Figures

Refers to the whole figure

Axis

An Axes refers to the actual plot in the figure. A figure can have multiple axes

Plot categories in seaborn

1.*Relational Plot*:Used to understand the relation between two variables

2.*Categorical Plot*:This plot deals with categorical variables and how they can be visualized

3.*Distribution Plot*:This plot is used for examining univariate and bivariate distributions

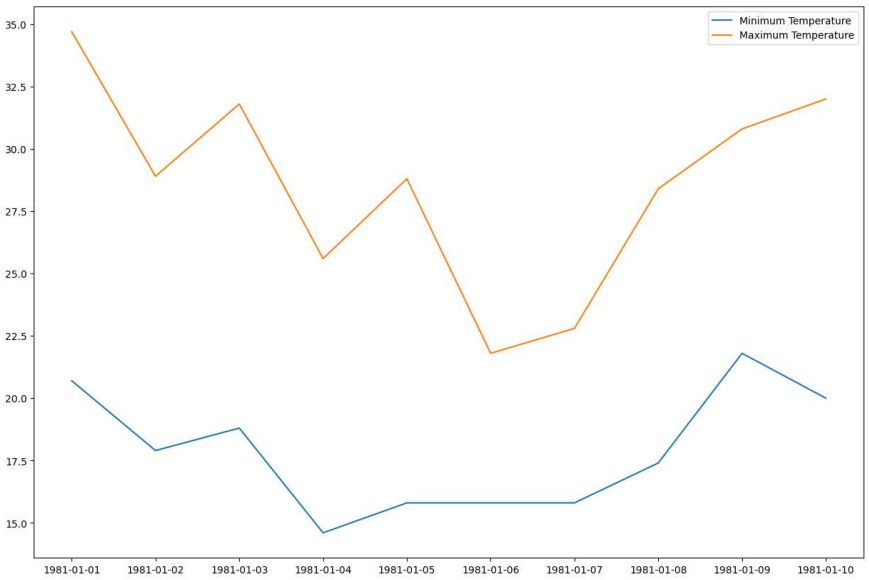
4.*Matrix Plot*:A matrix plot is an array of scatterplots

5.*Regression Plots*:The regression plots in seaborn are primarily intended to add a visual

```
#Import neccessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure

import seaborn as sns

%matplotlib inline
```

```
#Simple Plotting with Seaborn

#Data

dates=['1981-01-01','1981-01-02','1981-01-03','1981-01-04','1981-01-05','1981-01-06','1981-01-07','1981-01-08','1981-01-09','1981-01-

min_temp=[20.7,17.9,18.8,14.6,15.8,15.8,15.8,17.4,21.8,20.0]
max_temp=[34.7,28.9,31.8,25.6,28.8,21.8,22.8,28.4,30.8,32.0]

#Plotting
fig,axes=plt.subplots(nrows=1,ncols=1,figsize=(15,10))
axes.plot(dates,min_temp,label="Minimum Temperature")
axes.plot(dates,max_temp,label="Maximum Temperature")
axes.legend()
```

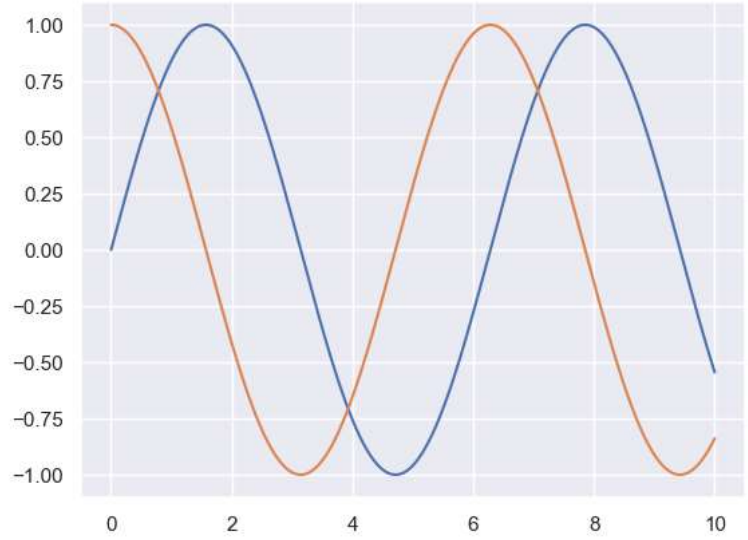
```
<matplotlib.legend.Legend at 0x237abc54d10>
```




```
1 #seaborn style  as the default matplotlib style
2 sns.set()
```

