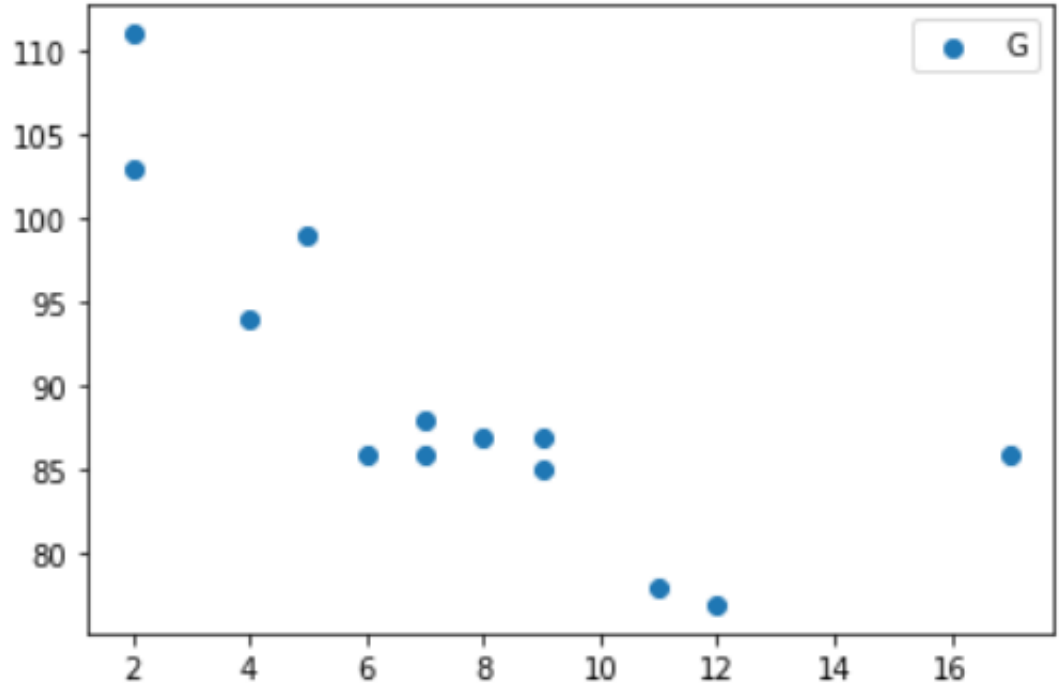
Scatter Plot

The scatter plots are preferred while comparing the data variables to determine the relationship between dependant and independent variables.

The data is displayed as a collection of points, each having the value of one variable which determines the position on the horizontal axis and the value of other variable determines the position on the vertical axis.

The scatter() method in the Matplotlib library is used for plotting.

```
]: #create the x and y axis coordinates
   x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
   y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
   plt.scatter(x, y)
   plt.legend('G')
   plt.show()
```



The area plots were also called as stack plots. It is quite similar to the line plots. Area plots are used in tracking the changes over time for two or more related groups that make one whole category.

fill_between() function is used to plot the area chart.
Parameter:
x,y represent the x and y coordinates of the plot. This will take an array of length n.

Interpolate is a boolean value and is optional. If true, interpolate between the two lines to find the precise point of intersection.

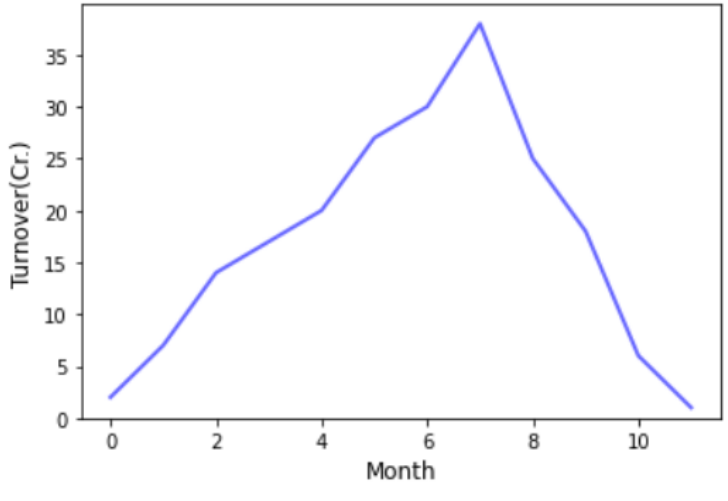
**kwargs: alpha, color, facecolor, edge colur, linewidth.

