YEDDU DIVYA SRI

Mail:divyasri252005@gmail.com

Infosys Springboard Virtual Internship 6.0 (Batch 4 & 5)

BudgetWise AI based Expense Forecasting Tool

- **NUMPY :** NumPy stands for Numerical Python.
  It is a Python library used for performing fast mathematical, statistical, and array-based operations.

 # ~ is output

```
Pip install Numpy

import numpy as np
n1=np.array([10,20,30,40,50,60])
n2=np.array([60,40,90,100,150])

np.intersect1d(n1,n2)
~array([40, 60])

np.setdiff1d(n1,n2)
~array([10, 20, 30, 50])

np.setdiff1d(n2,n1)
~array([ 90, 100, 150])

n1=np.array([1340,130])
n2=np.array([4,5])
np.sum([n1,n2])
~np.int64(1479)

np.sum([n1,n2],axis=0)
~array([1344,  135])

np.sum([n1,n2],axis=1)
~array([1470,    9])

n1=n1+1
n1
~array([1341,  131])

n1=n1*2
n1
~array([2682,  262])
```

```
n2=n2-1
n2
~array([2, 3])
n1=n1/5
n1
~array([536.4,  52.4])

np.mean(n1)
~np.float64(294.4)

np.std(n1)
~np.float64(241.99999999999997)

np.median(n1)
~np.float64(294.4)

import numpy as np
n1=np.array([[1,2,3],[4,5,6],[7,8,9]])
n1
~array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

n1[0]
~array([1, 2, 3])

n1[1]
~array([4, 5, 6])

n1[:,1]
~array([2, 5, 8])

n1[::]
~array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

n1[:,2]
~array([3, 6, 9])

import numpy as np
n1=np.array([[1,2,3],[3,4,5],[7,8,9]])
n1
~array([[1, 2, 3],
       [3, 4, 5],
```

```
       [7, 8, 9]])
n1.transpose()
~array([[1, 3, 7],
       [2, 4, 8],
       [3, 5, 9]])

n1=np.array([[1,2,3],[3,4,5],[7,8,9]])
n2=np.array([[10,20,30],[30,40,50],[70,80,90]])
n1.dot(n2)
~array([[ 280,  340,  400],
       [ 500,  620,  740],
       [ 940, 1180, 1420]])

n2.dot(n1)
~array([[ 280,  340,  400],
       [ 500,  620,  740],
       [ 940, 1180, 1420]])

import numpy as np
n1=np.array([10,20,30,40])
n1
~array([10, 20, 30, 40])

import numpy as np
n2=np.array([[10,20,30,40],[50,60,70,80]])
n2
~array([[10, 20, 30, 40],
       [50, 60, 70, 80]])

n1=np.zeros((1,2))
n1
~array([[0., 0.]])

n1=np.zeros((5,5))
n1
~array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])

n1=np.full((5,5),10)
n1
~array([[10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10],
```

```
       [10, 10, 10, 10, 10],
       [10, 10, 10, 10, 10]])

n1=np.arange(10,20)
n1
~array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

n1=np.arange(10,50,5)
n1
~array([10, 15, 20, 25, 30, 35, 40, 45])

n1=np.random.randint(1,100,5)
n1
~array([32, 81, 88,  8, 11], dtype=int32)

n1=np.array([10,20,30])
n2=np.array([40,50,60])
np.vstack((n1,n2))
~array([[10, 20, 30],
       [40, 50, 60]])

n1=np.array([10,20,30])
n2=np.array([40,50,60])
np.hstack((n1,n2))
~array([10, 20, 30, 40, 50, 60])

n1=np.array([[1,3,4],[45,65,66]])
n1.shape
~(2, 3)

n1.shape=(3,2)
n1.shape
~(3, 2)

n1=np.array([10,20,30])
n2=np.array([40,50,60])
np.column_stack((n1,n2))
~array([[10, 40],
       [20, 50],
       [30, 60]])
```

- **PANDAS :** Pandas is a Python library used for data manipulation and analysis. It provides data structures like Series and DataFrame to handle and analyze data easily.

```
import pandas as pd
```

```
s1=pd.Series([1,2,3,4,5])
s1
~0    1
1    2
2    3
3    4
4    5
dtype: int64

s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
s1
~a    1
b    2
c    3
d    4
e    5
dtype: int64

pd.Series({'a':10,'b':20,'c':30})
~a    10
b    20
c    30
dtype: int64

pd.Series({'a':10,'b':20,'c':30},index=['c','a','g','k'])
~c    30.0
a    10.0
g     NaN
k     NaN
dtype: float64

s1=pd.Series([1,2,3,4,5,6,7,8,9])
s1[3]
~np.int64(4)

s1[::]
~0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
8    9
dtype: int64
```

```
s1[:-1]
~0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64

s1[:-2]
~0    1
1    2
2    3
3    4
4    5
5    6
6    7
dtype: int64

s1[:-3]
s1[-3:]
~6    7
7    8
8    9
dtype: int64
s1[:4]
~0    1
1    2
2    3
3    4
dtype: int64

import pandas as pd

df = pd.read_csv("iris.csv")
df.head()
~    sepal_length  sepal_width  petal_length  petal_width    species
0          5.1          3.5           1.4          0.2  setosa
 1          4.9          3.0           1.4          0.2  setosa
 2          4.7          3.2           1.3          0.2  setosa
 3          4.6          3.1           1.5          0.2  setosa
 4          5.0          3.6           1.4          0.2  setosa
```

```
df.tail()
~ sepal_length  sepal_width  petal_length  petal_width     species
145           6.7          3.0           5.2          2.3  virginica
 146           6.3          2.5           5.0          1.9  virginica
 147           6.5          3.0           5.2          2.0  virginica
 148           6.2          3.4           5.4          2.3  virginica
 149           5.9          3.0           5.1          1.8  virginica
```

```
df.shape
~ (150, 5)
```

```
df.columns
~ Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species'],
dtype='object')
```

```
df.dtypes
~ sepal_length    float64
sepal_width     float64
 petal_length    float64
 petal_width     float64
 species         object
 dtype: object
```

```
df.describe()
~        sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
 mean       5.843333     3.057333      3.758000     1.199333
 std        0.828066     0.435866      1.765298     0.762238
 min        4.300000     2.000000      1.000000     0.100000
 25%        5.100000     2.800000      1.600000     0.300000
 50%        5.800000     3.000000      4.350000     1.300000
 75%        6.400000     3.300000      5.100000     1.800000
 max        7.900000     4.400000      6.900000     2.500000
```

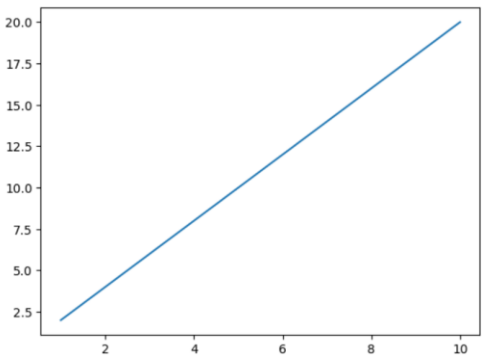
```
df.info()
~ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
 Data columns (total 5 columns):
   Column        Non-Null Count  Dtype
 ---  ------        --------------  -----
  0   sepal_length  150 non-null    float64
  1   sepal_width   150 non-null    float64
  2   petal_length  150 non-null    float64
  3   petal_width   150 non-null    float64
  4   species       150 non-null    object
 dtypes: float64(4), object(1)
```

❖ **Plotting using Matplotlib Pyplot:** Pyplot is a module in the Matplotlib library that provides simple functions for creating plots and visualizing data easily. It allows users to generate line graphs, bar charts, histograms, scatter plots, and more with simple commands.