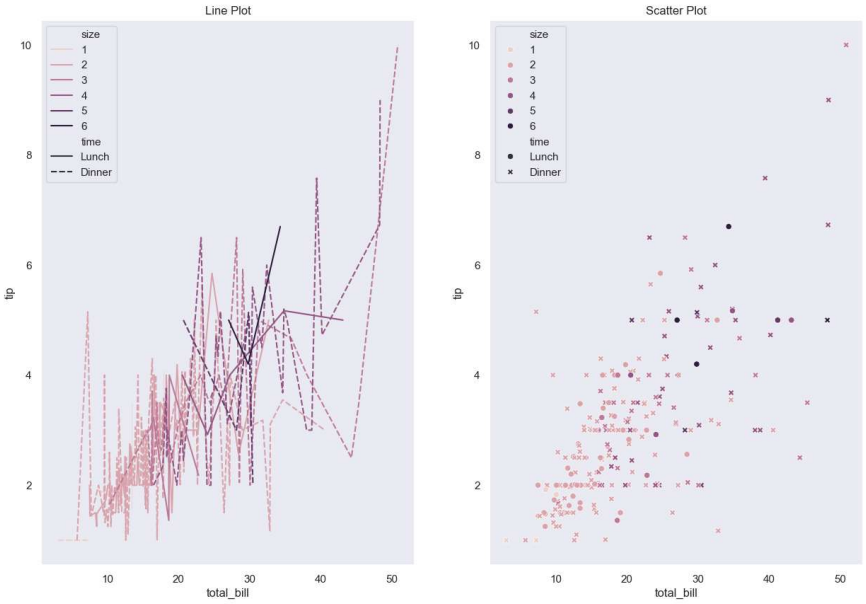
```
1 #Simple Since Plot
2 x=np.linspace(0,10,1000)
3 plt.plot(x,np.sin(x),x,np.cos(x))
```

```
[<matplotlib.lines.Line2D at 0x237abcafd50>,
 <matplotlib.lines.Line2D at 0x237ab9c1610>]
```



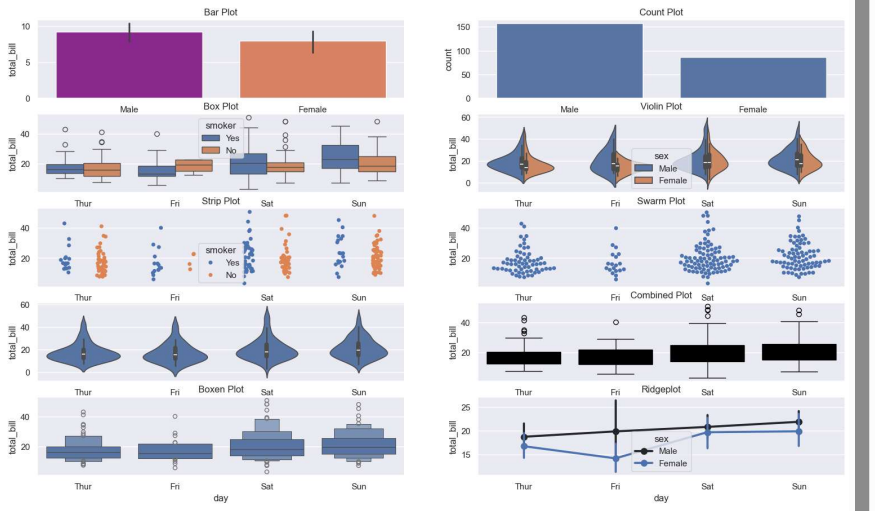
```
#1.Relational Plot

#Line Plot:Line plot is one of the most basic plot in seaborn library
#This plot is mainly used to visualize the data in form of some time series,i.e

sns.set(style="dark")
fig,ax=plt.subplots(ncols=2,nrows=1,figsize=(15,10))

#Loading data with seaborn
df=sns.load_dataset("tips")
print(df.head)

#Lineplot
sns.lineplot(x="total_bill",y="tip",hue="size",style="time",data=df,ax=ax[0]).set_title("Line Plot")

#Scatterplot
Sct_plt=sns.scatterplot(x="total_bill",y="tip",hue="size",style="time",data=df,ax=ax[1],).set_title("Scatter Plot")

#Saving the Plot
Sct_plt.figure.savefig('Scatter_plot1.png')
print('Plot Saved')
```