Area Plot

Fill the area in a line plot by using fill_between() for the area chart.

plt.fill_between(x,y, color="teal", alpha=0.5)

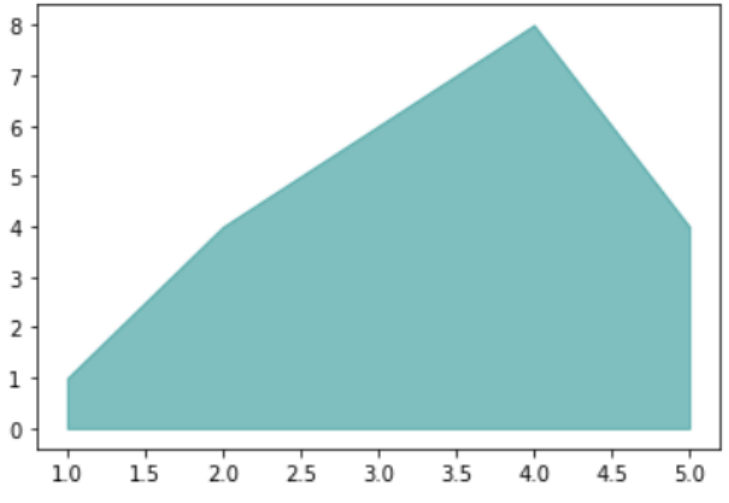
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
y = [2, 7, 14, 17, 20, 27, 30, 38, 25, 18, 6, 1]
#plot the line of the given data
plt.plot(np.arange(12),y, color="blue", alpha=0.6, linewidth=2)
#decorate thw plot by giving the labels
plt.xlabel('Month', size=12)
plt.ylabel('Turnover(Cr.)', size=12) #set y axis start with zero
plt.ylim(bottom=0)
plt.show()
```




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

# Create data
x=range(1,6)
y=[1,4,6,8,4]

# Area plot
plt.fill_between(x, y,color="teal",alpha=0.5)
plt.show()
```



BOX PLOT

A Box plot is used to show the summary of the whole dataset or all the numeric values in the dataset. The summary contains minimum, first quartile, median, third quartile, and maximum. Also, the median is present between the first and third quartile. Here x-axis contains the data values and y coordinates show the frequency distribution.

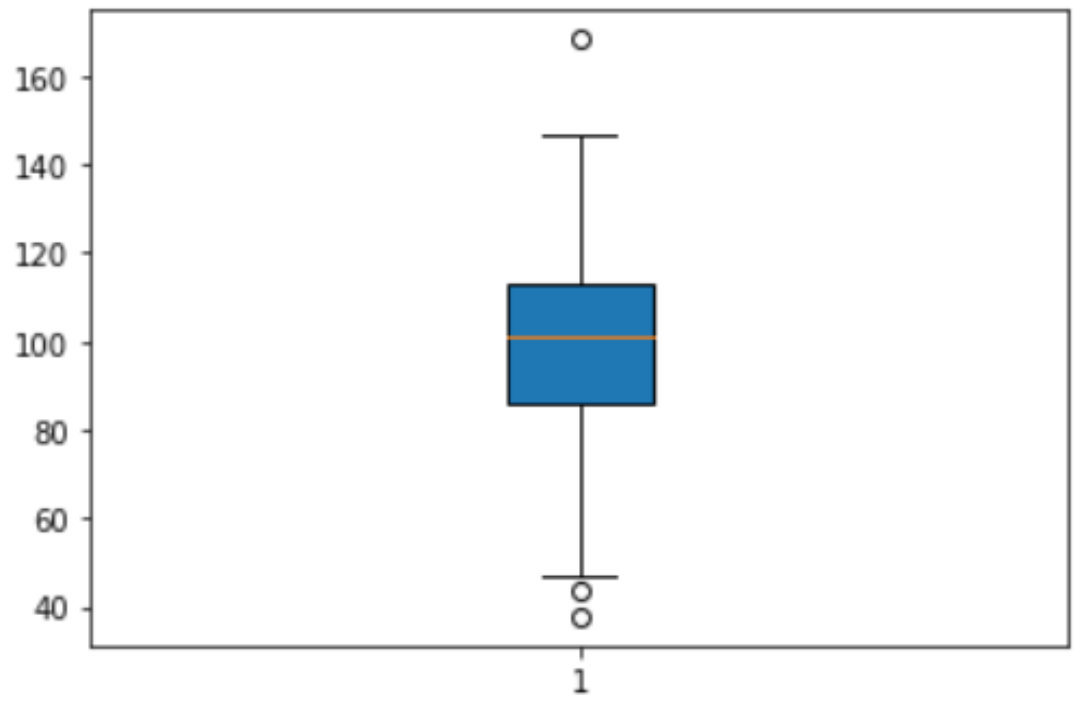
Parameters used in box plots are as follows:

Data: NumPy array
Vert: It will take boolean values i.e true or false for the vertical and horizontal plot default is True
width: This will take array and sets of the width of boxes, Optional parameters
Patch_artist: It is used to fill the boxes with color and its default value is false
labels: Array of strings which is used to set the labels of the dataset

Syntax -

matplotlib.pyplot.boxplot(data, notch=None, vert=None, patch_artist=None, widths=None)

```
#create the random values by using numpy
values= np.random.normal(100, 20, 300)
#creating the plot by boxplot() function which is avilable in matplotlib
plt.boxplot(values,patch_artist=True,vert=True)
plt.show()
```