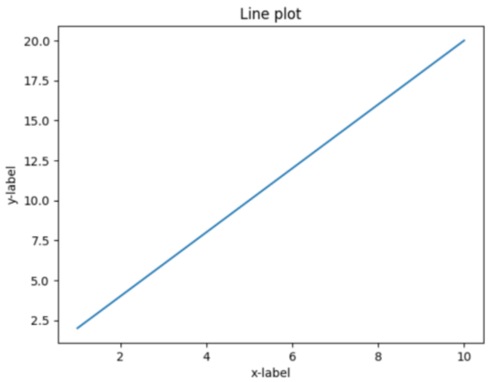
```
import numpy as np
from matplotlib import pyplot as plt

import numpy as np
x = np.arange(1, 11)
x
~ array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

y=2*x
y
~ array([ 2,  4,  6,  8, 10, 12, 14, 16, 18, 20])
```

- Line Plot: A **line plot** is a type of chart used to show data points connected by straight lines.

```
plt.plot(x,y)
plt.show()
~
```

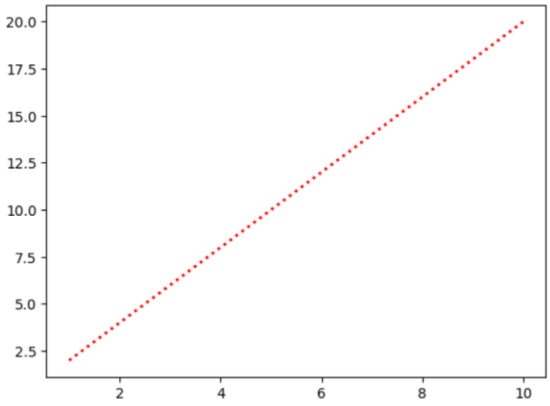



```
plt.plot(x,y)
plt.title("Line plot")
plt.xlabel("x-label")
plt.ylabel("y-label")
plt.show()
~
```



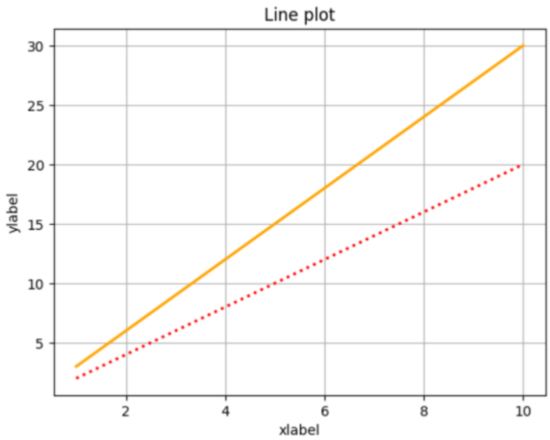
```
plt.plot(x,y,color='red',linestyle=':',linewidth=2)
plt.show()
```

~



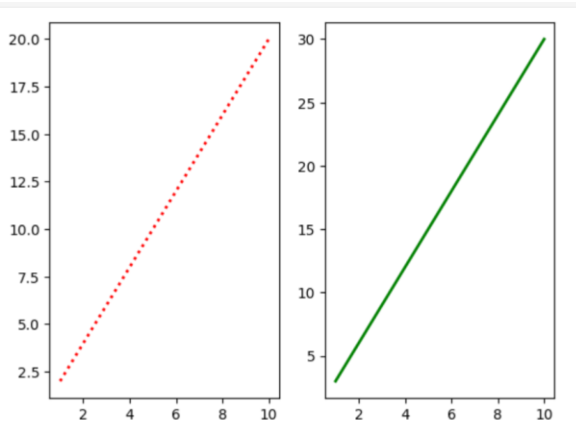

```
x=np.arange(1,11)
y1=2*x
y2=3*x

plt.plot(x,y1,color='red',linestyle=':',linewidth=2)
plt.plot(x,y2,color='orange',linestyle='-',linewidth=2)
plt.title("Line plot")
plt.xlabel('xlabel')
plt.ylabel('ylabel')
plt.grid(True)
plt.show()
```

~




```
plt.subplot(1,2,1)
plt.plot(x,y1,color='red',linestyle=':',linewidth=2)
plt.subplot(1,2,2)
plt.plot(x,y2,color='g',linestyle='-',linewidth=2)
plt.show()
```

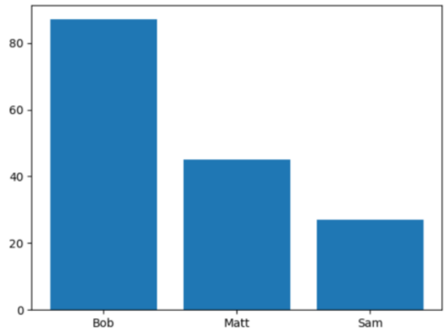
~



- Bar Plot: A **bar plot** (or bar chart) is a graph that represents **categorical data** with rectangular bars.

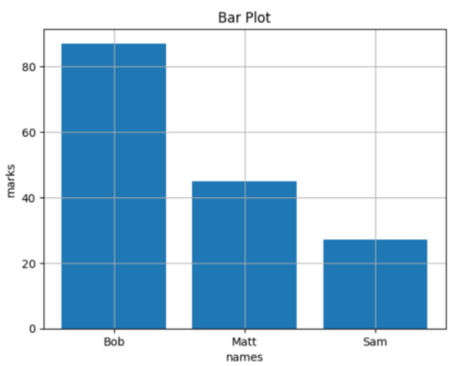
```
student={"Bob":87,"Matt":45,"Sam":27}
names=list(student.keys())
values=list(student.values())
plt.bar(names,values)
plt.show()
```

~




```
plt.bar(names,values)
plt.title("Bar Plot")
plt.xlabel('names')
plt.ylabel('marks')
plt.grid(True)
plt.show()
```

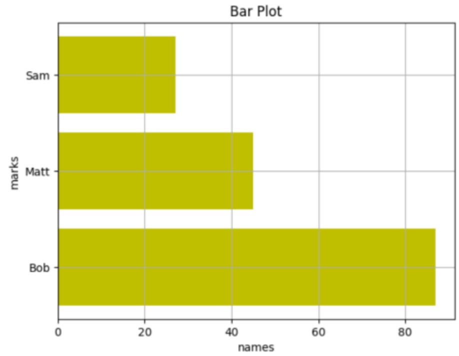
~




```
plt.barh(names,values,color='y')
```

```
plt.title("Bar Plot")
plt.xlabel('names')
plt.ylabel('marks')
plt.grid(True)
plt.show()
```

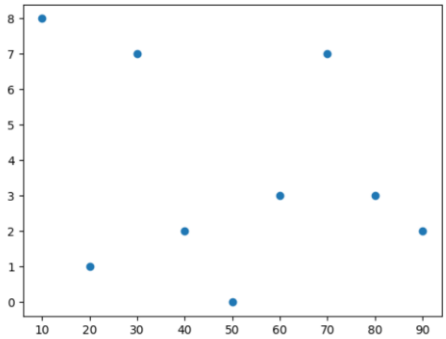
~



- Scatter Plot: A scatter plot is used to display the relationship between two variables using dots on a graph.

