Python for Beginners

Archer Infotech , PUNE

Python - Seaborn

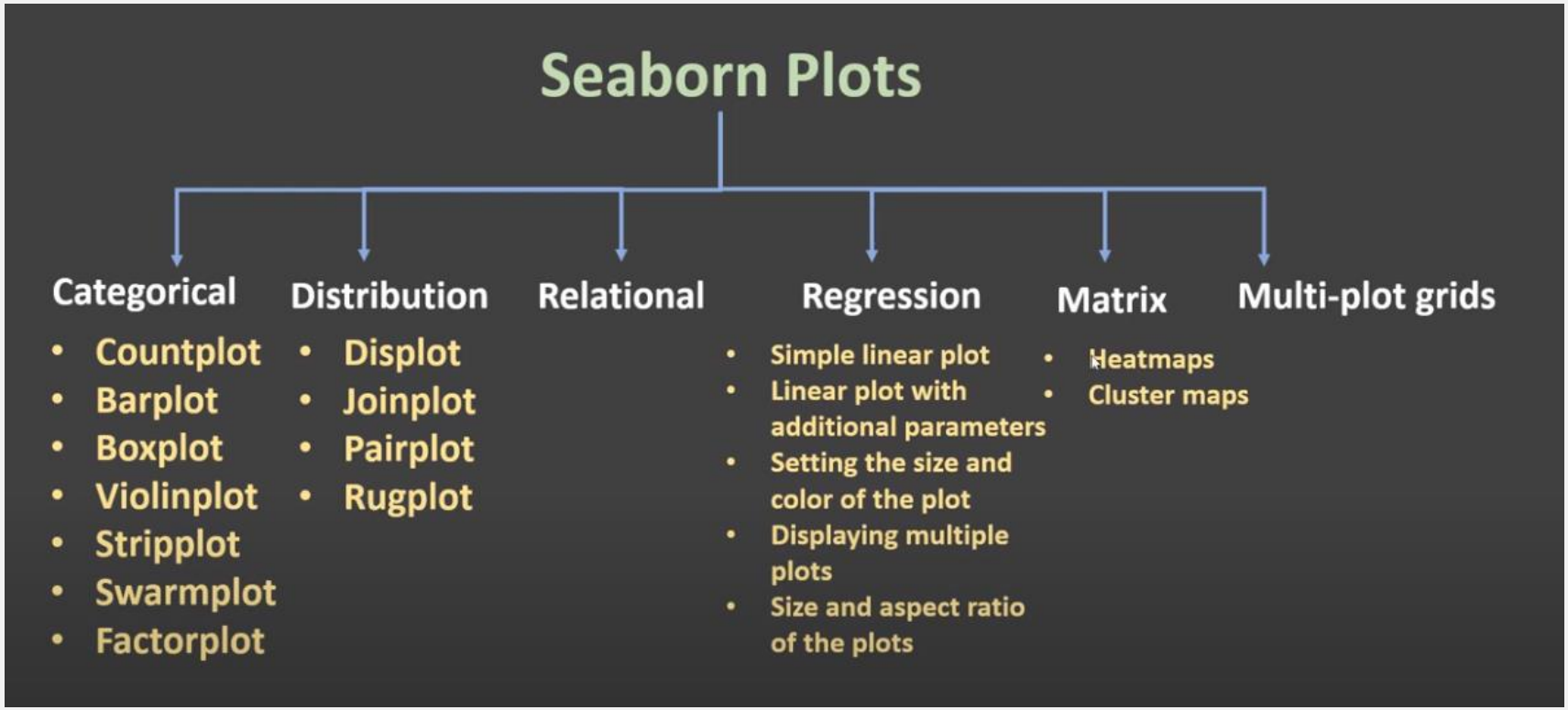
What is Seaborn ?

- **Seaborn is a Python visualization library for statistical plotting. It comes equipped with preset styles and color palettes so you can create complex, aesthetically pleasing charts with a few lines of code**

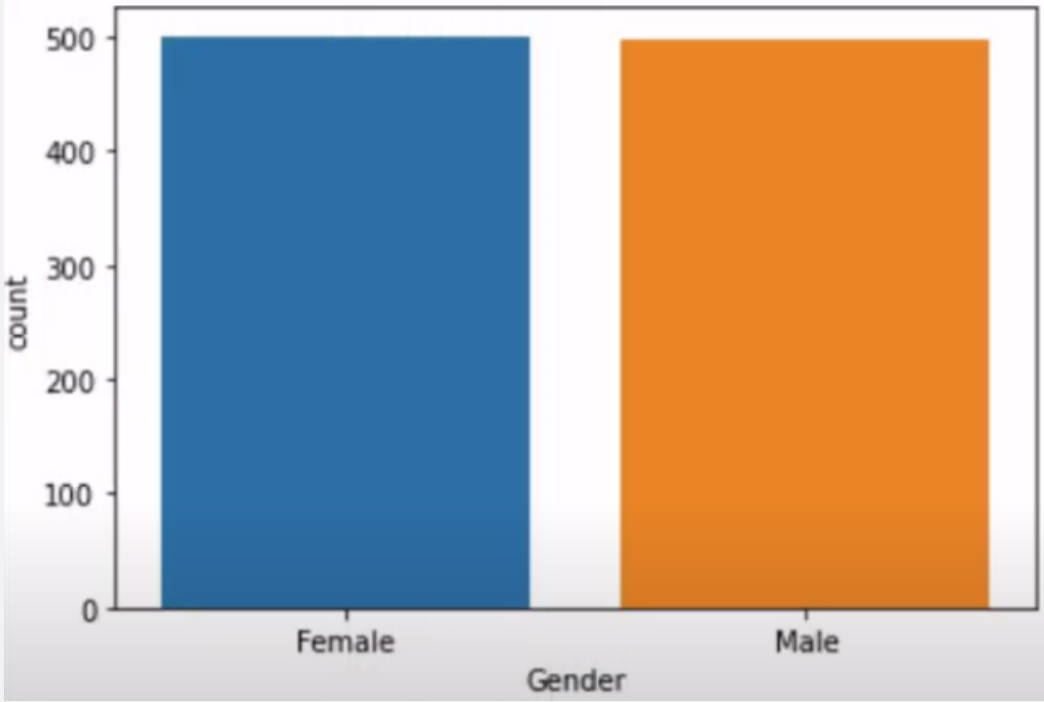
- It supports NumPy and Pandas data structure to represent the data sets.

- Seaborn is built on top of Python's core visualization library matplotlib, but it's meant to serve as a complement, not a replacement.

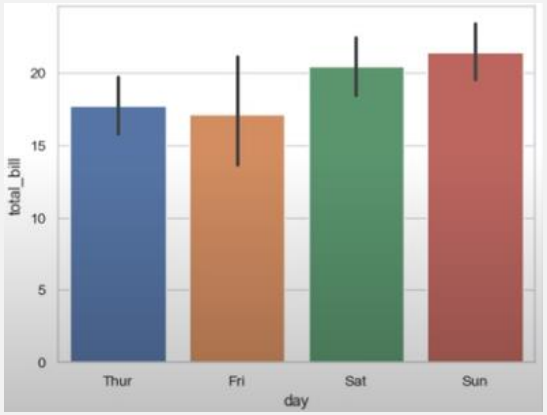
Seaborn Plots

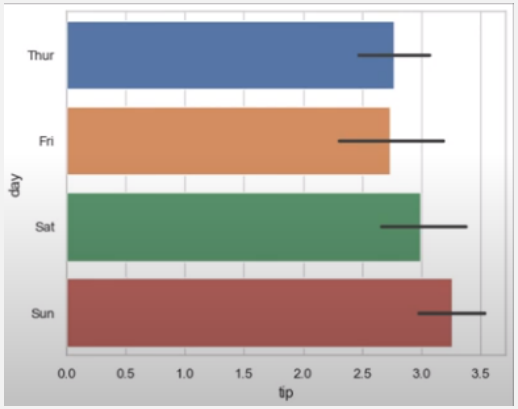


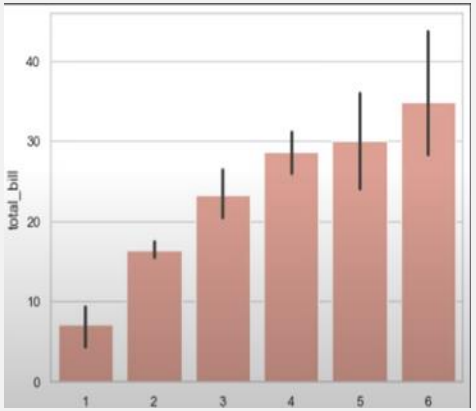
1. CountPlot



2. BarPlot







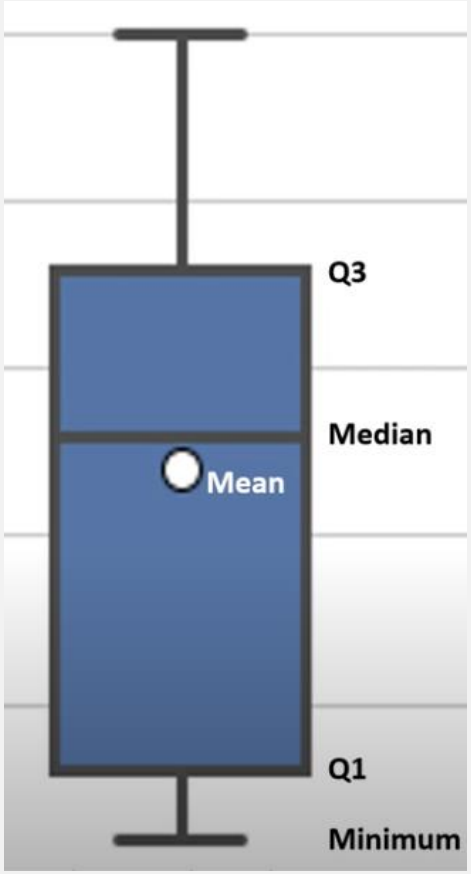
A bar plot represents an estimate of central tendency for a numeric variable with the height of each rectangle and provides some indication of the uncertainty around that estimate using error bars

```
seaborn.barplot(*, x=None, y=None, hue=None, data=None, order=None, hue_order=None, estimator=<function mean at 0x7ff320f315e0>, ci=95, n_boot=1000, units=None, seed=None, orient=None, color=None, palette=None, saturation=0.75, errcolor='.26', errwidth=None, capsize=None, dodge=True, ax=None, **kwargs)
```