Pie Chart

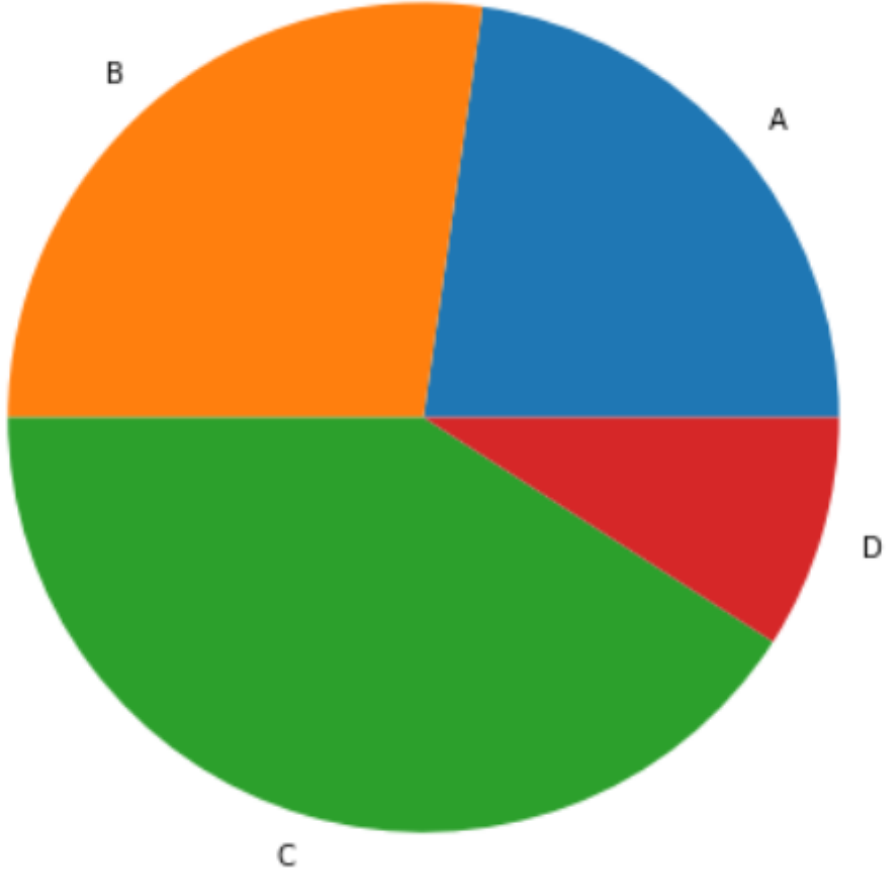
A pie chart is a circular graph which is divided into segments or slices of pie. It is used to represent the percentage or proportional data where each slice of pie represents a category.

x: Sequence of an array

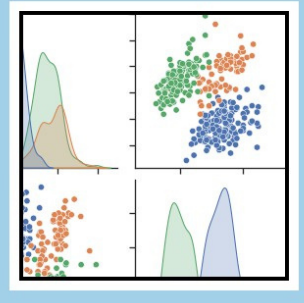
labels: List of strings which will be the name of each slice in the pie chart

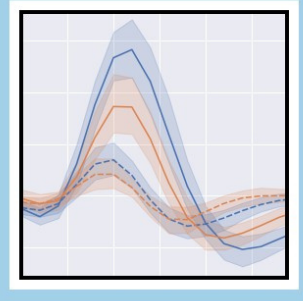
Autopct: It is used to label the wedges with numeric values. The labels will be placed inside the wedges. Its format is %1.2f%

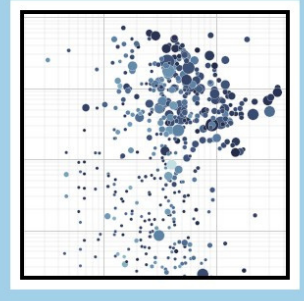
```
#define the figure size
plt.figure(figsize=(7,7))
x = [25,30,45,10]
#labels of the pie chart
labels = ['A','B','C','D']
plt.pie(x, labels=labels)
plt.show()
```

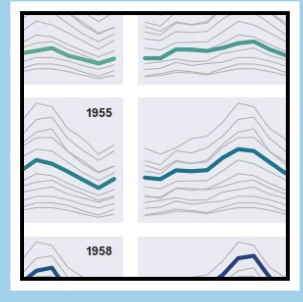


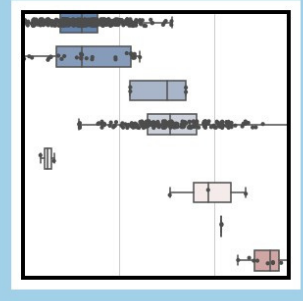
seaborn











Seaborn is a library for making statistical graphics in Python.
It is complimentary to Matplotlib.

One of these hard things or frustrations had to do with the default Matplotlib parameters. Seaborn works with different parameters, which undoubtedly speaks to those users that don't use the default looks of the Matplotlib plots.

FUNCTIONALITY OF SEABORN -

- Visuals that need ~20 lines of code using matplotlib to be created, with seaborn the number of lines of code is reduced by 5-fold.
- Helps to explore and understand data.
- Support high-level abstractions for multi-plot grids.
- Its dataset-oriented API to determine the relationship between variable.
- Automatic estimation and plotting of linear regression plots.
- Visualize univariate and bivariate distribution.