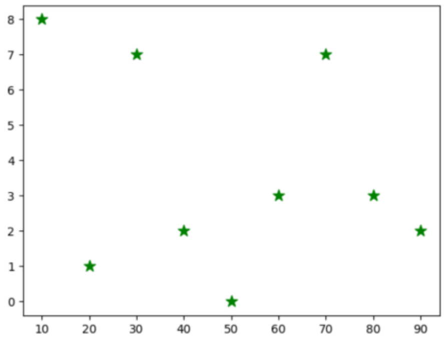
```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
plt.scatter(x,a)
plt.show()
```

~

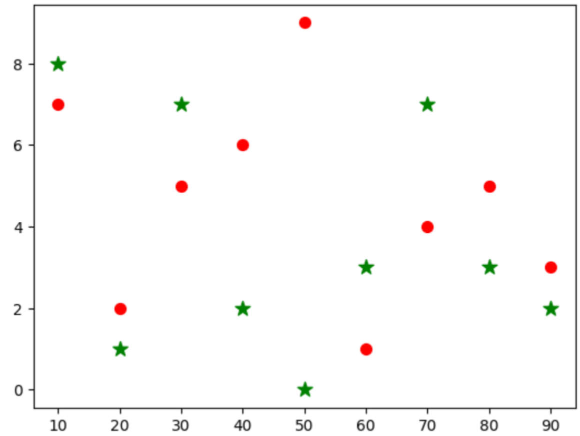



```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
plt.scatter(x,a,marker="*",c="g",s=100)
plt.show()
```

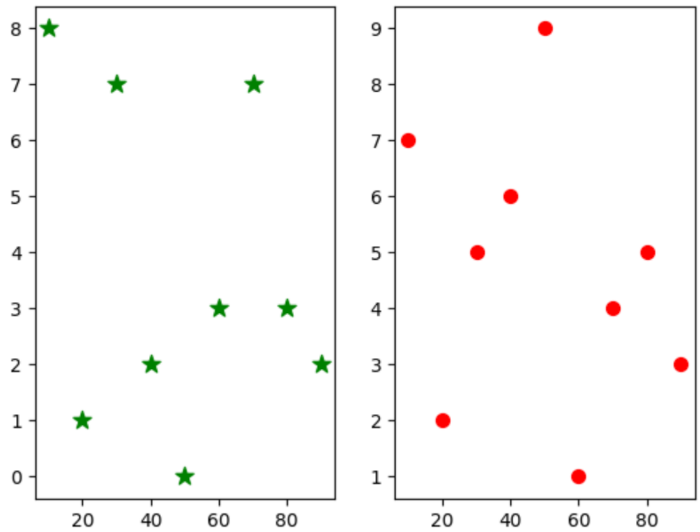
~



```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
b=[7,2,5,6,9,1,4,5,3]

plt.scatter(x,a,marker="*",c="g",s=100)
plt.scatter(x,b,marker=".",c="r",s=200)

plt.show()
~
```

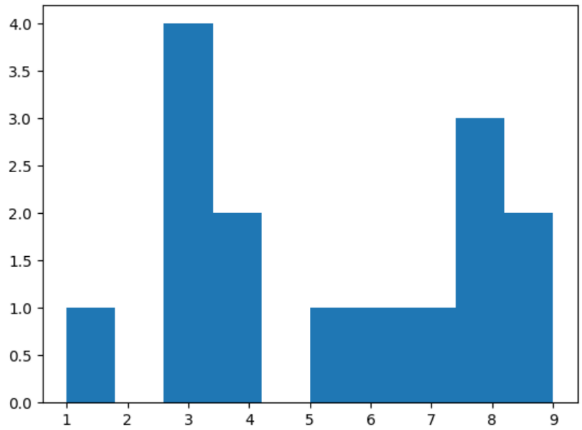



```
x=[10,20,30,40,50,60,70,80,90]
a=[8,1,7,2,0,3,7,3,2]
b=[7,2,5,6,9,1,4,5,3]
plt.subplot(1,2,1)
plt.scatter(x,a,marker="*",c="g",s=100)
plt.subplot(1,2,2)
plt.scatter(x,b,marker=".",c="r",s=200)

plt.show()
~
```

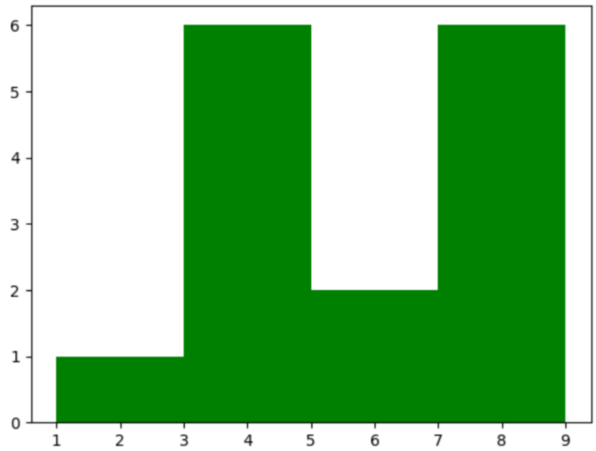


- Histogram Plot: A histogram plot is used to show the frequency distribution of continuous data using adjacent rectangular bars.

```
#histogram plot
#creating data
data=[1,3,3,3,3,9,9,5,4,4,8,8,8,6,7]
#making histogram
plt.hist(data)
plt.show()
~
```




```
plt.hist(data,color="g",bins=4)
plt.show()
~
```

