

Practical Application of Statistics Concepts

Shubham Verma

Linkedin <https://www.linkedin.com/in/shubham-verma-3968a5119>

Theory

1. stat 1: https://www.linkedin.com/posts/shubham-verma-3968a5119_stats-notes-1-activity-6971835885327716352-739b?utm_source=share&utm_medium=member_desktop
2. stat 2: https://www.linkedin.com/posts/shubham-verma-3968a5119_stats-day-2-activity-6972454558438502400-HFkm?utm_source=share&utm_medium=member_desktop
3. stat 3: https://www.linkedin.com/posts/shubham-verma-3968a5119_stats-3-activity-6974971613096140800-hkHU?utm_source=share&utm_medium=member_desktop
4. stat 4: https://www.linkedin.com/posts/shubham-verma-3968a5119_stats-4-activity-6974983725113630720-G8-U?utm_source=share&utm_medium=member_desktop

Credits Krish Naik Sir for Theory

```
In [1]: import numpy as np
import seaborn as sns
import statistics as stat
import matplotlib.pyplot as plt
import random
```

0.0 Tips dataset

```
In [2]: df = sns.load_dataset("tips")
```

```
In [3]: df.head()
```

```
Out[3]:
```

	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

```
In [4]: df.columns
```

```
Out[4]: Index(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')
```

```
In [5]: df_numeric_data = df[df.dtypes[df.dtypes != 'category'].index]
df_numeric_data.head()
```

```
Out[5]:
```

	total_bill	tip	size
0	16.99	1.01	2
1	10.34	1.66	3
2	21.01	3.50	3
3	23.68	3.31	2
4	24.59	3.61	4

1.0 Measure of Central Tendency

1.1 mean of all numeric columns

```
In [6]: np.mean(df_numeric_data,axis=0)
```

```
Out[6]: total_bill    19.785943  
        tip          2.998279  
        size         2.569672  
        dtype: float64
```

1.2 median of all numeric columns

```
In [7]: np.median(df_numeric_data, axis=0)
```

```
Out[7]: array([17.795,  2.9 ,  2.  ])
```

1.3 mode of all columns

```
In [8]: stat.mode(df['total_bill'])
```

```
Out[8]: 13.42
```

```
In [9]: stat.mode(df['tip'])
```

```
Out[9]: 2.0
```

```
In [10]: stat.mode(df['sex'])
```

```
Out[10]: 'Male'
```

```
In [11]: stat.mode(df['smoker'])
```

```
Out[11]: 'No'
```

```
In [12]: stat.mode(df['day'])
```

```
Out[12]: 'Sat'
```

```
In [13]: stat.mode(df['time'])
```

```
Out[13]: 'Dinner'
```

```
In [14]: stat.mode(df['size'])
```

```
Out[14]: 2
```

2.0 Measure of Dispersion

2.1 Variance

```
In [15]: stat.variance(df_numeric_data['total_bill'])
```

```
Out[15]: 79.25293861397827
```

```
In [16]: stat.variance(df_numeric_data['tip'])
```

```
Out[16]: 1.9144546380624705
```

```
In [17]: stat.variance(df_numeric_data['size'])
```

```
Out[17]: 0.9045908385616946
```

2.2 Standard Deviation

```
In [18]: stat.stdev(df_numeric_data['total_bill'])
```

```
Out[18]: 8.902411954856856
```

```
In [19]: stat.stdev(df_numeric_data['tip'])
```

```
Out[19]: 1.383638189001182
```

```
In [20]: stat.stdev(df_numeric_data['size'])
```

Out[20]: 0.9510998047322345

3.0 Five point summary

3.1 For total_bill Column

In [21]: `df_numeric_data['total_bill'].min()` *# zeroth Percentile or minimum value*

Out[21]: 3.07

In [22]: `np.percentile(df_numeric_data['total_bill'], 25)` *# 1st quartile*

Out[22]: 13.3475

In [23]: `np.percentile(df_numeric_data['total_bill'], 50)` *# 2nd quartile or median*

Out[23]: 17.795

In [24]: `np.percentile(df_numeric_data['total_bill'], 75)` *# 3rd quartile*

Out[24]: 24.127499999999998

In [25]: `df_numeric_data['total_bill'].max()` *# 100th Percentile or maximum value*