BEFORE WE CONTINUE, NOTE THAT SEABORN PLOTS BELONG TO ONE OF TWO GROUPS.

- AXES-LEVEL PLOTS – THESE MIMIC MATPLOTLIB PLOTS AND CAN BE BUNDLED INTO SUBPLOTS USING THE AX PARAMETER. THEY RETURN AN AXES OBJECT AND USE NORMAL MATPLOTLIB FUNCTIONS TO STYLE.

- FIGURE-LEVEL PLOTS – THESE PROVIDE A WRAPPER AROUND AXES PLOTS AND CAN ONLY CREATE MEANINGFUL AND RELATED SUBPLOTS BECAUSE THEY CONTROL THE ENTIRE FIGURE. THEY RETURN EITHER FACETGRID, PAIRGRID, OR JOINTGRID OBJECTS AND DO NOT SUPPORT THE AX PARAMETER. THEY USE DIFFERENT STYLING AND CUSTOMIZATION INPUTS.

Famous Seaborn Plots

- **PAIR PLOT**
- **HEAT PLOTS**
- **SCATTER PLOTS**
- **REG PLOTS**
- **BAR PLOTS**
- **CAT PLOTS**
- **DISTRIBUTION PLOTS(**JOIN PLOT, DISTPLOT, PAIRPLOT,RUGPLOT)

Pair Plot

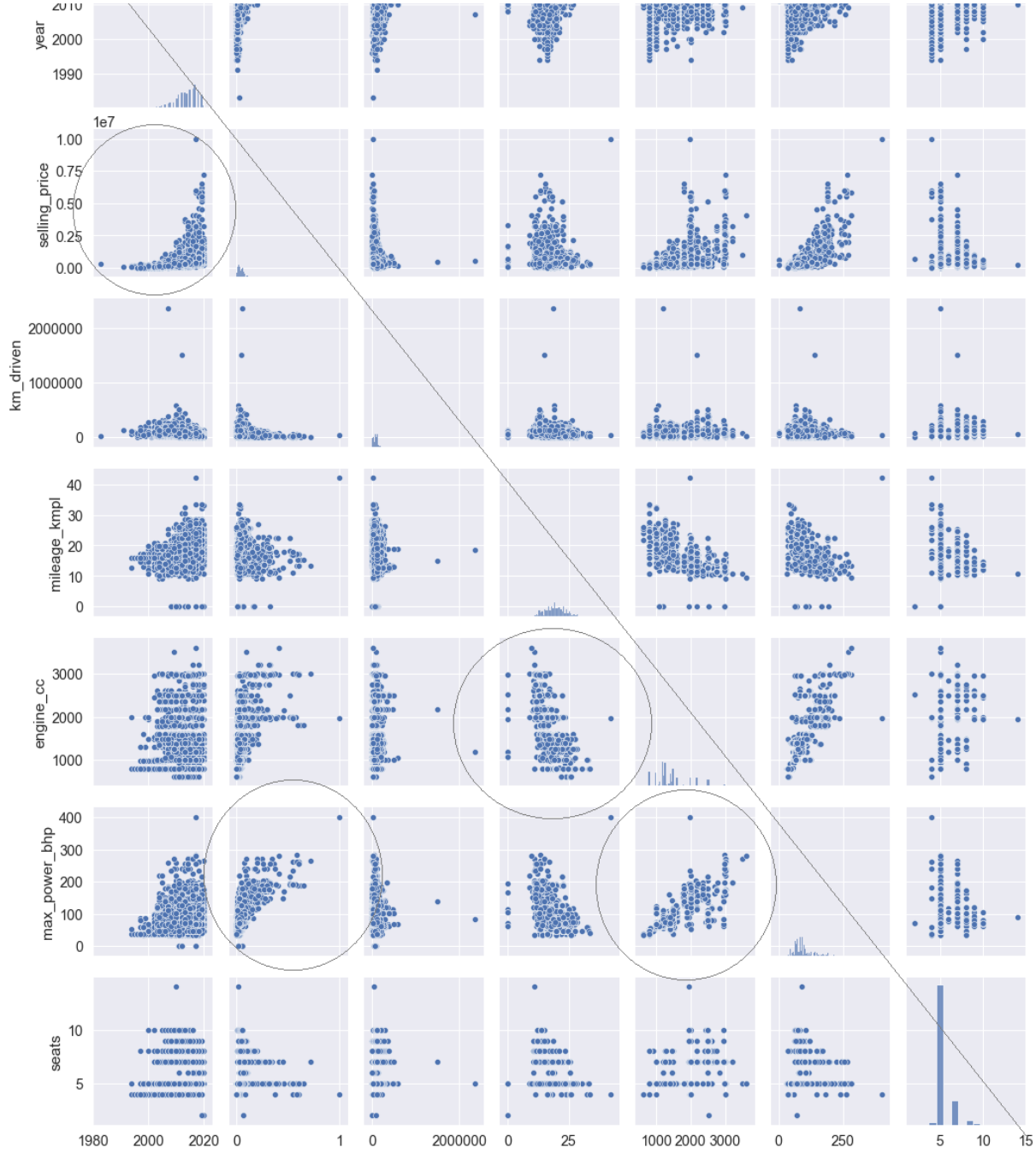
A pair plot creates a grid of scatter plots to compare the distribution of pairs of numeric variables. It also features a histogram for each feature in the diagonal boxes.