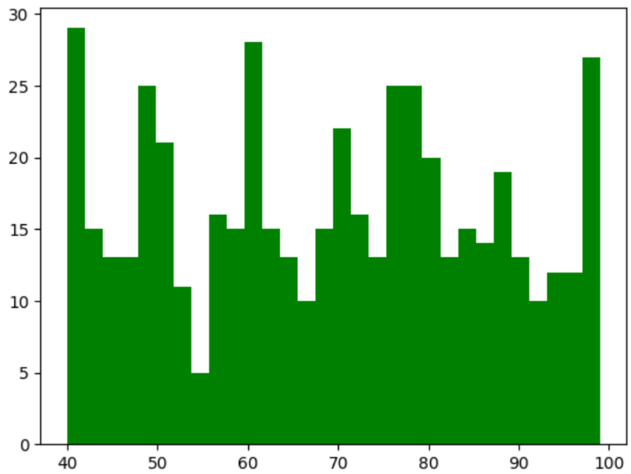



```
import pandas as pd
soil=pd.read_csv('soil_crop_dataset.csv')
soil.head()
~
```

	pH	Nitrogen	Phosphorus	Potassium	Moisture	Temperature	Rainfall	Soil_Type	Recommended_Crop
0	5.53	72	52	60	22.01	24.91	249	Sandy,Peaty,Loamy	Peanut,Maize,Tomato,Barley,Carrot
1	6.71	87	18	45	21.85	29.03	135	Clayey,Peaty,Red,Black	Peanut,Onion,Potato,Millet
2	8.49	88	13	45	20.36	18.88	52	Red,Black,Clayey	Potato,Millet,Tomato,Cotton
3	7.27	46	60	31	15.64	19.61	77	Silty,Red,Clayey	Potato,Rice,Tomato,Wheat,Onion
4	8.22	60	60	33	8.81	33.09	171	Silty,Sandy,Peaty	Tomato,Potato,Wheat,Rice

```
plt.hist(soil['Nitrogen'],bins=30,color='g')
plt.show()
```

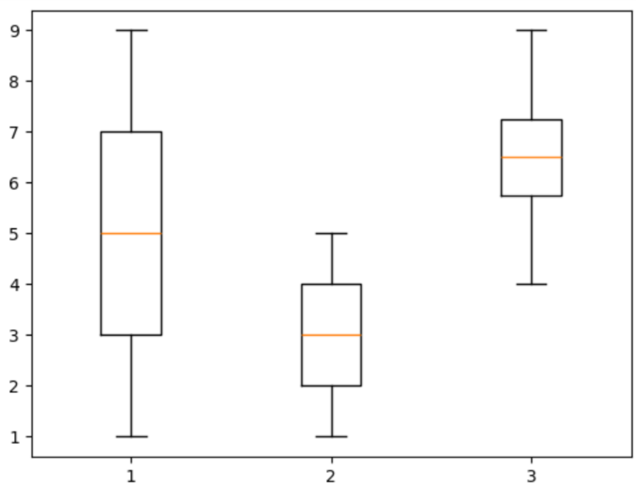
~



- Box Plot: A box plot is used to display the distribution, spread, and outliers of a dataset using quartiles.

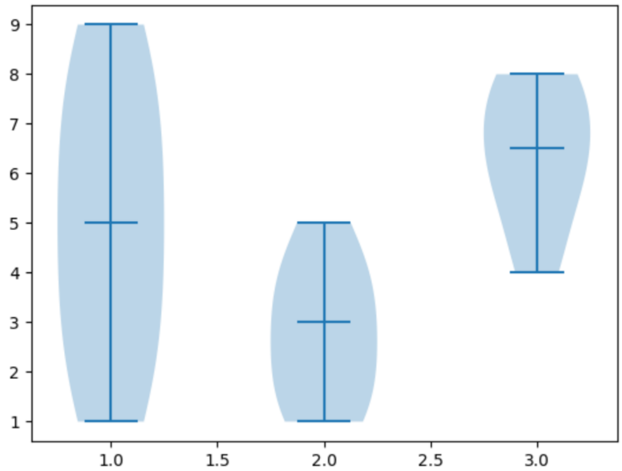
```
one=[1,2,3,4,5,6,7,8,9]
two=[1,2,3,4,5,4,3,2,1]
three=[6,7,8,9,7,6,5,4]
data=list([one,two,three])
plt.boxplot(data)
plt.show()
```

~



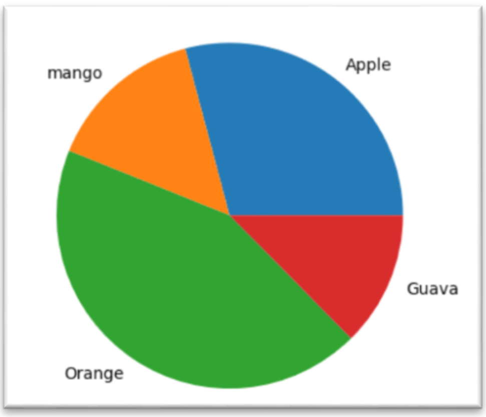
- Violin Plot: A violin plot is used to show the distribution of data across categories, combining features of a box plot and a density plot.

```
one=[1,2,3,4,5,6,7,8,9]
two=[1,2,3,4,5,4,3,2,1]
three=[6,7,8,7,8,6,5,4]
data=list([one,two,three])
plt.violinplot(data,showmedians=True)
plt.show()
~
```



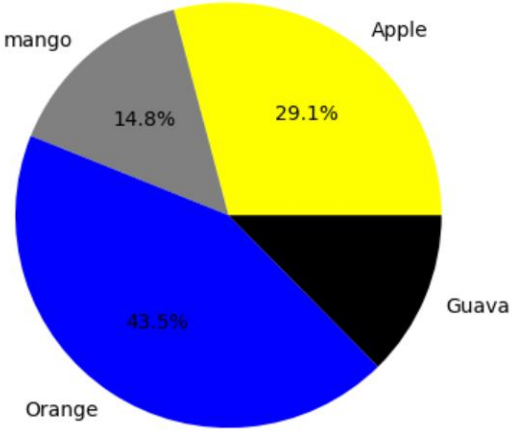
- Pie Plot: A pie plot is used to show the proportion or percentage of different categories as slices of a whole circle.

```
fruit=['Apple','mango',"Orange","Guava"]
quantity=[67,34,100,29]
plt.pie(quantity,labels=fruit)
plt.show()
~
```



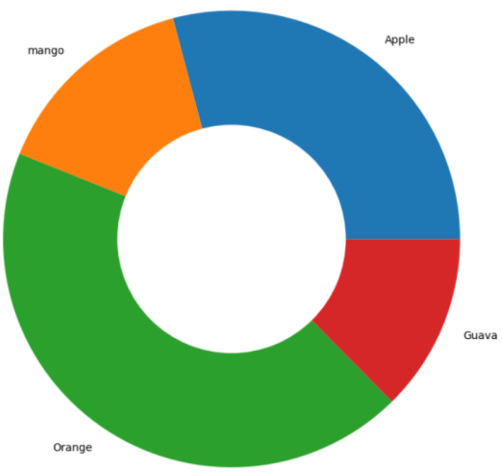

```
plt.pie(quantity,labels=fruit,autopct='%0.1f%%',colors=['yellow','grey','blue','b
lack'])
plt.show()
```

~




```
plt.pie(quantity,labels=fruit,radius=2)
plt.pie([1],colors=['w'],radius=1)
plt.show()
```

~



❖ Plotting using Seaborn and Matplotlib Pyplot:

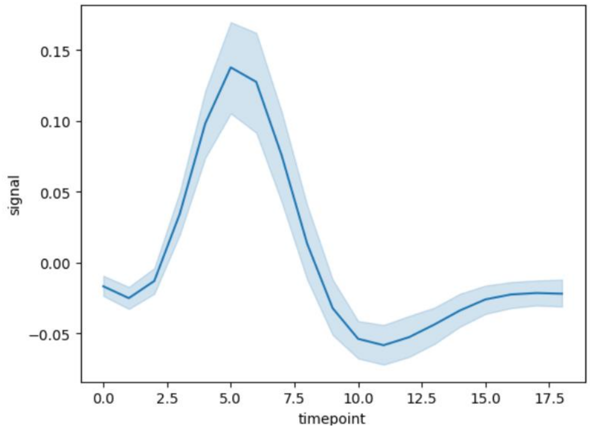
Seaborn is a Python data visualization library built on top of Matplotlib.