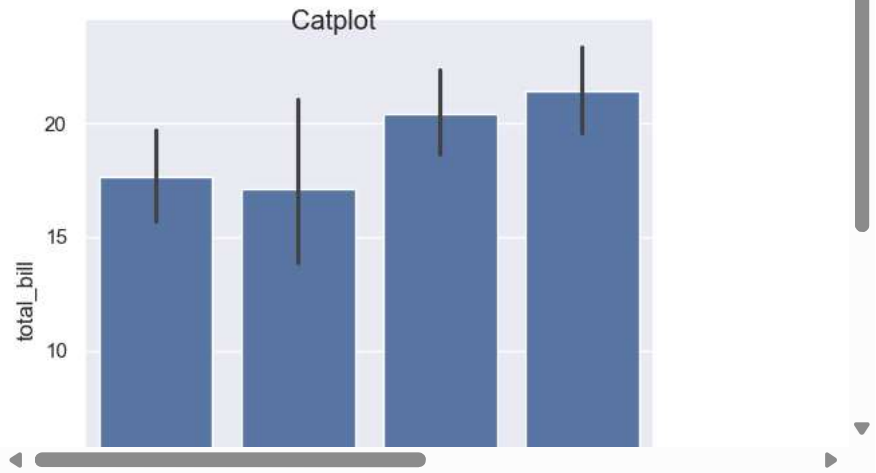
```
<bound method NDFrame.head of      total_bill   tip     sex smoker   day    time  siz
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          ...   ...     ...    ...   ...     ...   ...
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]>
Plot Saved
```



```
# 2.Categorical Plot
# Plot are basically used for visualizing the relationship between variables
# Variables can be either be completely numerical or a category like a group, class or division

sns.set_style('darkgrid')
fig, ax = plt.subplots(nrows=5,ncols=2)
fig.set_size_inches(18.5,10.5)

# Data
# 'tips' dataset contains information about people who probably had food at a restaurant
# whether or not they left a tip for the waiters,their gender, whether they smoke and so on.

df=sns.load_dataset('tips')

#barplot - basically used to aggregate the categorical data according to some

sns.barplot(x='sex',y='total_bill',data=df,palette='plasma',estimator=np.std,ax=ax[0,0]).set_title('Bar Plot')

#countplot - Counts the cateogires and returns a count of their occurences

sns.countplot(x='sex',data=df,ax=ax[0,1]).set_title('Count Plot')

#box plot- Also known as the Box and Whisker Plot
#It shows distribution of the Quantitive Data that represents the comparison

sns.boxplot(x='day',y='total_bill',data=df,hue='smoker',ax=ax[1,0]).set_title('Box Plot')

#Similar to the boxplot except that it provides a high
#Uses the kernel density estimation to give a better description about the data

sns.violinplot(x='day',y='total_bill',data=df,hue='sex',split=True,ax=ax[1,1]).set_title('Violin Plot')

#Stripplot - Scatter plot based on the category

sns.stripplot(x='day',y='total_bill',data=df,jitter=True,hue='smoker',dodge=True,ax=ax[2,0]).set_title('Strip Plot')

#Swarmplot - Similar to stripplot except

sns.swarmplot(x='day',y='total_bill',data=df,ax=ax[2,1]).set_title('Swarm Plot')
sns.violinplot(x='day',y='total_bill',data=df,ax=ax[3,0])
sns.boxplot(x='day',y='total_bill',data=df,color='black',ax=ax[3,1]).set_title('Combined Plot')

#Boxen Plot

sns.boxenplot(x='day',y='total_bill',color='b',scale='linear',data=df,ax=ax[4,0]).set_title('Boxen Plot')

#Ridgeplot

sns.pointplot(x='day',y='total_bill',color='b',hue='sex',data=df,ax=ax[4,1]).set_title('Ridgeplot')

catplot=sns.catplot(x='day',y='total_bill',data=df,kind='bar')
catplot.fig.suptitle('Catplot')

Sct_plt.figure.savefig('ALL DAV plots.png')
print("Plot Saved")
```

```
C:\Users\SpaceScientist\AppData\Local\Temp\ipykernel_13296\1902770095.py:17: Futur

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.

  sns.barplot(x='sex',y='total_bill',data=df,palette='plasma',estimator=np.std,ax=
C:\Users\SpaceScientist\AppData\Local\Temp\ipykernel_13296\1902770095.py:45: Futur

The `scale` parameter has been renamed to `width_method` and will be removed in v0
  sns.boxenplot(x='day',y='total_bill',color='b',scale='linear',data=df,ax=ax[4,0]
C:\Users\SpaceScientist\AppData\Local\Temp\ipykernel_13296\1902770095.py:49: Futur

Setting a gradient palette using color= is deprecated and will be removed in v0.14

  sns.pointplot(x='day',y='total_bill',color='b',hue='sex',data=df,ax=ax[4,1]).set_
Plot Saved
```





```
sns.set_style('whitegrid')
df = sns.load_dataset('iris')
print(df.head())
sns.distplot(df['petal_length'],kde=True,color='red',bins=30).set_title('Dist Plot')
jointgrid = sns.JointGrid(x='petal_length',y='petal_width',data=df)
jointgrid.plot_joint(sns.scatterplot)
jointgrid.plot_marginals(sns.displot)
g = sns.jointplot(x='petal_length',y='petal_width',data=df,kind='hex')
g.fig.suptitle("Joint Plot")
g = sns.pairplot(df,hue="species",palette='coolwarm')
g.fig.suptitle("Pair Plot - 1")
g.add_legend()
pairgrid = sns.PairGrid(data=df)
pairgrid = pairgrid.map_offdiag(sns.scatterplot)
pairgrid = pairgrid.map_diag(plt.hist)
pairgrid = sns.PairGrid(data=df)
pairgrid = pairgrid.map_upper(sns.scatterplot)
pairgrip = pairgrid.map_diag(plt.hist)
pairgrid = pairgrid.map_lower(sns.kdeplot)
g=sns.PairGrid(data=df)
g.map_lower(sns.scatterplot)
g.map_diag(sns.kdeplot)
sns.rugplot(df['petal_length'])
plt.show()
```