Functions to use:

sns.pairplot() – figure-level plot
The kind parameter changes the type of bivariate plots created with kind= 'scatter' (default), 'kde', 'hist'or 'reg'.

Two columns per grid

(Bivariate) sns.pairplot(cars);



Three columns (multivariate): two numeric and one categorical.We can add a third variable that segments the scatter plots by color using the parameter

hue='cat_col'.

- **sns.pairplot(**
- **data=cars,**
- **aspect=.85,**
- **hue='transmission');**

Heat Plot

A HEAT MAP IS A COLOR-CODED GRAPHICAL REPRESENTATION OF VALUES IN A GRID. IT'S AN IDEAL PLOT TO FOLLOW A PAIR PLOT BECAUSE THE PLOTTED VALUES REPRESENT THE CORRELATION COEFFICIENTS OF THE PAIRS THAT SHOW THE MEASURE OF THE LINEAR RELATIONSHIPS.

IN SHORT, A PAIR PLOT SHOWS THE INTUITIVE TRENDS OF THE DATA, WHILE A HEAT MAP PLOTS THE ACTUAL CORRELATION VALUES USING COLOR.

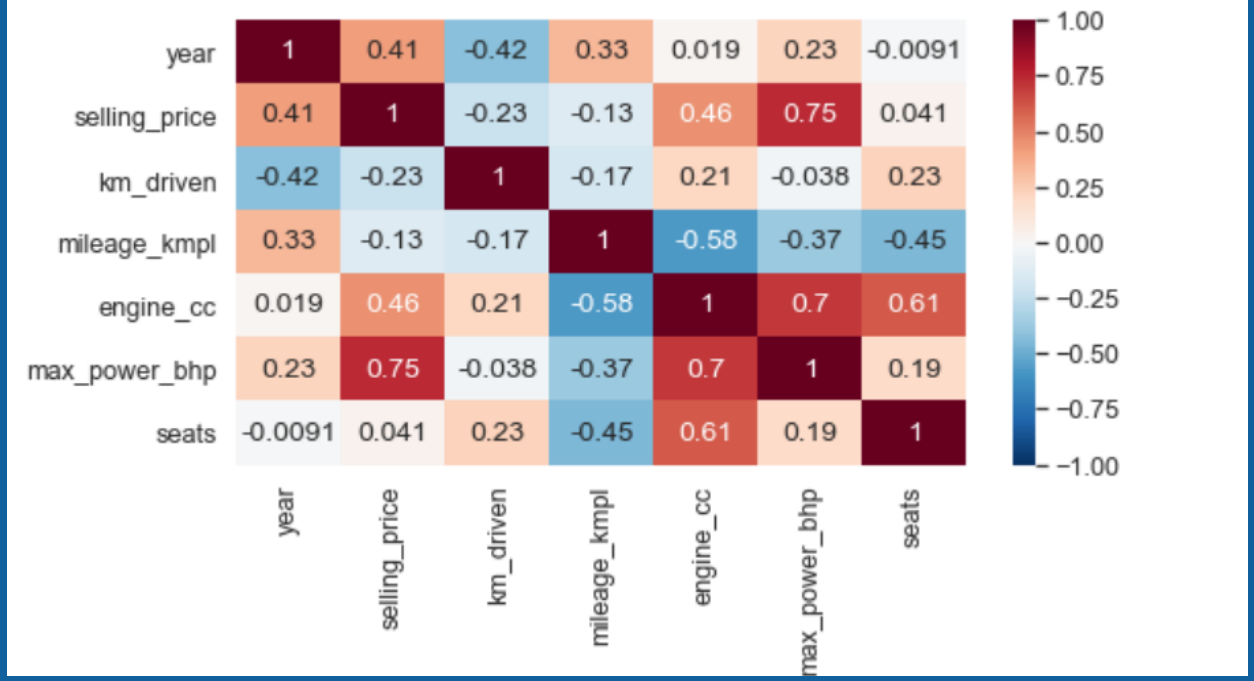
Functions to use:

sns.heatmap() —axes-level plot

First use df.corr() to get a table with the correlation coefficients. That table is also known as a correlation matrix.

sns.heatmap() - cmap='RdBu_r' sets the color scheme, annot=True draws the values inside the cells, and vmin and vmax ensures the color codes start at -1 to 1.

```
sns.set(font_scale=1.15)
plt.figure(figsize=(8,4))
sns.heatmap(cars.corr(),
            cmap='RdBu_r',
            annot=True,
            vmin=-1, vmax=1);
```



Scatter Plot

A scatter plot shows the relationship between two numeric features by using dots to visualize how these variables move together.