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

fmri = sns.load_dataset("fmri")
fmri.head()
```

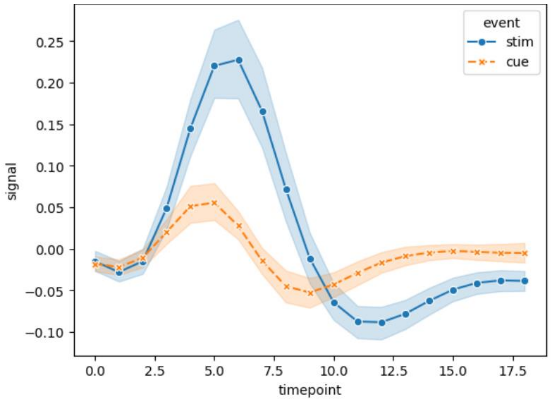
~

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

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




```
sns.lineplot(x="timepoint",y="signal",hue="event",style="event",markers=True,data=fmri)
plt.show()
```
~




```
fmri=sns.load_dataset("fmri")
fmri.head()
```
~

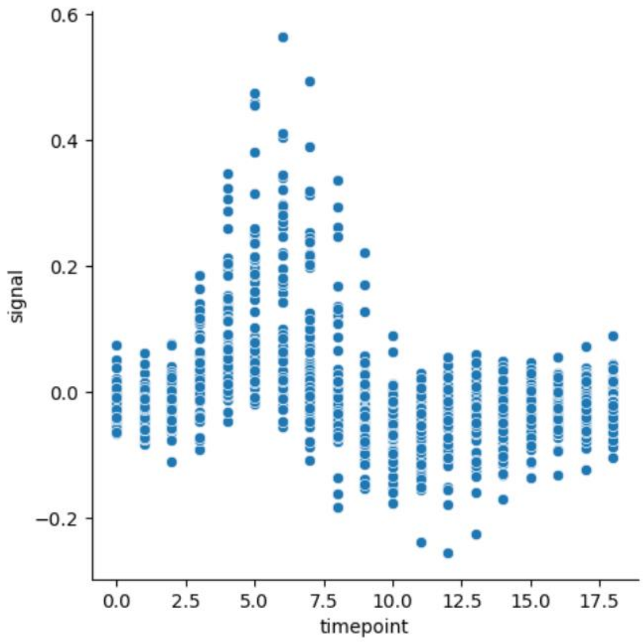
	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970

- Relplot: A relplot (relationship plot) in Seaborn is used to visualize relationships between two variables, often as scatter or line plots.

```
fmri.shape
```
~ (1064, 5)

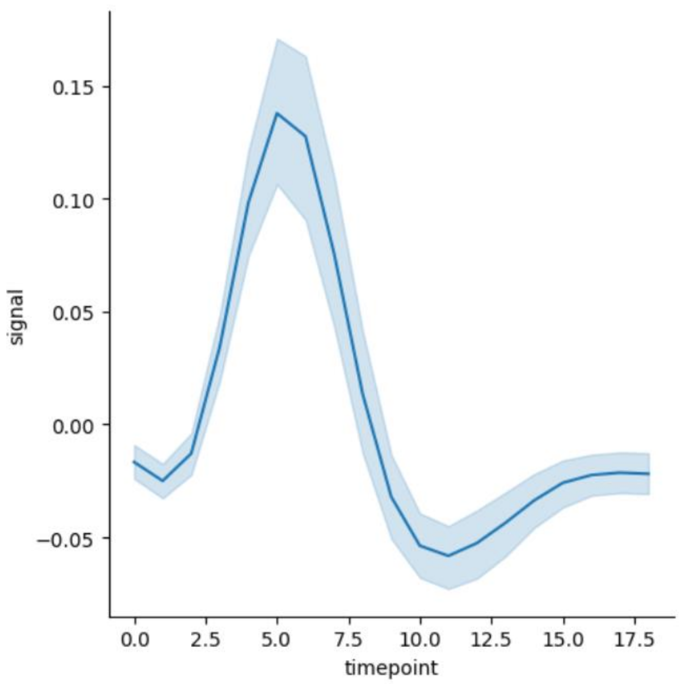
```
sns.relplot(data=fmri,x="timepoint",y="signal")
~
```

```
<seaborn.axisgrid.FacetGrid at 0x26ceb742780>
```




```
sns.relplot(data=fmri,x="timepoint",y="signal",kind="line")
~
```

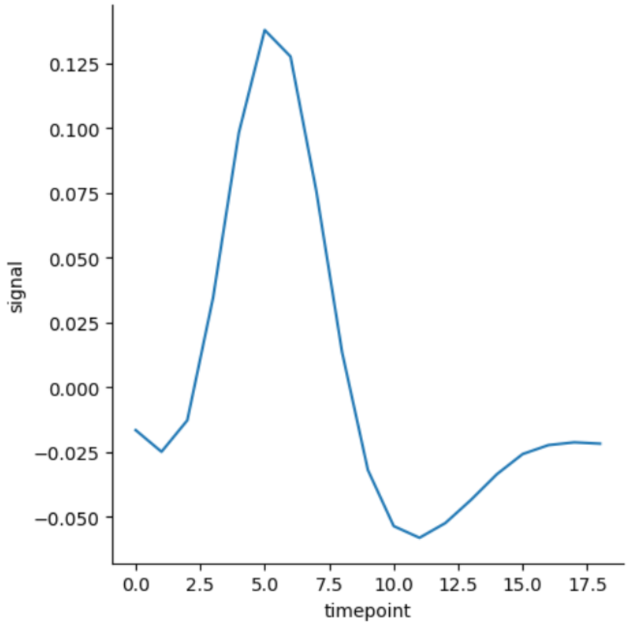
```
<seaborn.axisgrid.FacetGrid at 0x26cec7f9bb0>
```