3. BoxPlot

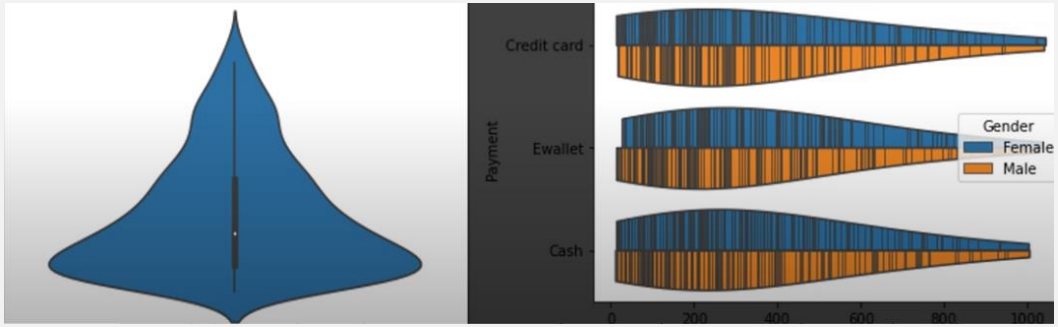
This provides a summary of supplied data, which includes the information like:

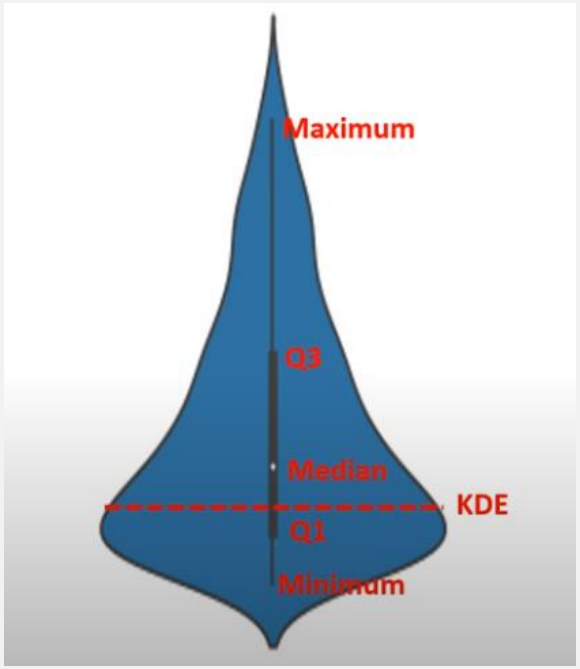
- Mean
- Median
- Minimum
- Maximum
- 1st Quartile
- 3rd Quartile
- Outliers




```
seaborn.boxplot(*, x=None, y=None, hue=None, data=None,
order=None, hue_order=None, orient=None, color=None,
palette=None, saturation=0.75, width=0.8, dodge=True,
fliersize=5, linewidth=None, whis=1.5, ax=None, **kwargs)
```

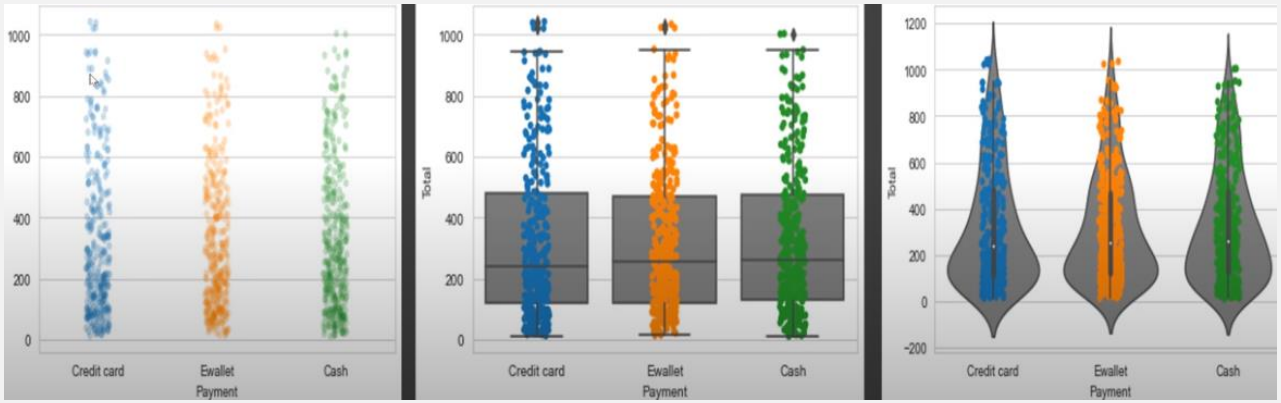
4.ViolinPlot






```
seaborn.violinplot(*, x=None, y=None, hue=None, data=None,
order=None, hue_order=None, bw='scott', cut=2, scale='area',
scale_hue=True, gridsize=100, width=0.8, inner='box', split=False,
dodge=True, orient=None, linewidth=None, color=None, palette=None,
saturation=0.75, ax=None, **kwargs)
```

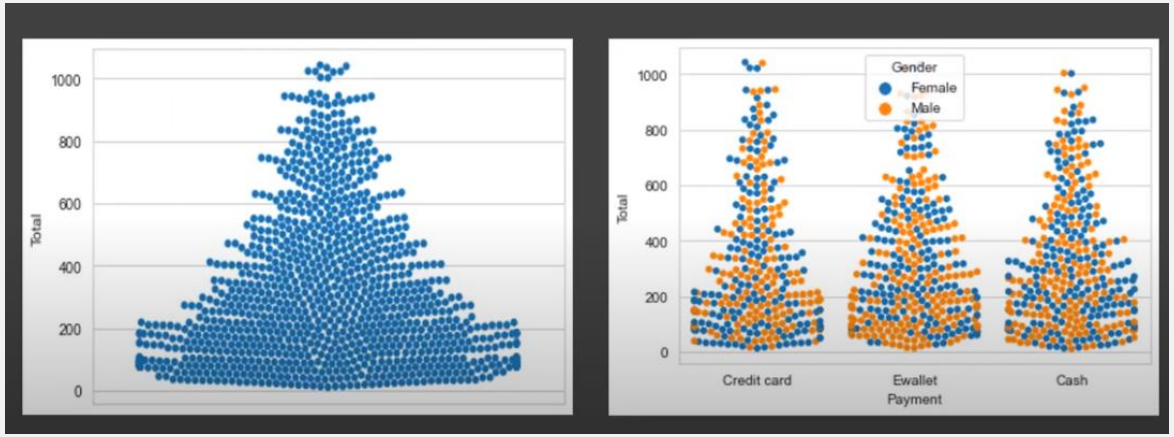
5 . StripPlot

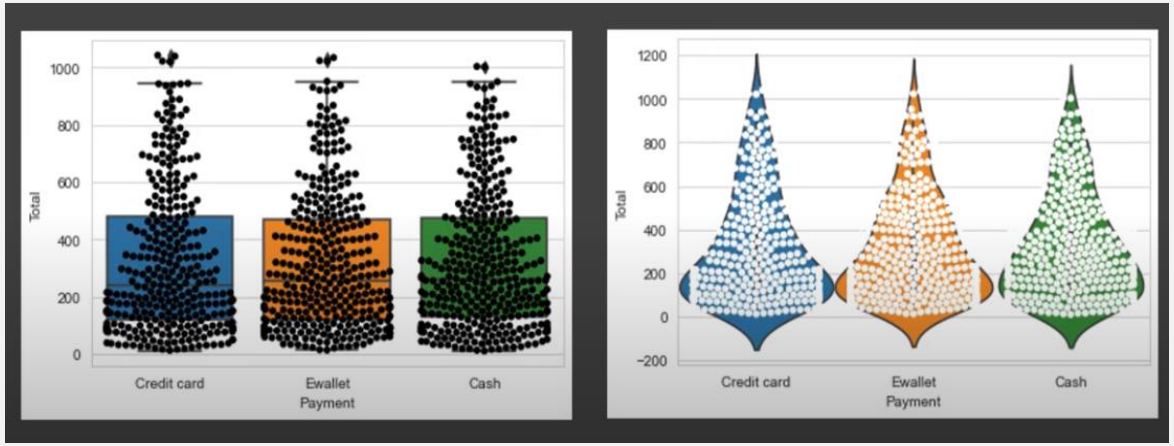


A strip plot is a graphical data anlysis technique for summarizing a univariate data set

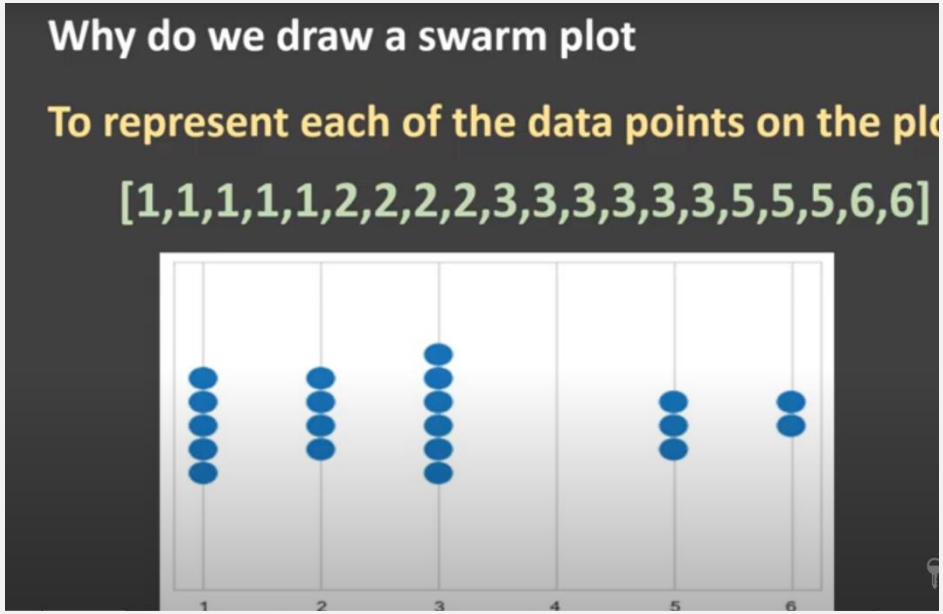
A strip plot can be drawn on its own, but it is also a good complement to a box or violin plot in cases where you want to show all observations along with some representation of the underlying distribution