Functions to use:

sns.scatterplot() — axes-level plot
sns.relplot(kind='line') — figure-level

FUNCTIONS WITH REGRESSION LINE;

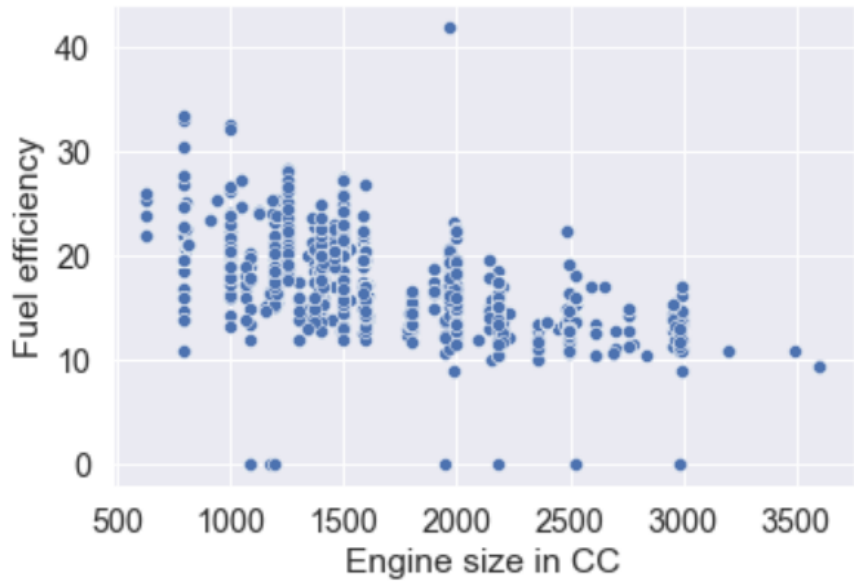
- SNS.REGPLOT() – AXES-LEVEL
- SNS.LMPLOT() – FIGURE-LEVEL

TWO NUMERIC COLUMNS (BIVARIATE)

SNS.SCATTERPLOT(X='NUM_COL1', Y='NUM_COL2', DATA=DF)

```
sns.set(font_scale=1.3)
sns.scatterplot(x = 'engine_cc',
                y = 'mileage_kmpl',
                data=cars)
plt.xlabel('Engine size in CC')
plt.ylabel('Fuel efficiency')
```

```
Text(0, 0.5, 'Fuel efficiency')
```



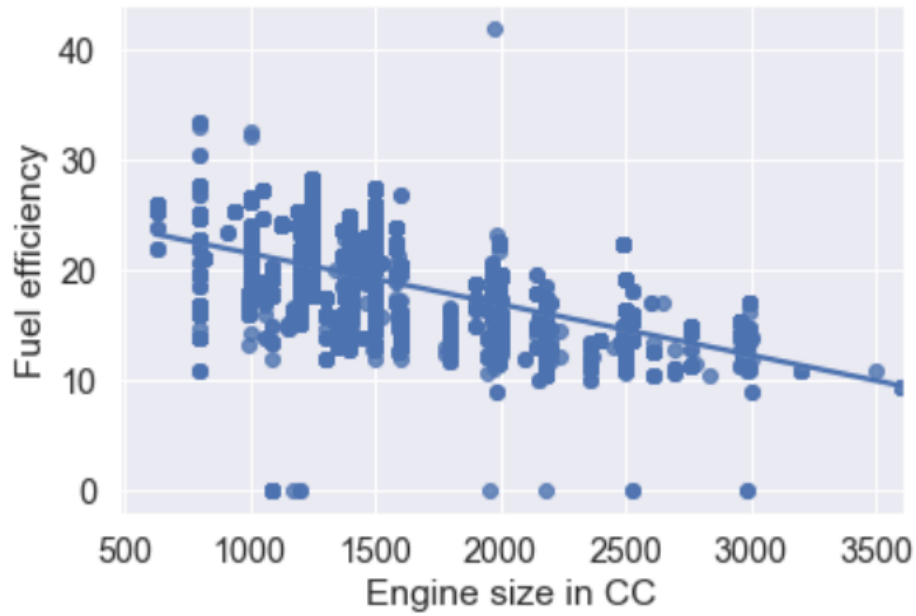
Reg plot

A **reg plot** draws a scatter plot with a ***regression line*** showing the trend of the data.

Regplot (axes level) and lmplot (figure-level) is a scatter plot that gives us a trend line, also called a regression line.

The shade around the line represents the dispersion on points in those areas.

```
sns.regplot(x = 'engine_cc',
            y = 'mileage_kmpl',
            data=cars)
plt.xlabel('Engine size in CC')
plt.ylabel('Fuel efficiency');
```



BAR PLOT

THE BAR CHART USES BARS OF DIFFERENT HEIGHTS TO COMPARE THE DISTRIBUTION OF A NUMERIC VARIABLE BETWEEN GROUPS OF A CATEGORICAL VARIABLE.