Out[25]: 50.81

3.2 For tip Column

In [26]: `df_numeric_data['tip'].min()` *# zeroth Percentile or minimum value*

Out[26]: 1.0

In [27]: `np.percentile(df_numeric_data['tip'], 25)` *# 1st quartile*

Out[27]: 2.0

In [28]: `np.percentile(df_numeric_data['tip'], 50)` *# 2nd quartile or median*

Out[28]: 2.9

In [29]: `np.percentile(df_numeric_data['tip'], 75)` *# 3rd quartile*

Out[29]: 3.5625

In [30]: `df_numeric_data['tip'].max()` *# 100th Percentile or maximum value*

Out[30]: 10.0

3.3 For size Column

In [31]: `df_numeric_data['size'].min()` *# zeroth Percentile or minimum value*

Out[31]: 1

In [32]: `np.percentile(df_numeric_data['size'], 25)` *# 1st quartile*

Out[32]: 2.0

In [33]: `np.percentile(df_numeric_data['size'], 50)` *# 2nd quartile or median*

Out[33]: 2.0

In [34]: `np.percentile(df_numeric_data['size'], 75)` *# 3rd quartile*

Out[34]: 3.0

In [35]: `df_numeric_data['size'].max()` *# 100th Percentile or maximum value*

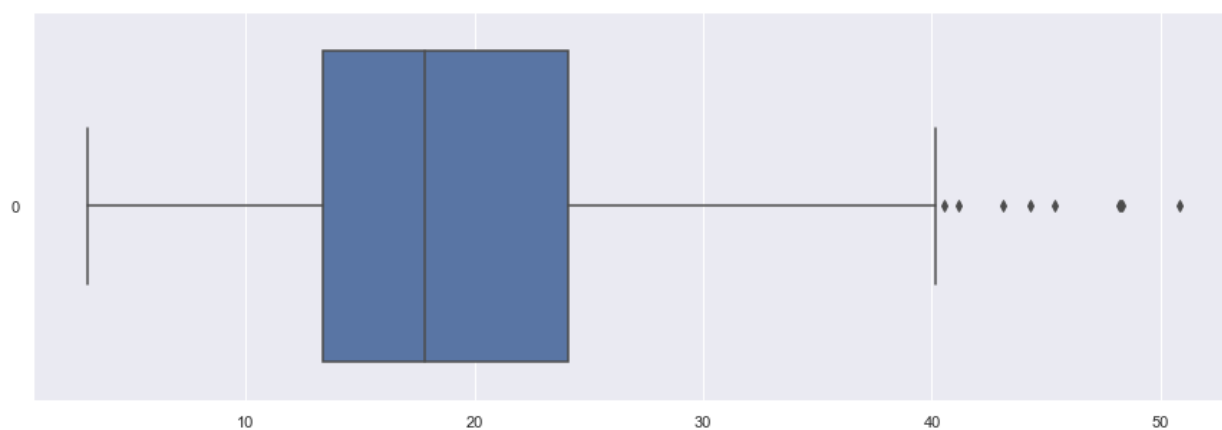
Out[35]: 6

4.0 Box Plot

4.1 Box plot for total_bill

```
In [36]: sns.set(rc={'figure.figsize':(15,5)})  
sns.boxplot(data = df_numeric_data['total_bill'], orient="h")
```

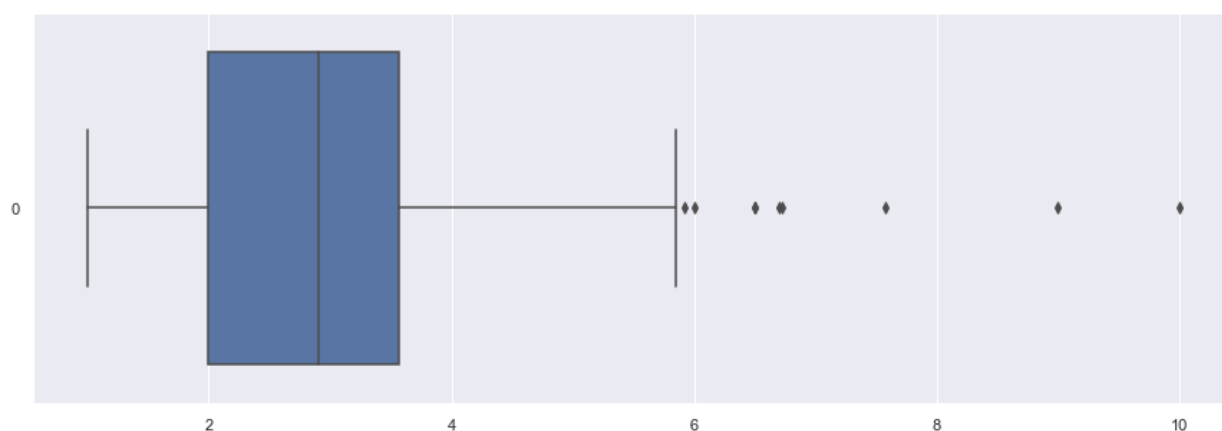
Out[36]: <AxesSubplot:>



4.2 Box plot for tip

```
In [37]: sns.set(rc={'figure.figsize':(15,5)})  
sns.boxplot(data = df_numeric_data['tip'], orient="h")
```