6 . SwarmPlot


Total
1000
800
600
400
200
0
Gender
Female
Male
Credit card
Ewallet
Cash
Payment


Total
1000
800
600
400
200
0
1200
−200
Credit card
Ewallet
Cash
Payment

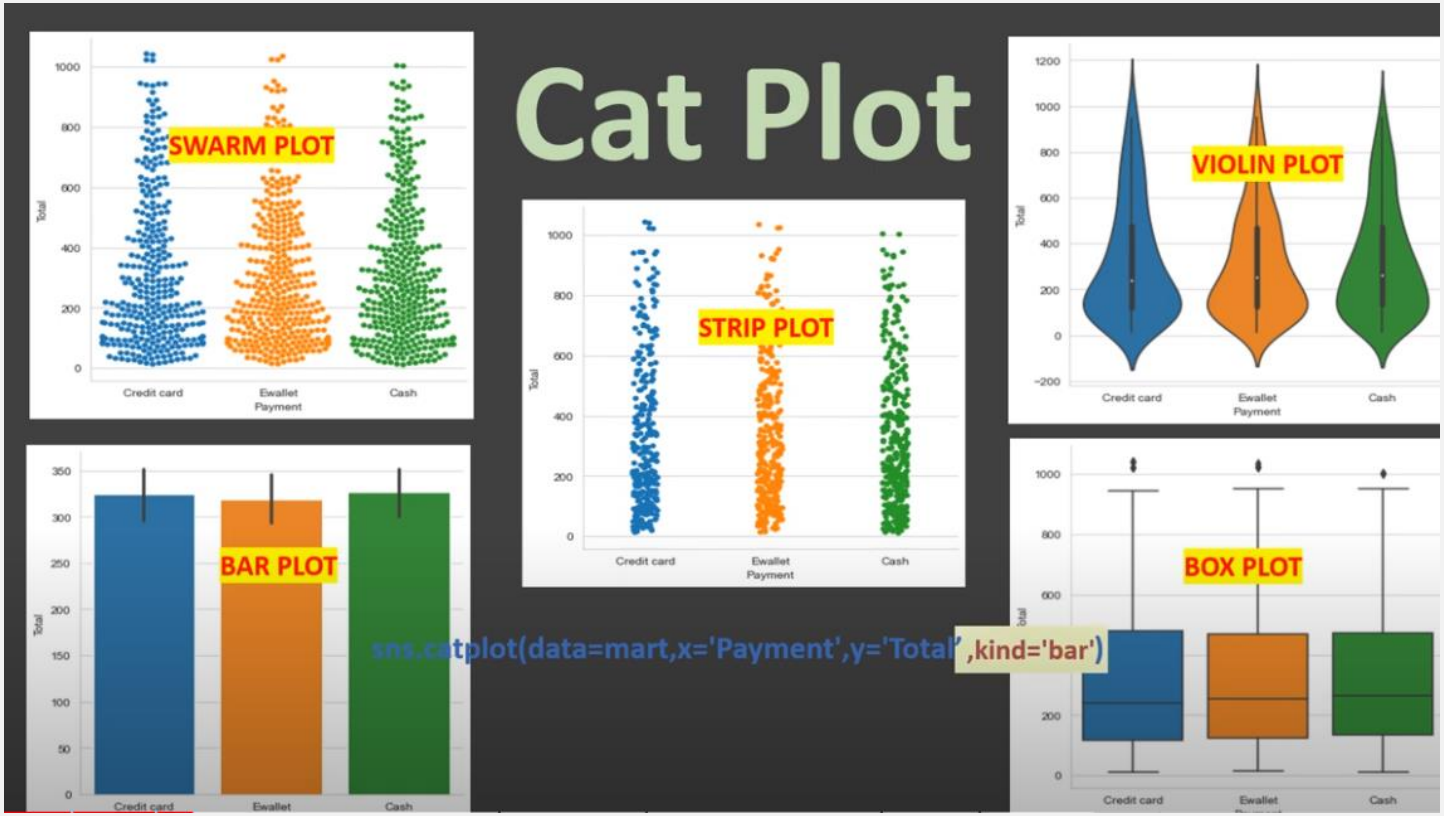
6 . SwarmPlot




```
seaborn.swarmplot(*, x=None, y=None, hue=None, data=None, order=None,
hue_order=None, dodge=False, orient=None, color=None, palette=None,
size=5, edgecolor='gray', linewidth=0, ax=None, **kwargs)
```

6 . SwarmPlot

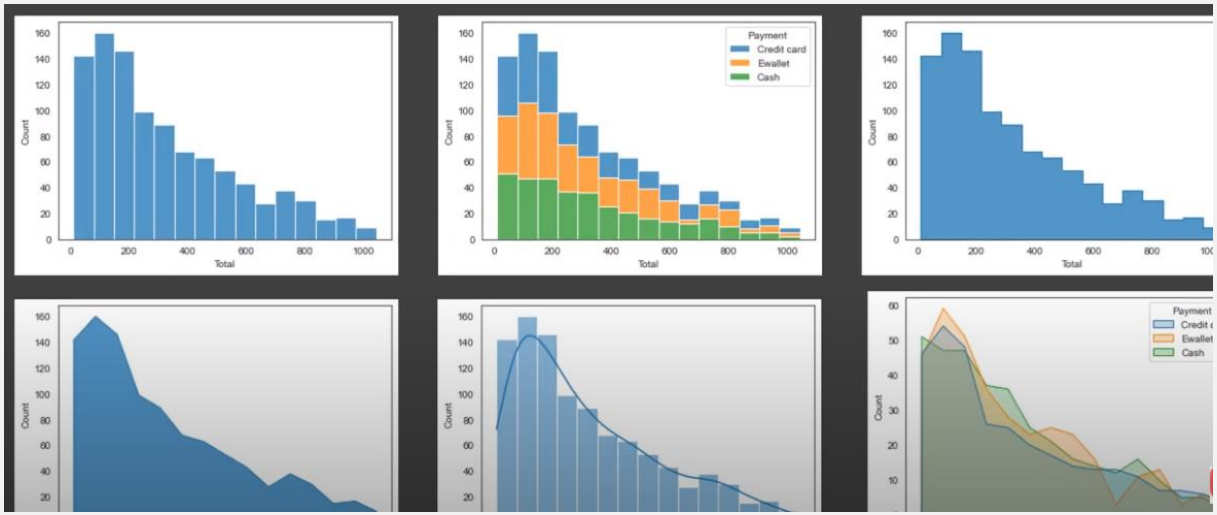
7 . CatPlot


Cat Plot
SWARM PLOT
STRIP PLOT
VIOLIN PLOT
BAR PLOT
BOX PLOT
sns.catplot(data=mart,x='Payment',y='Total',kind='bar')
Credit card
Ewallet
Payment
Cash
Total

7 . CatPlot

```
seaborn.catplot(*, x=None, y=None, hue=None, data=None, row=None,
col=None, col_wrap=None, estimator=<function mean at 0x7ff320f315e0>,
ci=95, n_boot=1000, units=None, seed=None, order=None, hue_order=None,
row_order=None, col_order=None, kind='strip', height=5, aspect=1,
orient=None, color=None, palette=None, legend=True, legend_out=True,
sharex=True, sharey=True, margin_titles=False, facet_kws=None, **kwargs)
```

8 . HistPlot



A histogram is a classic visualization tool that represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.

This function can normalize the statistic computed within each bin to estimate frequency, density or probability mass, and it can add a smooth curve obtained using a kernel density estimate, similar to kdeplot().

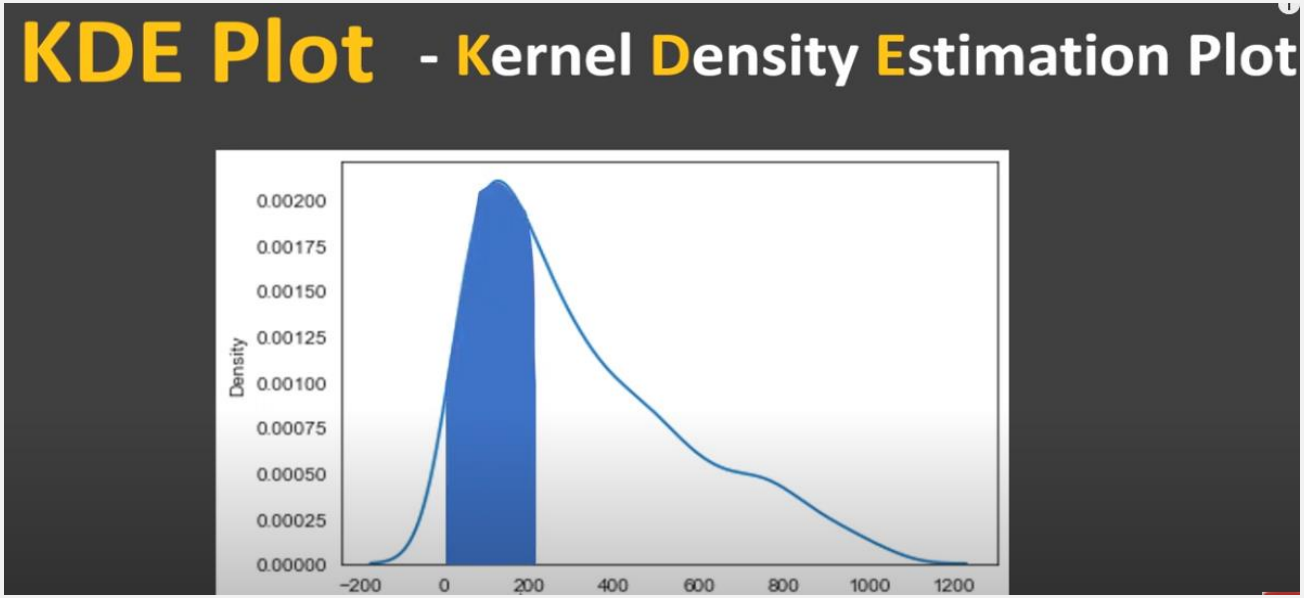
8 . HistPlot

```
seaborn.histplot(data=None, *, x=None, y=None, hue=None, weights=None,
stat='count', bins='auto', binwidth=None, binrange=None, discrete=None,
cumulative=False, common_bins=True, common_norm=True, multiple='layer',
element='bars', fill=True, shrink=1, kde=False, kde_kws=None, line_kws=None,
thresh=0, pthresh=None, pmax=None, cbar=False, cbar_ax=None,
cbar_kws=None, palette=None, hue_order=None, hue_norm=None, color=None,
log_scale=None, legend=True, ax=None, **kwargs)
```

8 . HistPlot

```
seaborn.histplot(data=None, *, x=None, y=None, hue=None, weights=None,
stat='count', bins='auto', binwidth=None, binrange=None, discrete=None,
cumulative=False, common_bins=True, common_norm=True, multiple='layer',
element='bars', fill=True, shrink=1, kde=False, kde_kws=None, line_kws=None,
thresh=0, pthresh=None, pmax=None, cbar=False, cbar_ax=None,
cbar_kws=None, palette=None, hue_order=None, hue_norm=None, color=None,
log_scale=None, legend=True, ax=None, **kwargs)
```