BY DEFAULT, BAR HEIGHTS ARE ESTIMATED USING THE "MEAN". THE *ESTIMATOR* PARAMETER CHANGES THIS AGGREGATION FUNCTION BY USING PYTHON'S INBUILT FUNCTIONS SUCH AS *ESTIMATOR=MAX* OR LEN, OR NUMPY FUNCTIONS LIKE NP.MAX AND NP.MEDIAN.

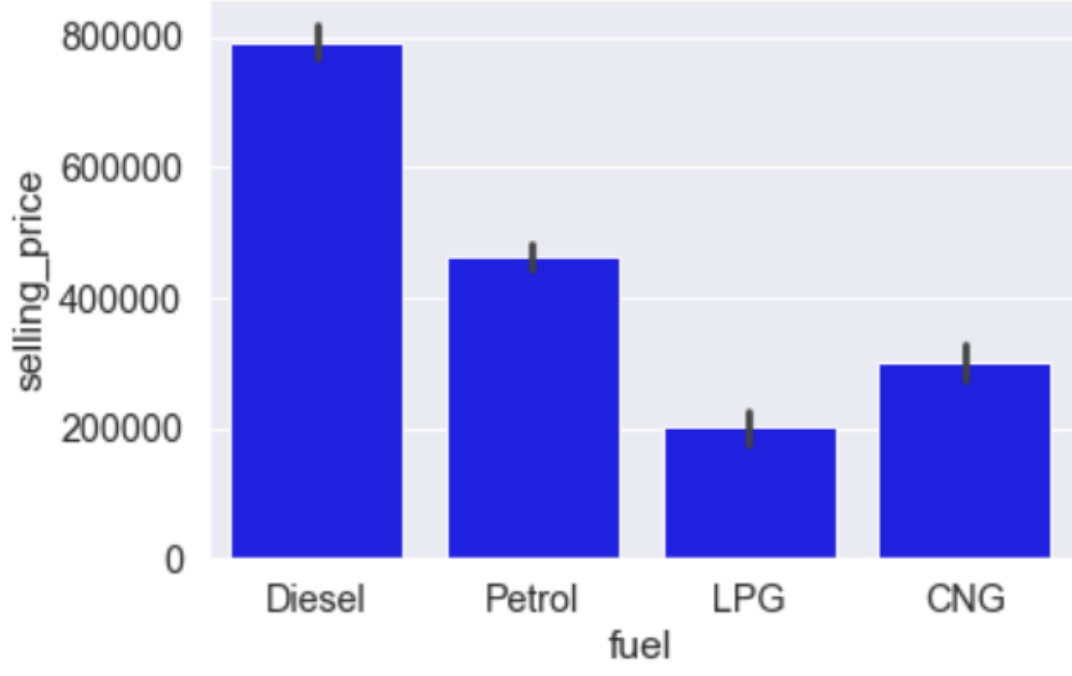
FUNCTIONS TO USE:

SNS.BARPLOT() – AXES-LEVEL PLOT
SNS.CATPLOT(KIND='BAR') – FIGURE-LEVEL PLOT

Two columns (bivariate): numeric and categorical

sns.barplot(x='cat_col', y='num_col', data=df)

```
sns.barplot(x='fuel',
            y='selling_price',
            data=cars,
            color='blue');
```

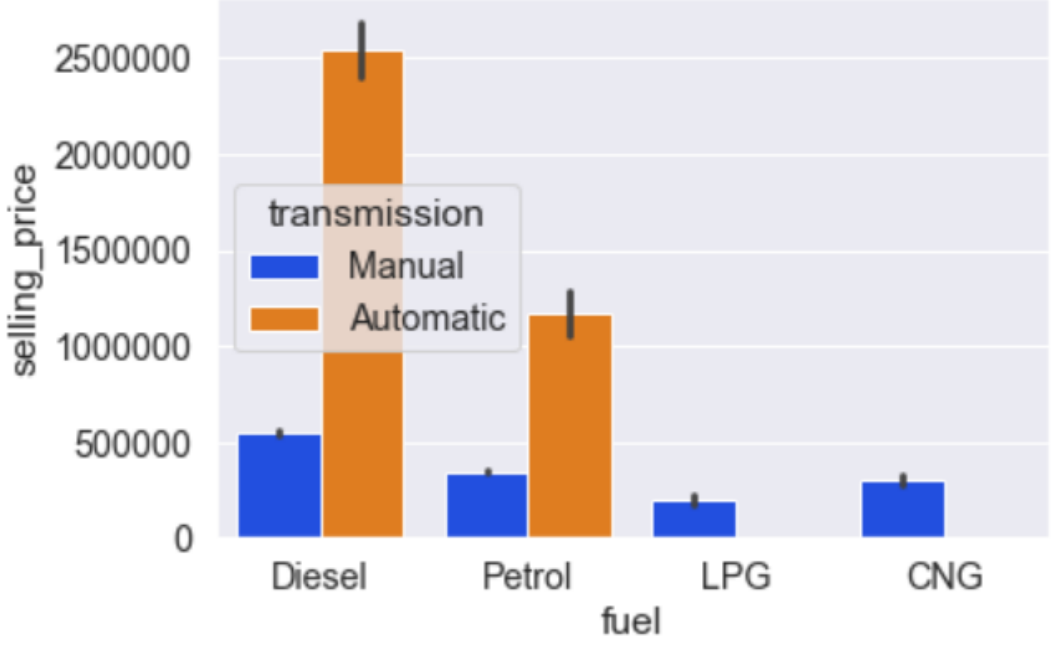


Three columns (multivariate): two categorical and one numeric.