```
sns.relplot(
    data=fmri,kind="line",
    x="timepoint",y="signal",errorbar=None
)
```

~

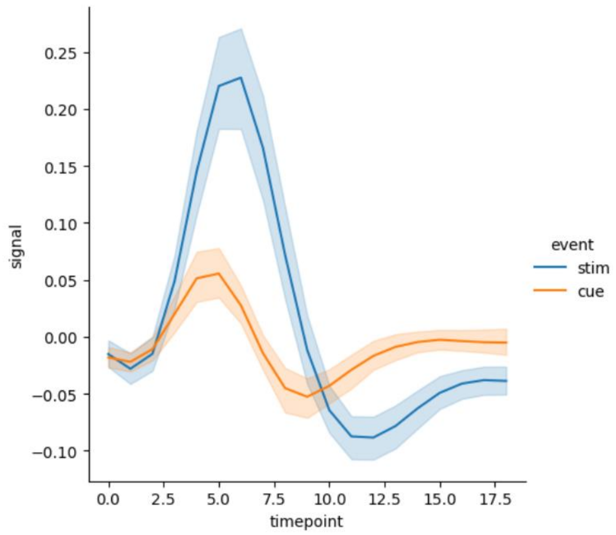
<seaborn.axisgrid.FacetGrid at 0x26cec811be0>



Hue: It is used to add a third variable in a plot by coloring the data points according to different categories.
Kind: It specifies the type of plot to draw, such as 'scatter' or 'line' in functions like relplot().
style: It is used to change the appearance of data points or lines (like marker shapes or line styles) based on a categorical variable.

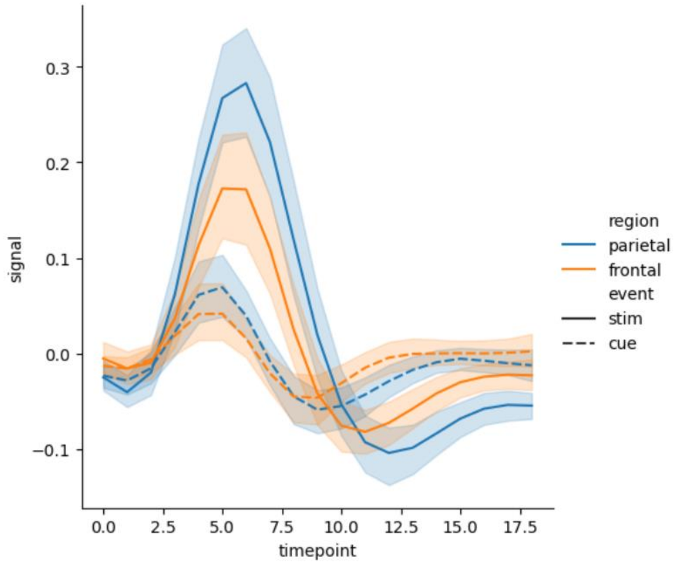
```
sns.relplot(
   data=fmri,kind="line",
   x="timepoint",y="signal",
   hue="event"
)
```
~

<seaborn.axisgrid.FacetGrid at 0x26cec8ca780>



```
sns.relplot(
   data=fmri,kind="line",
   x="timepoint",y="signal",
   hue="region",style="event"
)
~
```

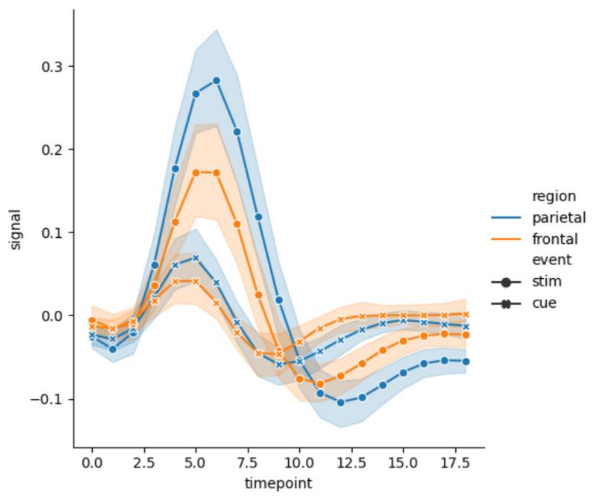
<seaborn.axisgrid.FacetGrid at 0x26cec7c6930>



Dashes: Used to control or remove the line style (solid, dashed, etc.) in line plots.
Markers: Used to display symbols at data points (like circles, squares, or triangles) in line or scatter plots.

```
sns.relplot(
   data=fmri,kind="line",
   x="timepoint",y="signal",
   hue="region",style="event",
   dashes=False,markers=True
)
~
```

<seaborn.axisgrid.FacetGrid at 0x26cec937860>



❖ Categorical scatterplots

- Categorical scatterplots are used to show the relationship between a categorical variable and a numerical variable using points instead of bars.
- **Categorical data** represents values that belong to distinct groups or categories, such as colors, names, or types.

two types of categorical data:

- **Nominal Data:** Categories that **do not have any specific order** (e.g., colors — red, blue, green).
- **Ordinal Data:** Categories that **have a meaningful order or ranking** (e.g., ratings — poor, average, good, excellent).

```
import seaborn as sns
tips=sns.load_dataset("tips")
tips.head()
```

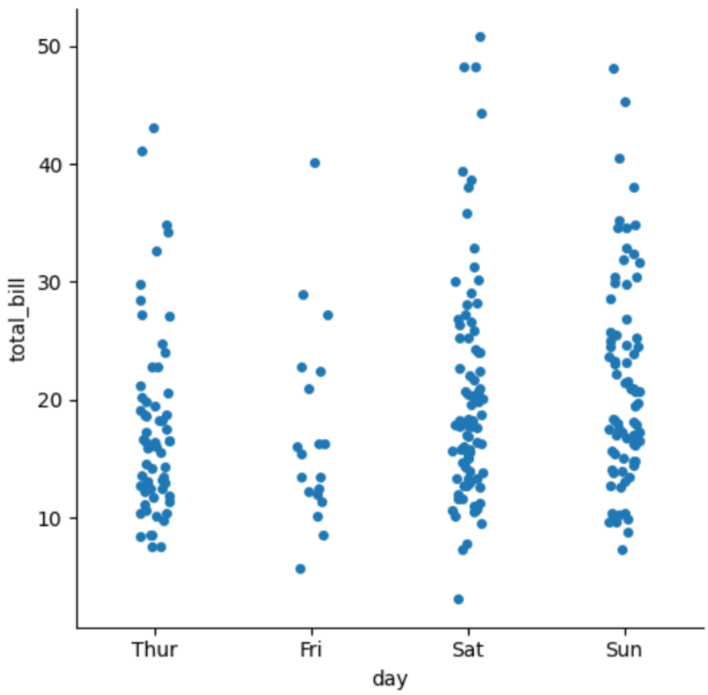
~

	total_bill	tip	sex	smoker	day	time	size
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

```
sns.catplot(data=tips,x='day',y='total_bill')
```

~

<seaborn.axisgrid.FacetGrid at 0x26ceabf75f0>

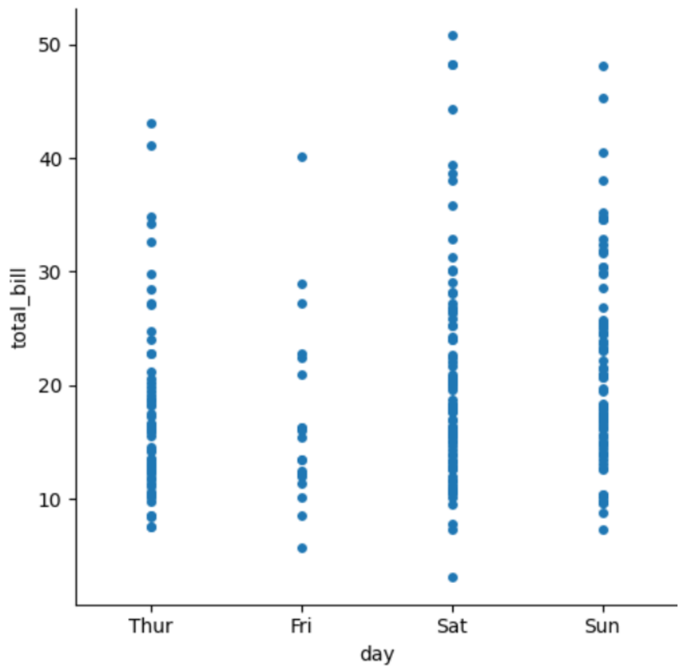


Jitter: adds small random noise to data points in a plot to prevent overlapping and make the distribution clearer.

sns.catplot(data=tips,x='day',y='total_bill',jitter=False)

~