9. KDE Plot

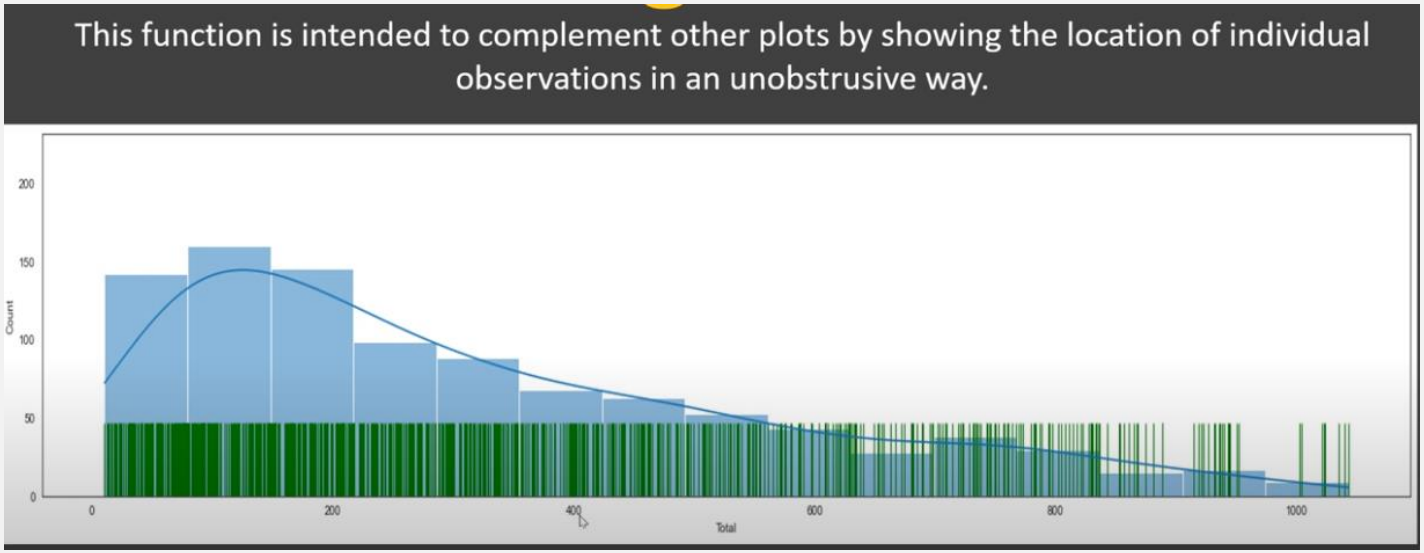



```
seaborn.kdeplot(x=None, *, y=None, shade=None, vertical=False, kernel=None, bw=None,
gridsize=200, cut=3, clip=None, legend=True, cumulative=False, shade_lowest=None,
cbar=False, cbar_ax=None, cbar_kws=None, ax=None, weights=None, hue=None,
palette=None, hue_order=None, hue_norm=None, multiple='layer', common_norm=True,
common_grid=False, levels=10, thresh=0.05, bw_method='scott', bw_adjust=1,
log_scale=None, color=None, fill=None, data=None, data2=None, warn_singular=True,
**kwargs)
```

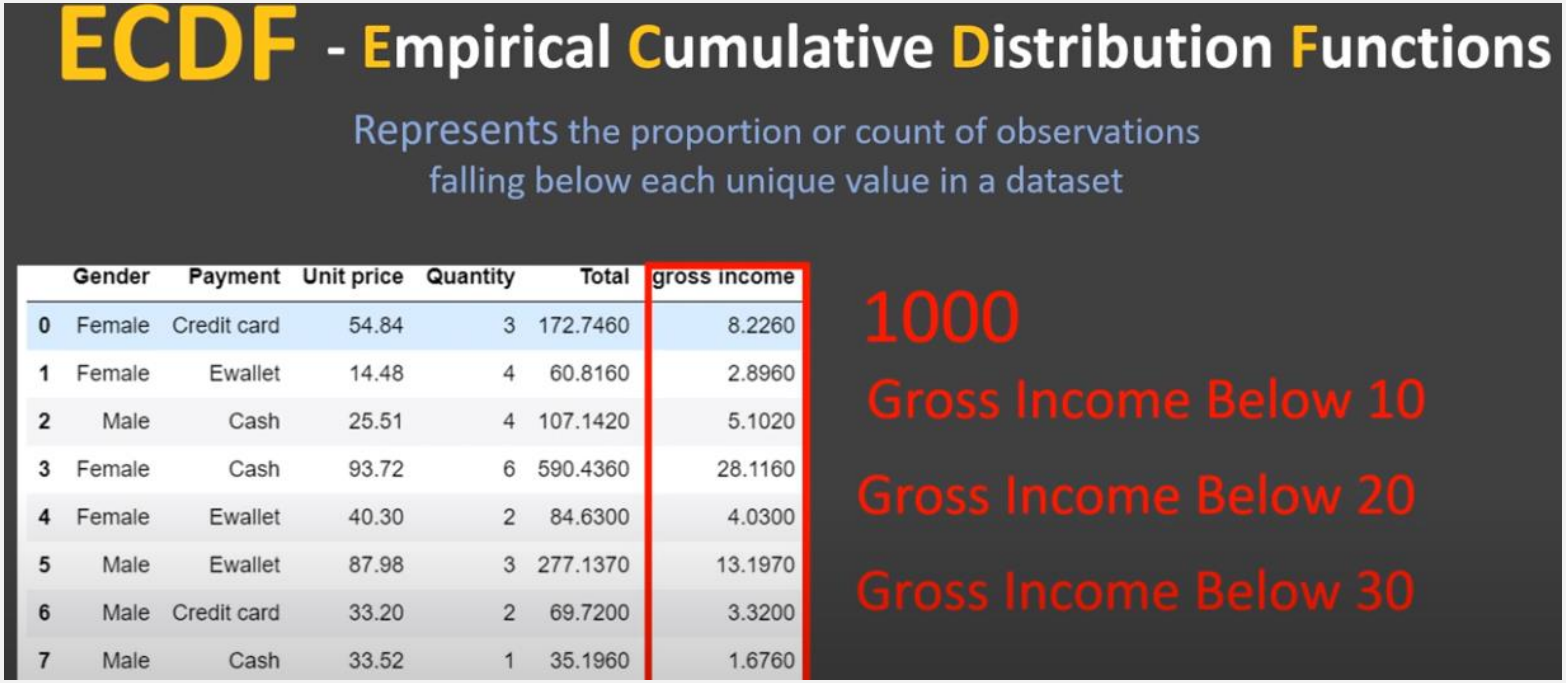
10. Rug Plot

Plot marginal distributions by drawing ticks along the x and y axes.

This function is intended to complement other plots by showing the location
of individual observations in an unobstrusive way.