sns.barplot(x, y, data, hue='cat_col2')

```
sns.barplot(x="fuel",
            y="selling_price",
            data=cars,
            palette='bright'
            hue="transmission",)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x85e2a49188>
```



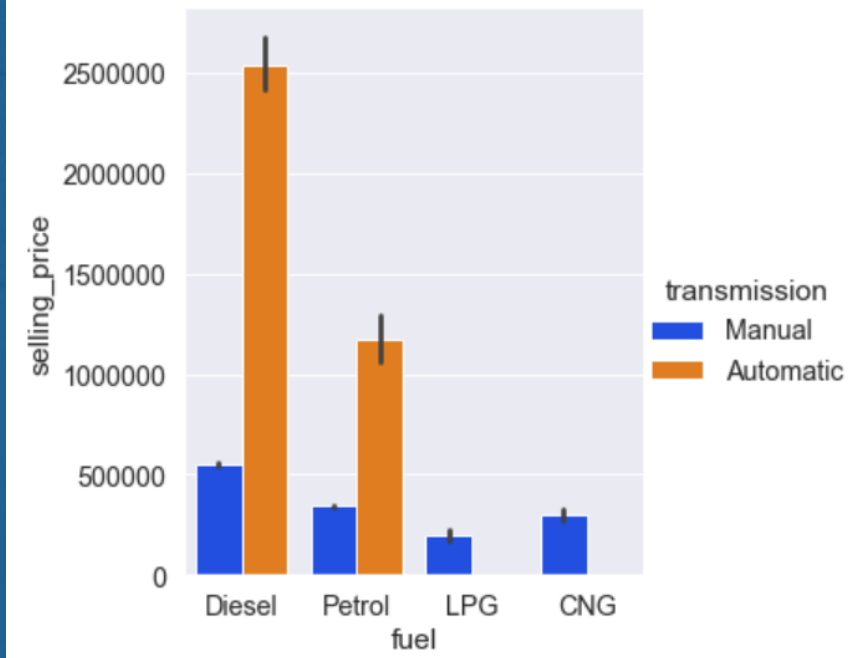
Cat Plot

SNS.CATPLOT(X, Y, DATA, KIND='BAR', HUE='CAT_COL')

A CATPLOT OR CATEGORICAL PLOT, USES THE **KIND** *PARAMETER TO SPECIFY WHAT CATEGORICAL PLOT TO DRAW WITH OPTIONS BEING* ***'STRIP'(DEFAULT), 'SWARM', 'BOX', 'VIOLIN', 'BOXEN', 'POINT' AND 'BAR'.***

```
sns.catplot(x="fuel",
            y="selling_price",
            data=cars,
            palette='bright',
            kind="bar",
            hue="transmission")
```

```
<seaborn.axisgrid.FacetGrid at 0xc04a723088>
```



There are many other plotting technique to represent your data in vivid visuals with so much professional and approachable look.

I.e-

- **Line plot**
- **Point plot**
- **Violine plot**
- **Strip Plot , etc.**

matplotlib

seaborn

Features	Matplotlib	Seaborn
Functionality	It is utilized for making basic graphs. Datasets are visualised with the help of bargraphs, histograms, piecharts, scatter plots, lines and so on.	Seaborn contains a number of patterns and plots for data visualization. It uses fascinating themes. It helps in compiling whole data into a single plot. It also provides distribution of data.
Syntax	It uses comparatively complex and lengthy syntax. Example: Syntax for bargraph- matplotlib.pyplot.bar(x_axis, y_axis).	It uses comparatively simple syntax which is easier to learn and understand. Example: Syntax for bargraph- seaborn.barplot(x_axis, y_axis).
Dealing Multiple Figures	We can open and use multiple figures simultaneously. However they are closed distinctly. Syntax to close one figure at a time: matplotlib.pyplot.close(). Syntax to close all the figures: matplotlib.pyplot.close("all")	Seaborn sets time for the creation of each figure. However, it may lead to (OOM) out of memory issues

Features	Matplotlib	Seaborn
Visualization	Matplotlib is well connected with Numpy and Pandas and acts as a graphics package for data visualization in python. Pyplot provides similar features and syntax as in MATLAB. Therefore, MATLAB users can easily study it.	Seaborn is more comfortable in handling Pandas data frames. It uses basic sets of methods to provide beautiful graphics in python.
Pliability	Matplotlib is a highly customized and robust	Seaborn avoids overlapping of plots with the help of its default themes
Use Cases	Matplotlib plots various graphs using Pandas and Numpy	Seaborn is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas for plotting graphs

Thank you!