```
&lt;seaborn.axisgrid.FacetGrid at 0x26cdaf3d850&gt;
```

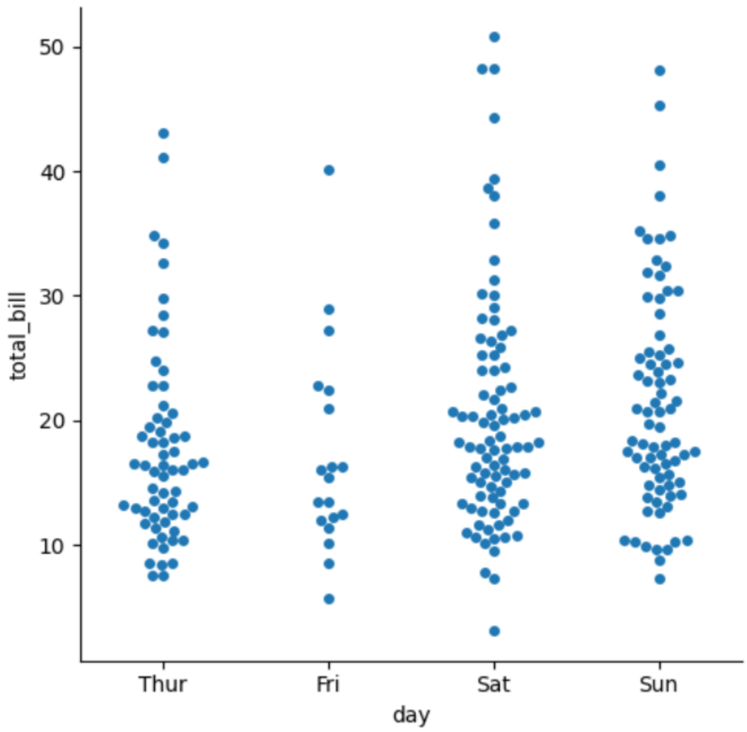


#prevent from overlapping (swarm plot)

sns.catplot(data=tips,x='day',y='total_bill',kind='swarm')

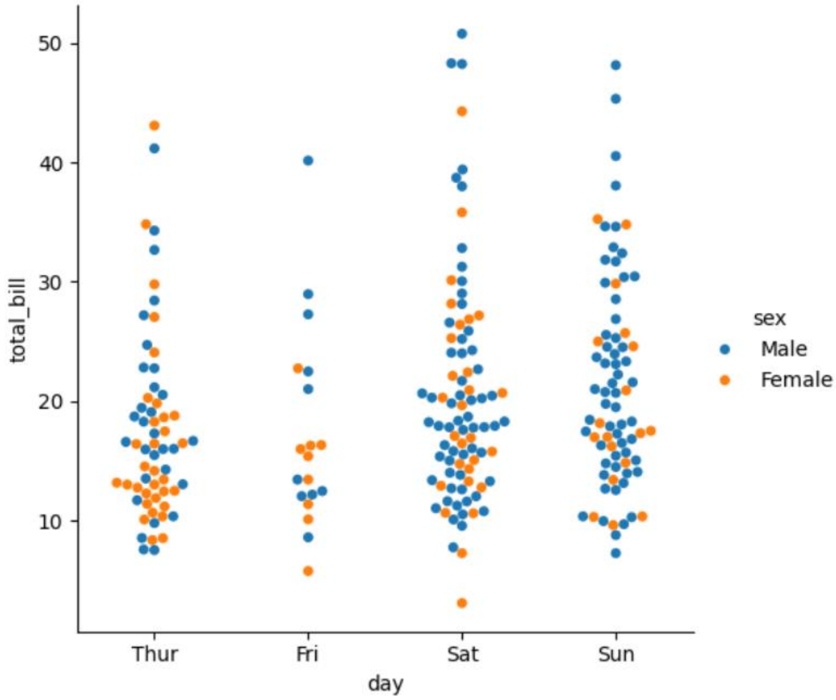
~

```
<seaborn.axisgrid.FacetGrid at 0x26ceca84fb0>
```



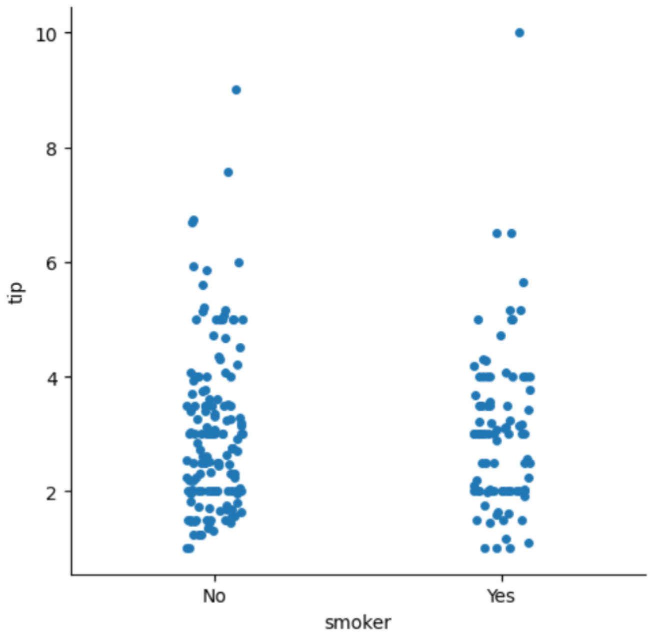
```
#add the hue sematic
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',kind='swarm')
```
~

<seaborn.axisgrid.FacetGrid at 0x26cee0e0b90>



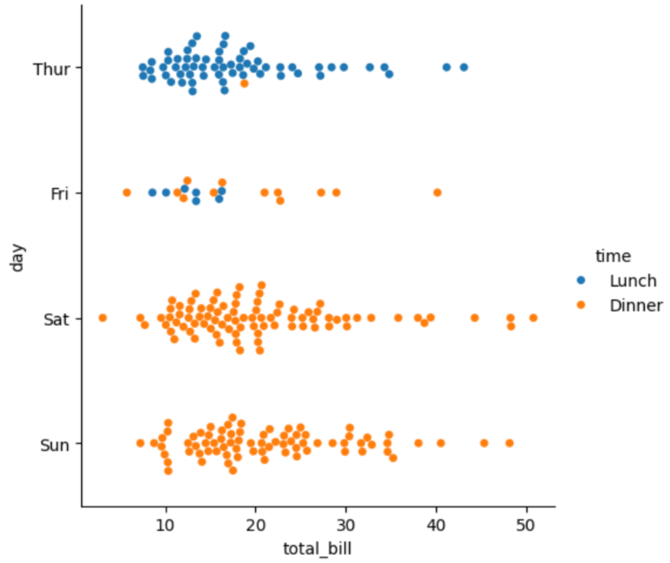