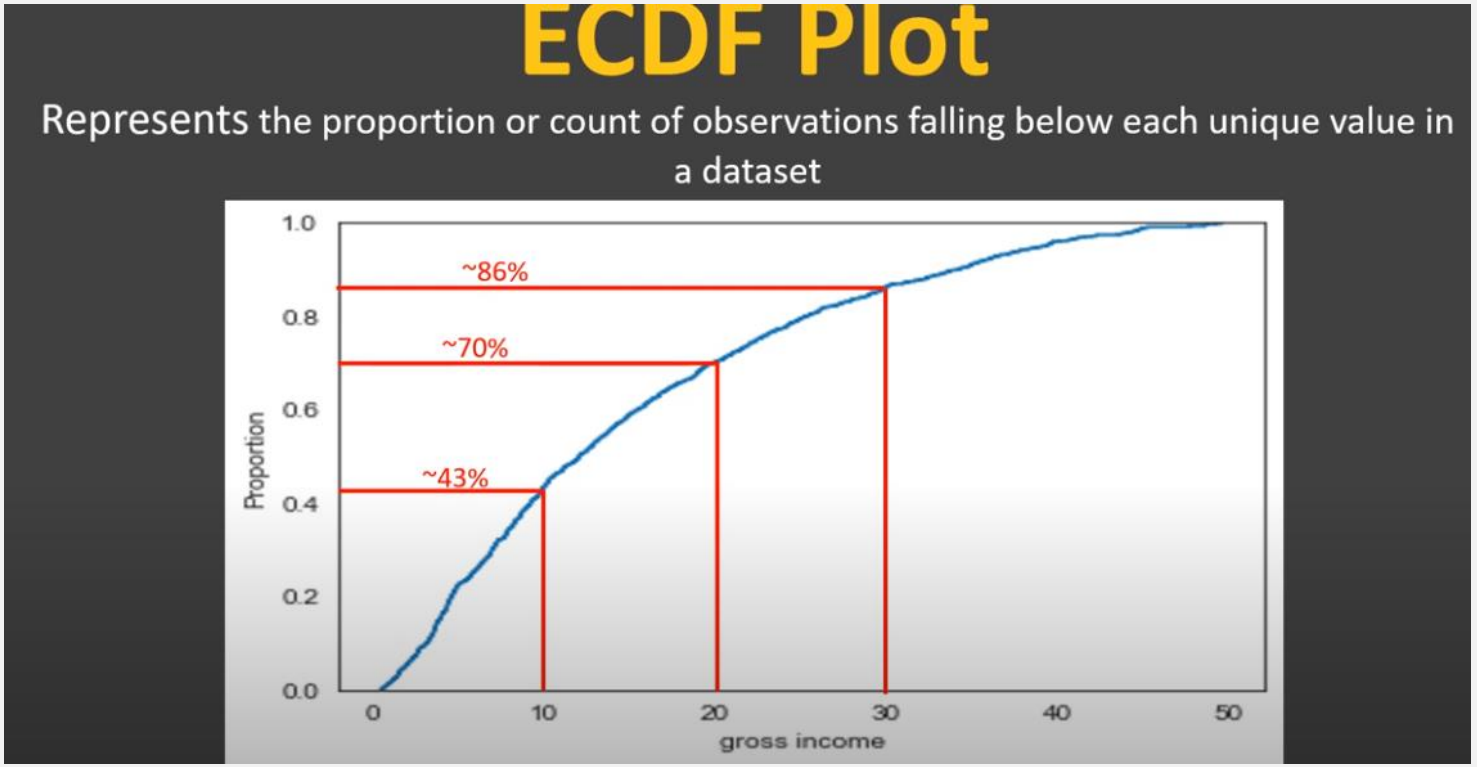


11. ECDF Plot

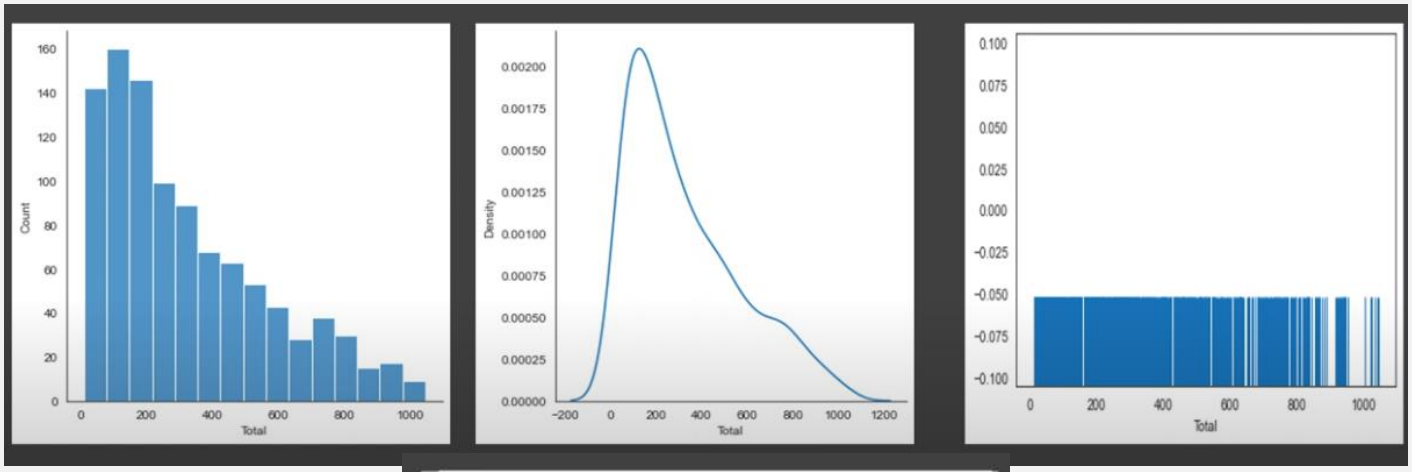


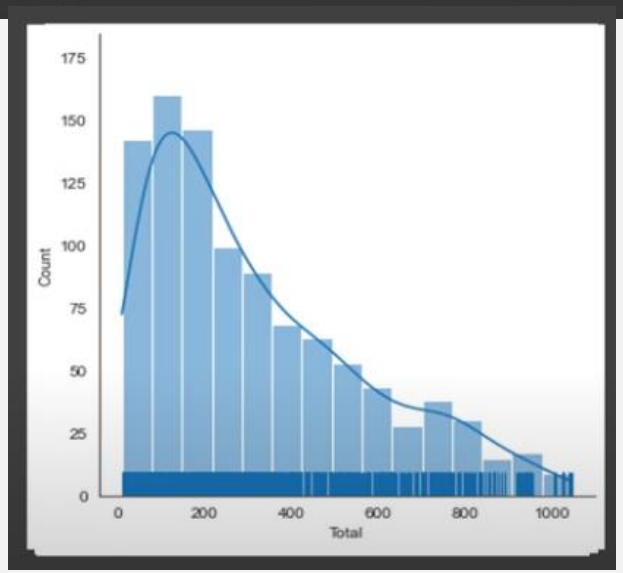
	Gender	Payment	Unit price	Quantity	Total	gross income
0	Female	Credit card	54.84	3	172.7460	8.2260
1	Female	Ewallet	14.48	4	60.8160	2.8960
2	Male	Cash	25.51	4	107.1420	5.1020
3	Female	Cash	93.72	6	590.4360	28.1160
4	Female	Ewallet	40.30	2	84.6300	4.0300
5	Male	Ewallet	87.98	3	277.1370	13.1970
6	Male	Credit card	33.20	2	69.7200	3.3200
7	Male	Cash	33.52	1	35.1960	1.6760

11. ECDF Plot


ECDF Plot
Represents the proportion or count of observations falling below each unique value in a dataset
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
~86%
~70%
~43%
0
10
20
30
40
50
gross income

12. Dis Plot





12. Dis Plot

```
seaborn.displot(data=None, *, x=None, y=None, hue=None, row=None,
                col=None, weights=None, kind='hist', rug=False,
                rug_kws=None, log_scale=None, legend=True, palette=None,
                hue_order=None, hue_norm=None, color=None,
                col_wrap=None, row_order=None, col_order=None, height=5,
                aspect=1, facet_kws=None, **kwargs)
```

13. Joint Plot

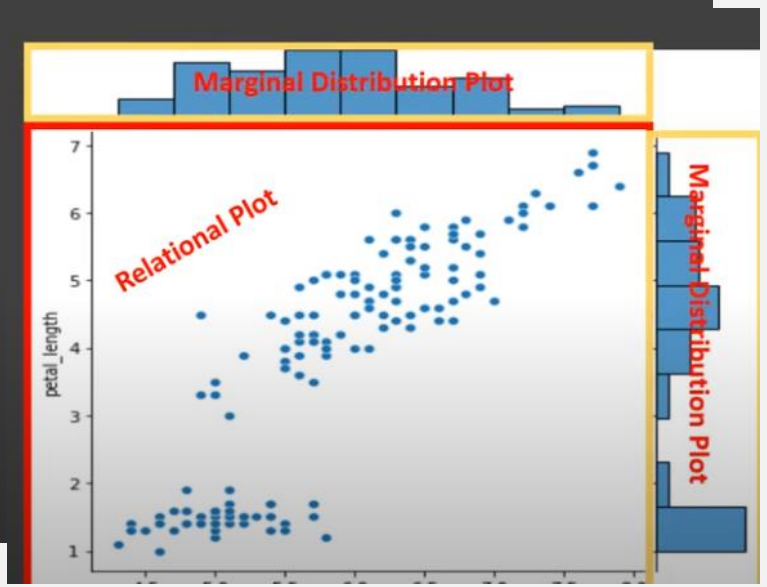
Joint Plot

Draw a plot of two variables with bivariate and univariate graphs

	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

R = 150

C = 5



14. Pair Plot

15.ScatterPlot

A diagram which shows the relationship between two variables by plotting the point/dots

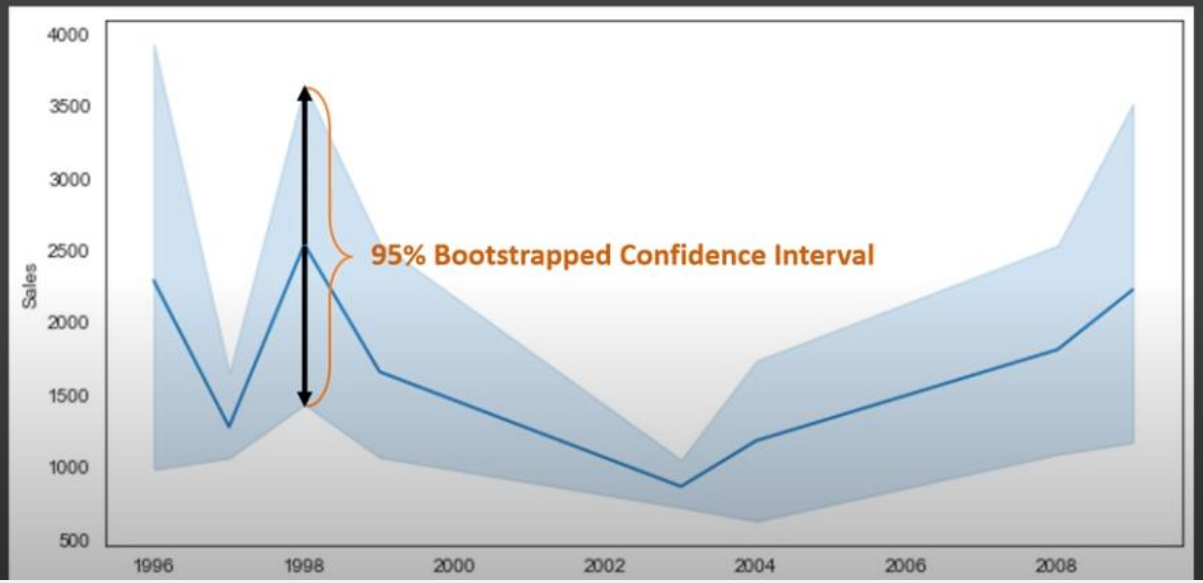
```
seaborn.scatterplot(*, x=None, y=None, hue=None, style=None, size=None,
                    data=None, palette=None, hue_order=None,
                    hue_norm=None, sizes=None, size_order=None,
                    size_norm=None, markers=True, style_order=None,
                    x_bins=None, y_bins=None, units=None,
                    estimator=None, ci=95, n_boot=1000, alpha=None,
                    x_jitter=None, y_jitter=None, legend='auto', ax=None,
                    **kwargs)
```

16. LinePlot

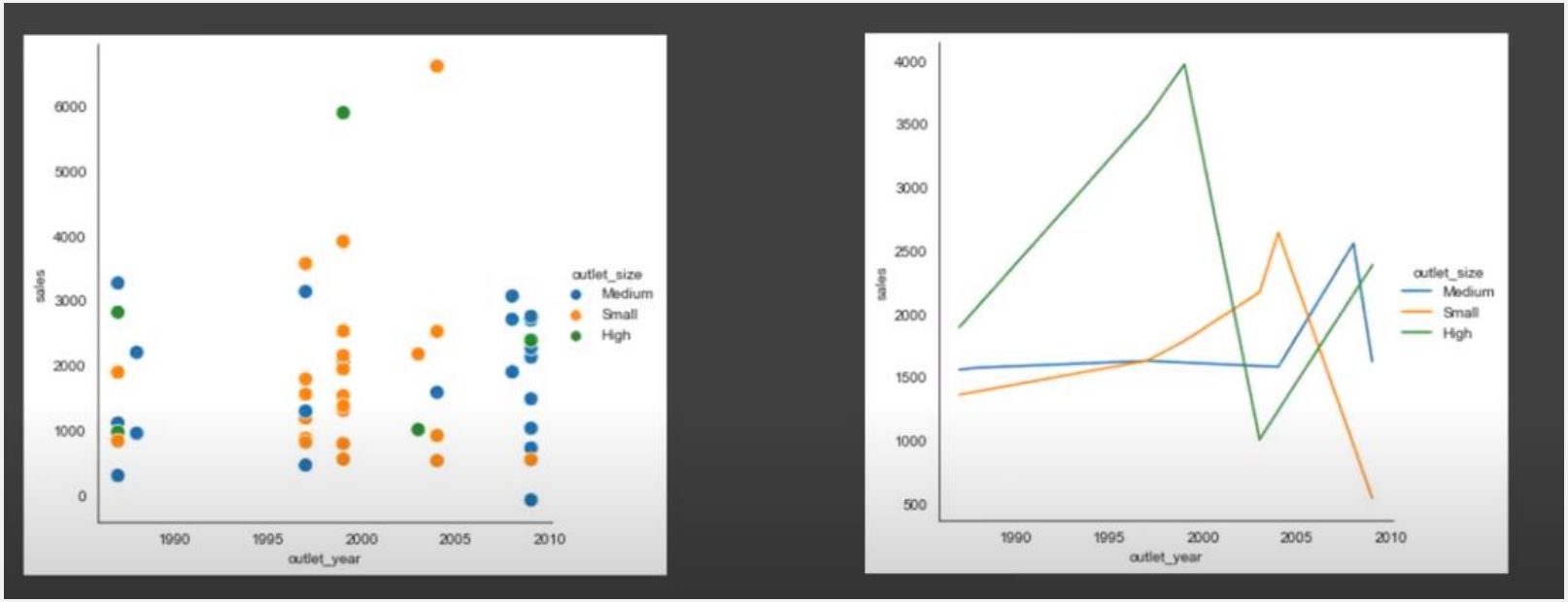
shows the relation between two variables

Majorly used in time series analysis

Showcases how the value of variable changes over the time



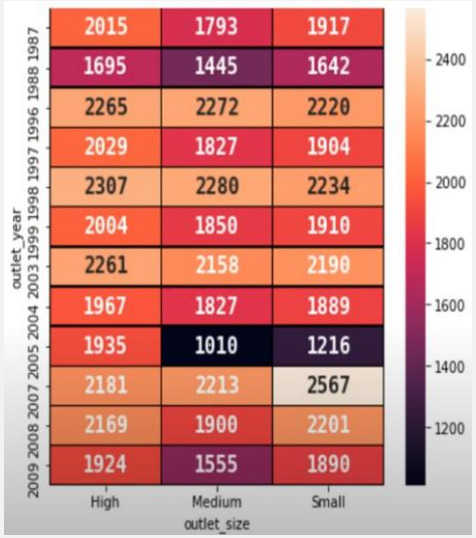
17. RelPlot


sales
6000
5000
4000
3000
2000
1000
0
1990
1995
2000
2005
2010
outlet_year
outlet_size
Medium
Small
High
sales
4000
3500
3000
2500
2000
1500
1000
500
1990
1995
2000
2005
2010
outlet_year
outlet_size
Medium
Small
High

HeatMap

	Item_ID	Item_W	Item_Type	Item_MRP	Outlet_ID	Outlet_Year	Outlet_Size	Outlet_Location_Type	Sales
0	FDU32	21.027499	Baking Goods	197.352319	OUT046	2004	Small	Tier 2	2689.457781
1	NCT54	21.102371	Meat	148.250214	OUT035	1987	Small	Tier 1	3437.350375
2	FDW08	20.882263	Hard Drinks	205.465010	OUT035	1999	Small	Tier 3	3129.967268
3	FDJ22	21.050435	Starchy Foods	253.417583	OUT046	1996	Small	Tier 1	1306.514376
4	FDF47	21.247876	Baking Goods	240.871039	OUT035	1988	Small	Tier 3	1739.769829
5	DRK12	20.956395	Baking Goods	130.264868	OUT049	1999	Small	Tier 1	1963.629422
6	FDA32	21.196562	Breads	239.259785	OUT035	1999	Small	Tier 2	581.887837
7	FDH24	20.949318	Hard Drinks	167.267122	OUT046	1997	Small	Tier 2	679.055015
8	FDW03	20.884811	Baking Goods	185.453864	OUT049	1997	Small	Tier 1	1991.320168
9	FDE11	21.183640	Others	239.191172	OUT018	2009	Small	Tier 1	730.148977

outlet_size	High	Medium	Small
outlet_year			
1987	2015.037160	1792.973492	1917.302712
1988	1695.209700	1444.865311	1641.739583
1996	2265.268983	2272.371502	2219.790139
1997	2029.428925	1826.732664	1903.967543
1998	2306.542273	2279.666103	2233.775392
1999	2004.082749	1850.282194	1909.920236
2003	2261.028030	2158.063891	2190.118601
2004	1966.898730	1826.582596	1889.009488
2005	1935.238262	1010.230431	1215.937098
2007	2180.578424	2213.387887	2567.411612
2008	2169.407763	1900.378559	2201.448849
2009	1923.770187	1554.601061	1890.260032



ClusterMap

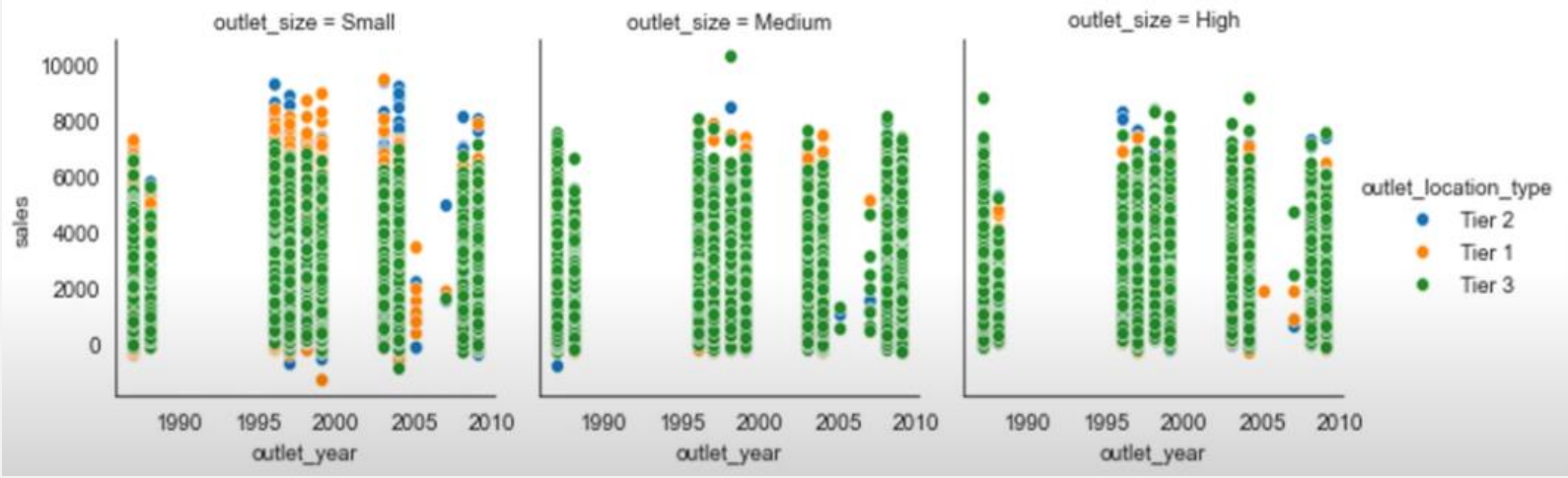
Plot a matrix dataset as a hierarchically-clustered heatmap

```
seaborn.clustermap(data, *, pivot_kws=None, method='average', metric='euclidean',
                   z_score=None, standard_scale=None, figsize=(10, 10),
                   cbar_kws=None, row_cluster=True, col_cluster=True,
                   row_linkage=None, col_linkage=None, row_colors=None,
                   col_colors=None, mask=None, dendrogram_ratio=0.2,
                   colors_ratio=0.03, cbar_pos=(0.02, 0.8, 0.05, 0.18), tree_kws=None,
                   **kwargs)
```

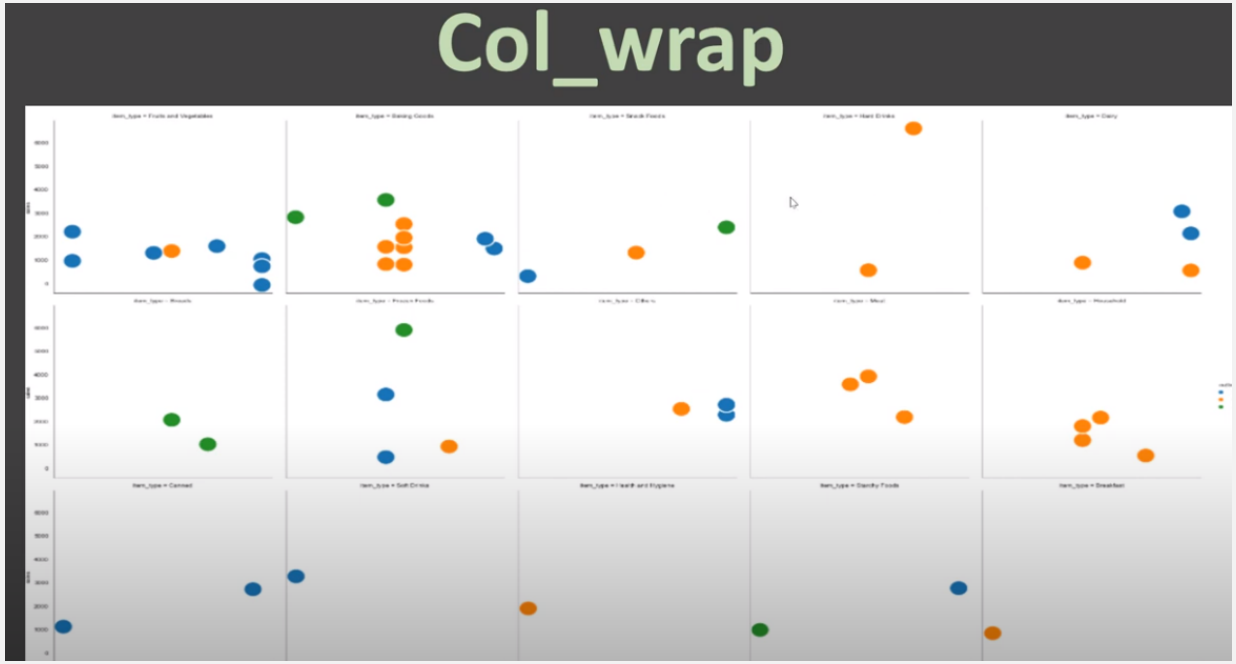
FacetGrid Plot

Multi-plot grid for plotting conditional relationships

```
seaborn.FacetGrid(self, data, *, row=None, col=None, hue=None, col_wrap=None,
                  sharex=True, sharey=True, height=3, aspect=1, palette=None,
                  row_order=None, col_order=None, hue_order=None,
                  hue_kws=None, dropna=False, legend_out=True, despine=True,
                  margin_titles=False, xlim=None, ylim=None, subplot_kws=None,
                  gridspec_kws=None, size=None)
```



RelPlot



Relational Plot - seaborn.relplot

- **This function provides access to several different axes-level functions that show the relationship between two variables with semantic mappings of subsets.**

  The kind parameter selects the underlying axes-level function to use:

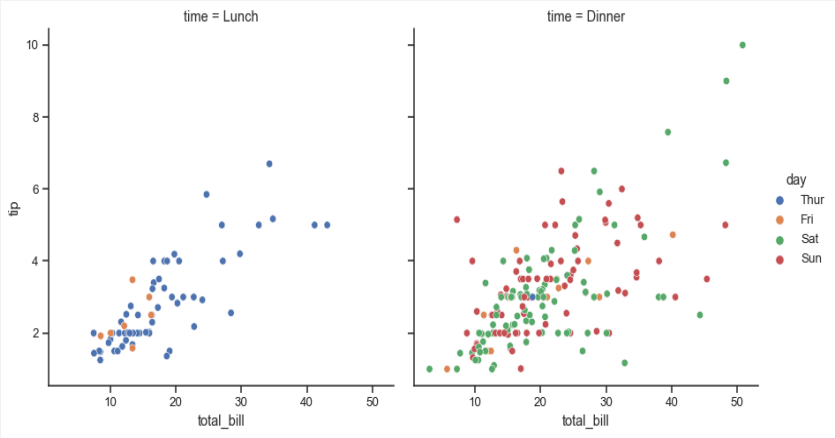
  scatterplot() (with kind="scatter"; the default)
  lineplot() (with kind="line")