```
#order parameter - to display multiple categorical plot in same figure
sns.catplot(data=tips,x='smoker',y='tip',order=['No','Yes'])
```
~

<seaborn.axisgrid.FacetGrid at 0x26cee0f2c60>



#plot on vertical
sns.catplot(data=tips,x='total_bill',y='day',hue='time',kind='swarm')
~

<seaborn.axisgrid.FacetGrid at 0x26cec812240>



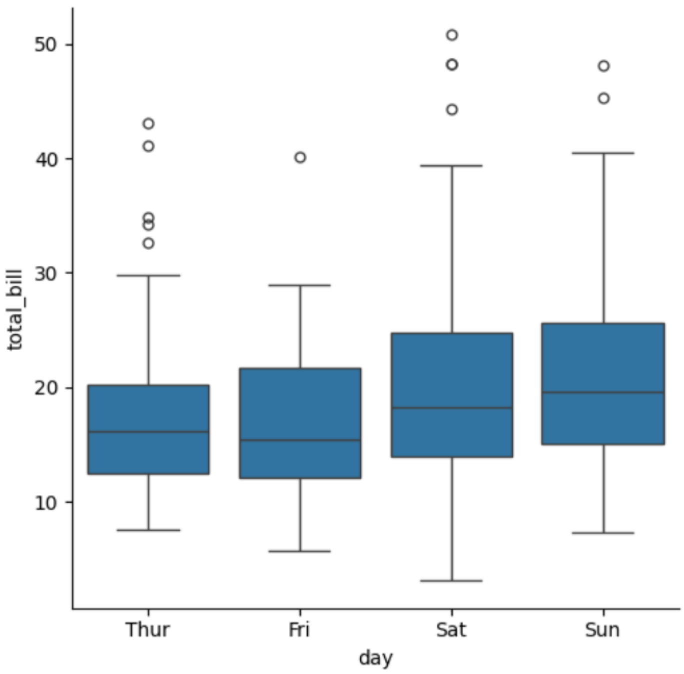
- **Comparing distribution** means analyzing how data values are spread across different groups or categories to identify patterns or differences.

#comparing distribution

- **box plot:** A box plot is used to display the distribution of data based on five summary statistics — minimum, first quartile, median, third quartile, and maximum.

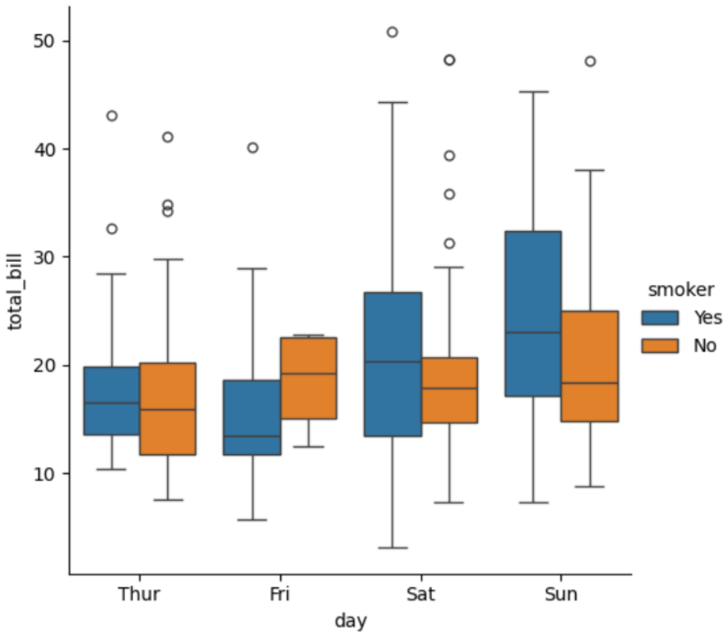
#sns.catplot(data=tips,x='total_bill',y='day',kind='box')
sns.catplot(data=tips,x='day',y='total_bill',kind='box')
~

<seaborn.axisgrid.FacetGrid at 0x26cf0237e30>



#adding the hue sematic
sns.catplot(data=tips,x='day',y='total_bill',hue='smoker',kind='box')
~

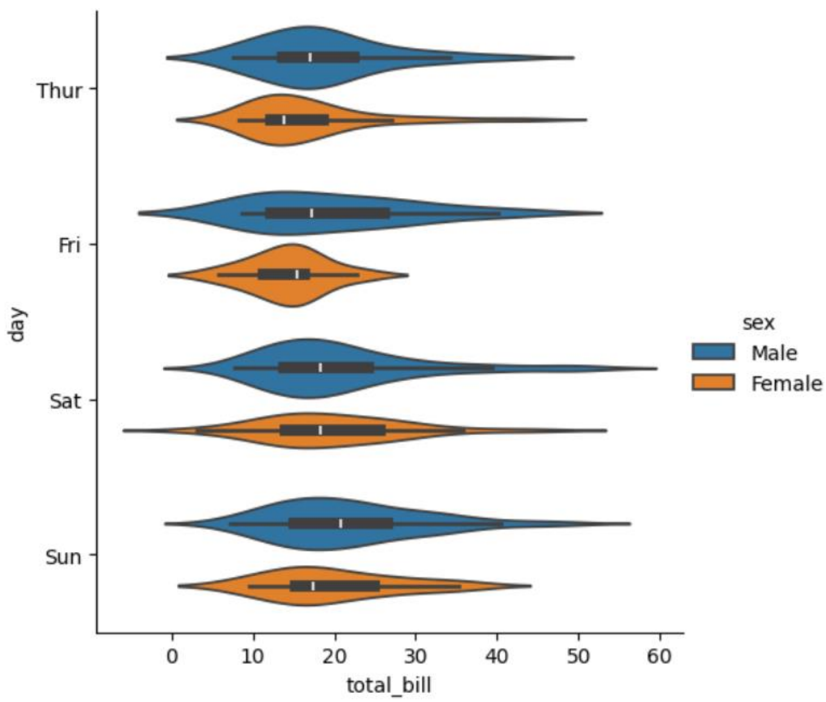
<seaborn.axisgrid.FacetGrid at 0x26cf1c3eb10>



- **violin plot**: A violin plot is used to visualize the distribution of data and its probability density, combining features of a box plot and a kernel density plot.

sns.catplot(data=tips,x='total_bill',y='day',hue='sex',kind='violin')
~

<seaborn.axisgrid.FacetGrid at 0x26cf1d8a480>



#split in the violin plot
sns.catplot(data=tips,x='day',y='total_bill',hue='sex',kind='violin',split=True)
~

```
<seaborn.axisgrid.FacetGrid at 0x26cf57783e0>
```