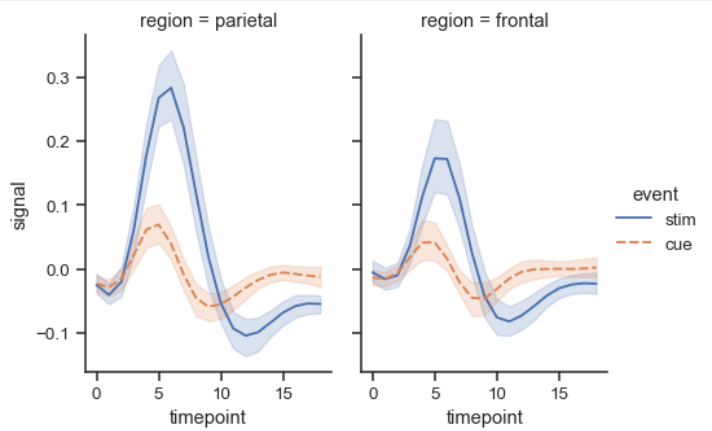
The relationship between x and y can be shown for different subsets of the data using the hue, size, and style parameters. These parameters control what visual semantics are used to identify the different subsets.

Relational Plot - seaborn.relplot

e.g.

sns.relplot(data=tips, x="total_bill", y="tip", hue="day")
sns.relplot(data=tips, x="total_bill", y="tip", hue="day", col="time")

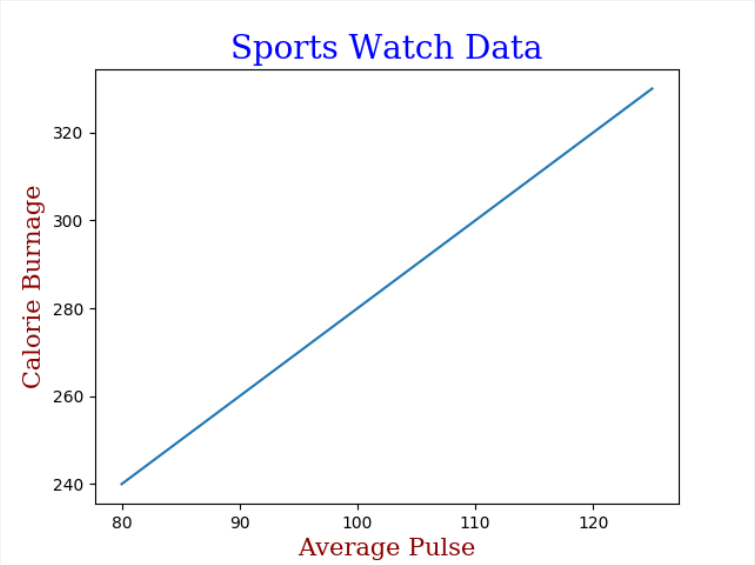




Labels

```
font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)
```



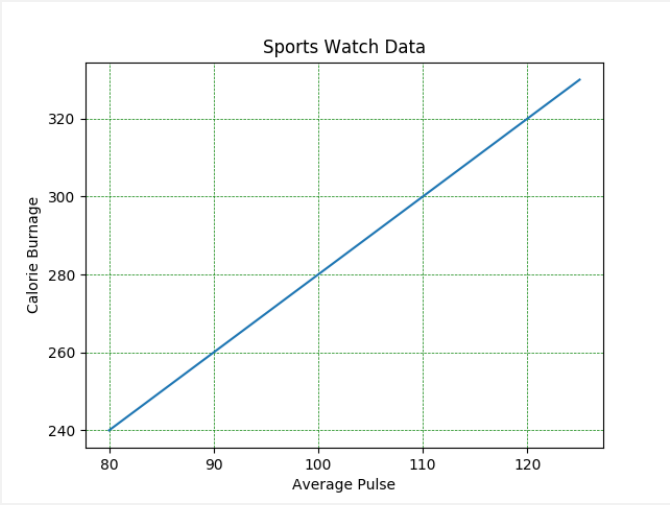
Grids

With Pyplot, you can use the grid() function to add grid lines to the plot

plt.grid()

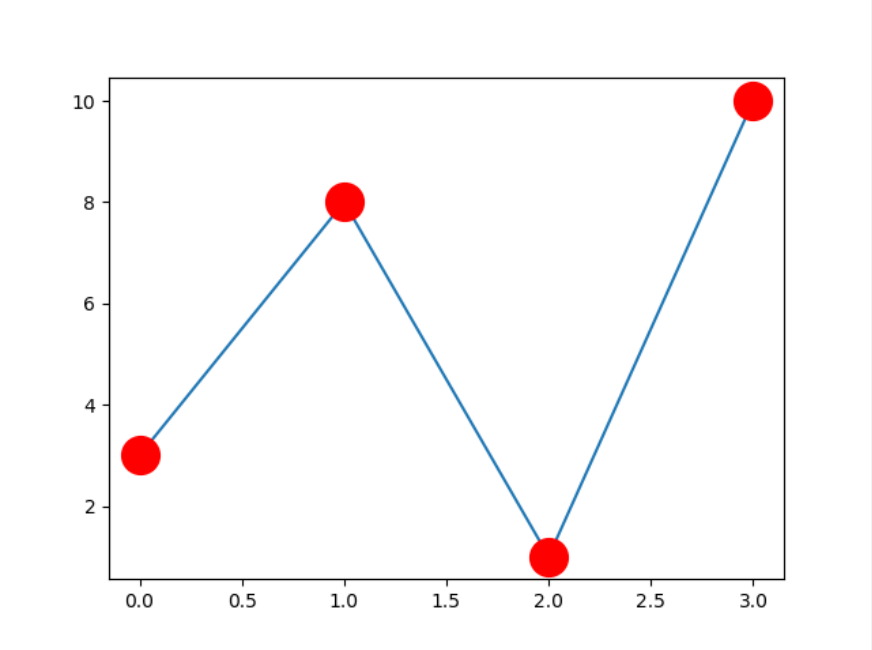
Or

plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)



Matplotlib Markers

plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r', mfc = 'r')

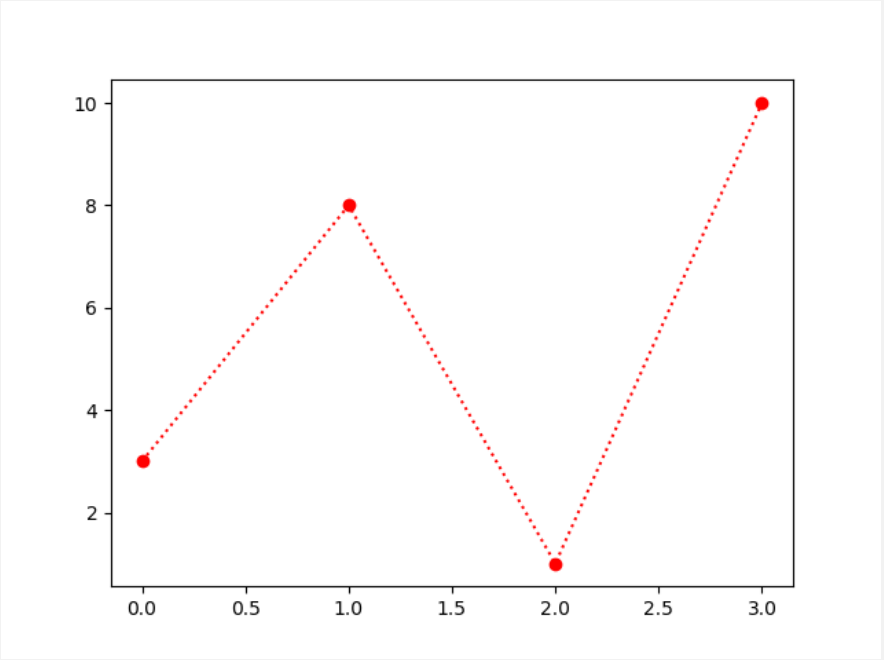


Matplotlib Markers Using format String

This parameter is also called fmt, and is written with this syntax:

marker|*line*|*color*

plt.plot(ypoints, 'o:r')



Matplotlib Lines

Line Style :

```
plt.plot(ypoints, linestyle = 'dotted')
```

```
plt.plot(ypoints, ls = ':')
```

Line Color :

```
plt.plot(ypoints, color = 'r')
```

```
plt.plot(ypoints, c = '#4CAF50')
```

Line Width :

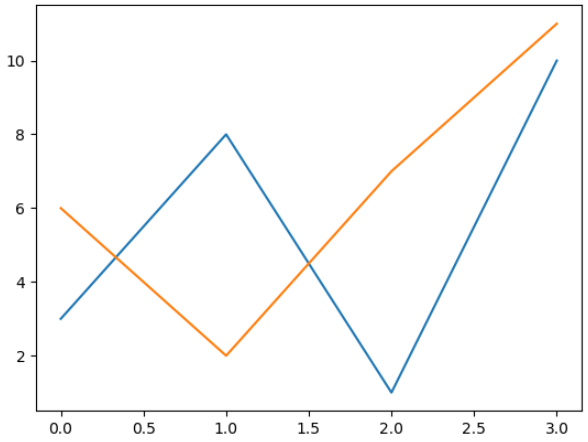
```
plt.plot(ypoints, linewidth = '20.5')
```

Multiple Lines

You can plot as many lines as you like by simply adding more plt.plot() functions:

```
y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])

plt.plot(y1)
plt.plot(y2)
```




```
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1, x2, y2)
plt.show()
```

Subplots

With the subplots() function you can draw multiple plots in one figure

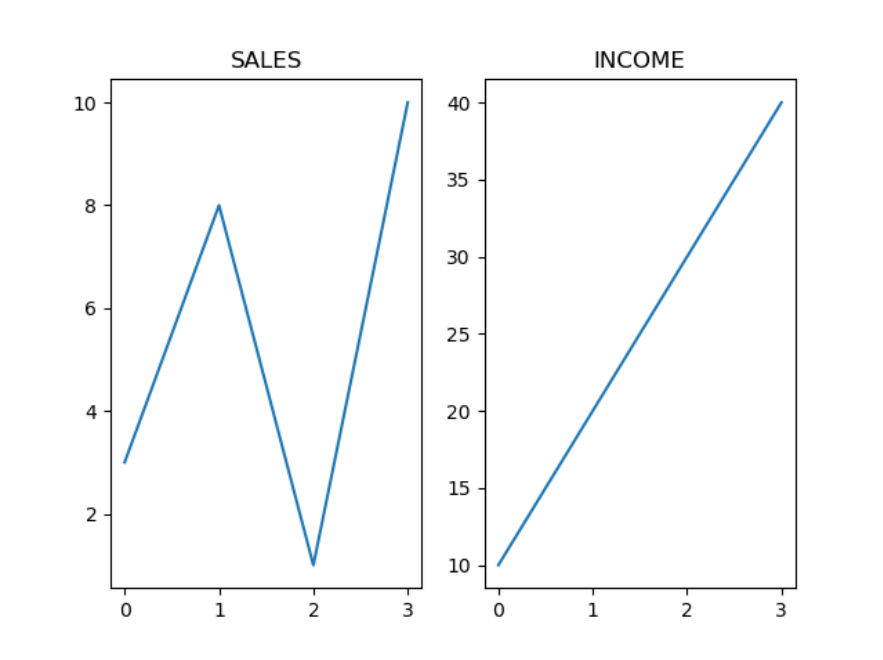
The subplots() function takes three arguments that describes the layout of the figure. The layout is organized in rows and columns, which are represented by the *first* and *second* argument. The third argument represents the index of the current plot.

```
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")
```

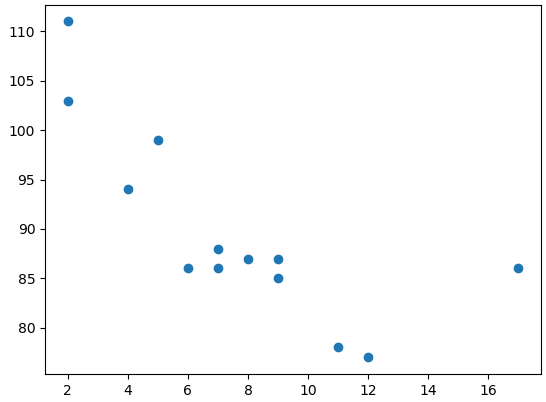


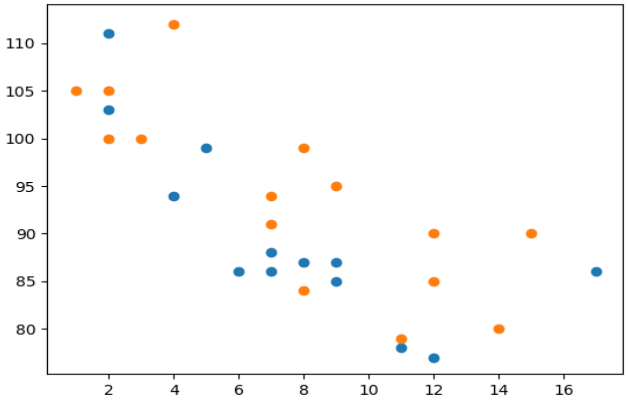
Matplotlib Scatter

With Pyplot, you can use the scatter() function to draw a scatter plot.

e.g.

```
plt.scatter(x, y)
```



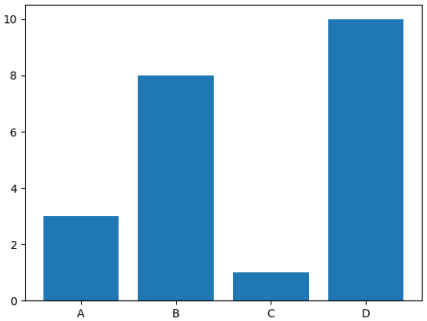


Matplotlib Bars

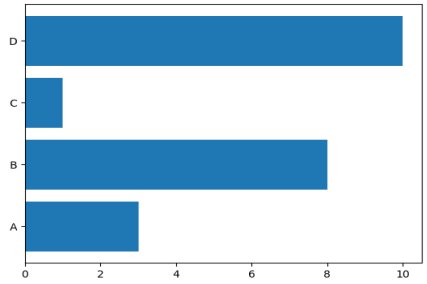
- **Creating Bars**

  With Pyplot, you can use the bar() function to draw bar graphs:

```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
```




```
plt.barh(x, y ,color ..  ,width  ..)
```

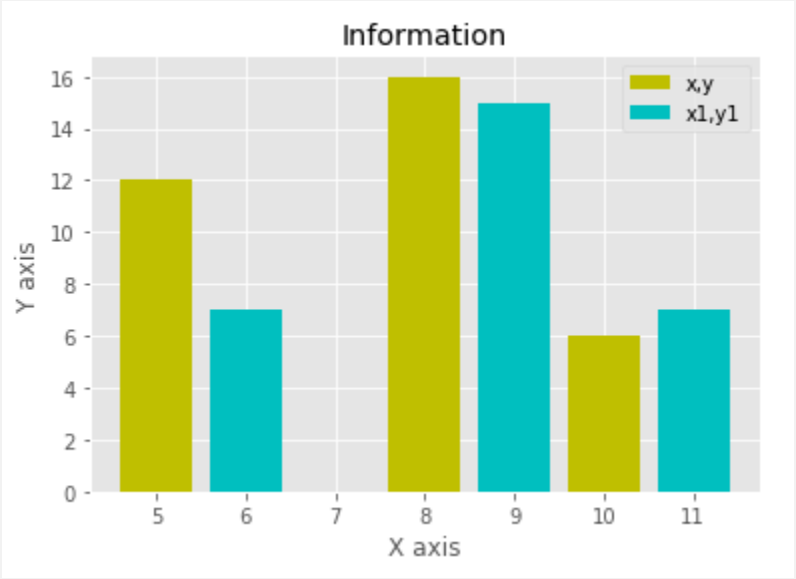


Matplotlib Bars Using Style

```
from matplotlib import style

style.use('ggplot')
x = [5,8,10]
y = [12,16,6]
x2 = [6,9,11]
y2 = [7,15,7]

plt.bar(x, y, color = 'y', align='center')
plt.bar(x2, y2, color='c', align='center')
```

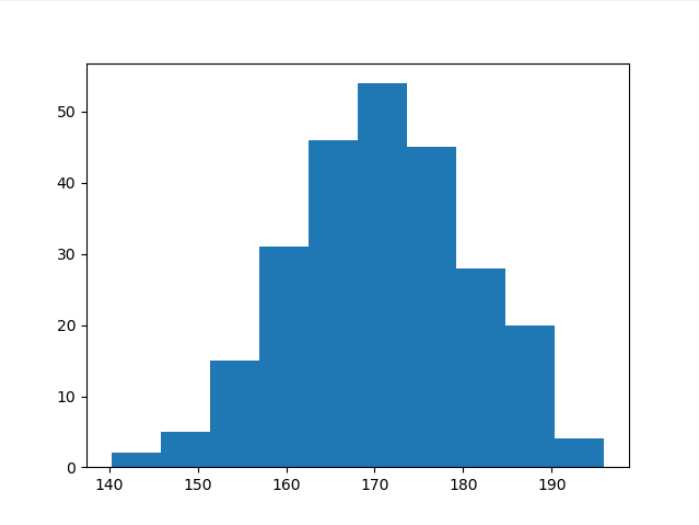


Matplotlib Histograms

- **A histogram is a graph showing *frequency* distributions.**
- **It is a graph showing the number of observations within each given interval.**

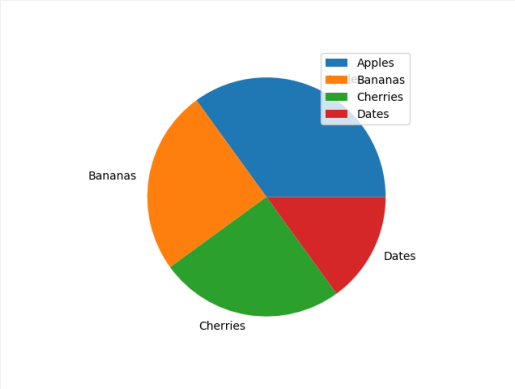
```
x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



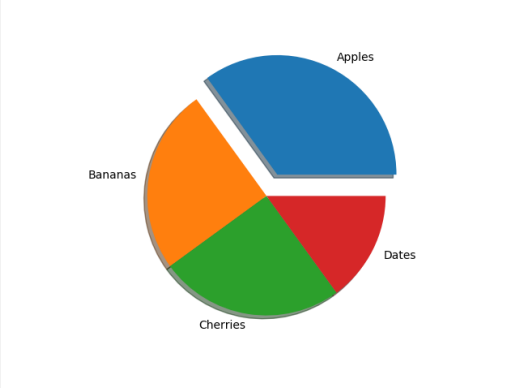
Matplotlib Pie Charts

- y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  plt.pie(y, labels = mylabels)
  plt.legend()




```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, explode = myexplode, shadow
= True)
```

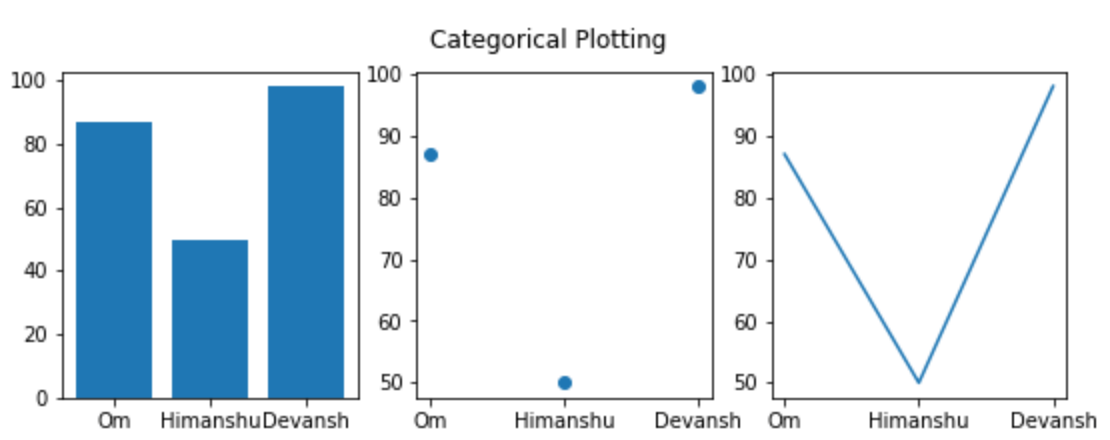


Plotting with categorical variables

names = ['Abhishek', 'Himanshu', 'Devansh']

marks= [87,50,98]

```
plt.figure(figsize=(9,3))
plt.subplot(131)
plt.bar(names, marks)
plt.subplot(132)
plt.scatter(names, marks)
plt.subplot(133)
plt.plot(names, marks)
plt.suptitle('Categorical Plotting')
```



THANK YOU !!!

Amol Patil - 9822291613