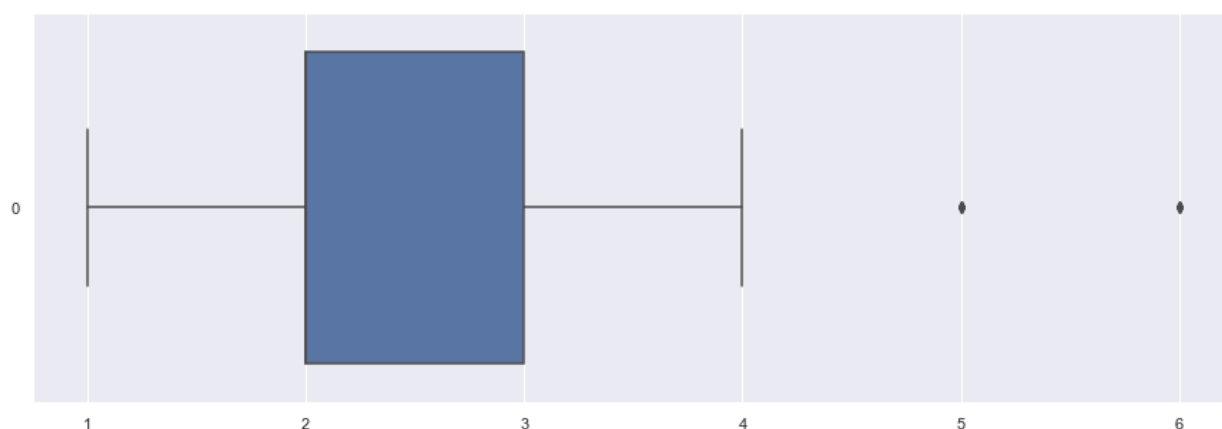
Out[37]: <AxesSubplot:>



4.3 Box plot for size

```
In [38]: sns.set(rc={'figure.figsize':(15,5)})  
sns.boxplot(data = df_numeric_data['size'], orient="h")
```

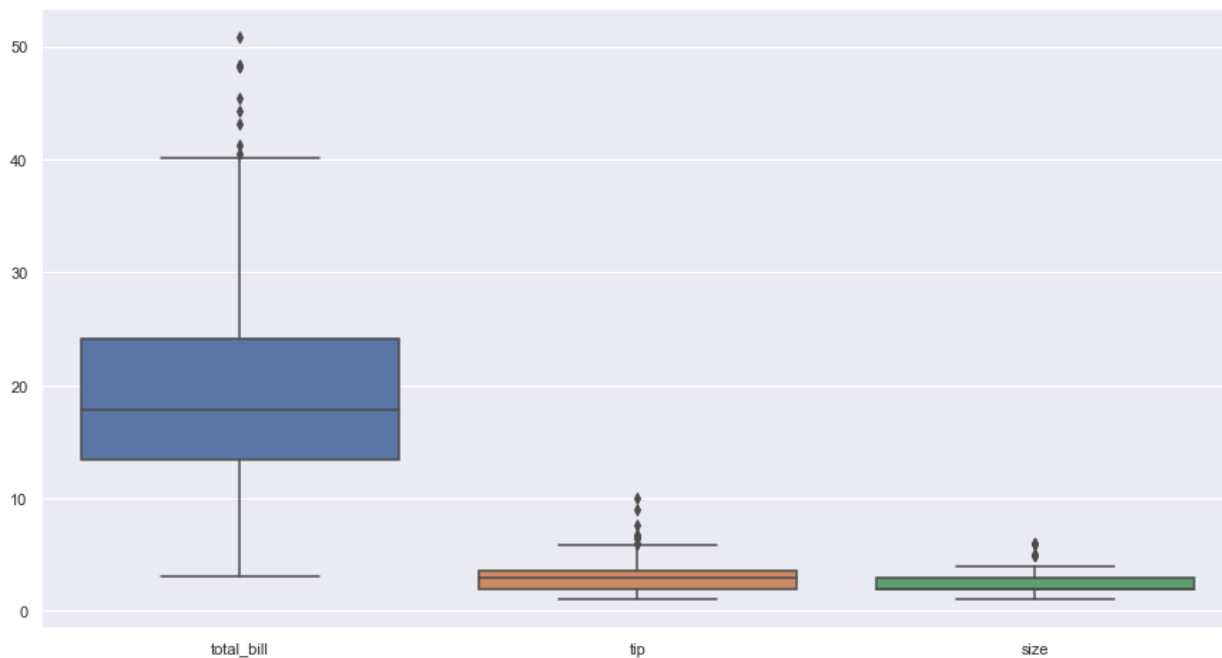
Out[38]: <AxesSubplot:>



4.4 Box plot combined

```
In [39]: sns.set(rc={'figure.figsize':(15,8)})  
sns.boxplot(data = df_numeric_data, orient="v")
```

Out[39]: <AxesSubplot:>

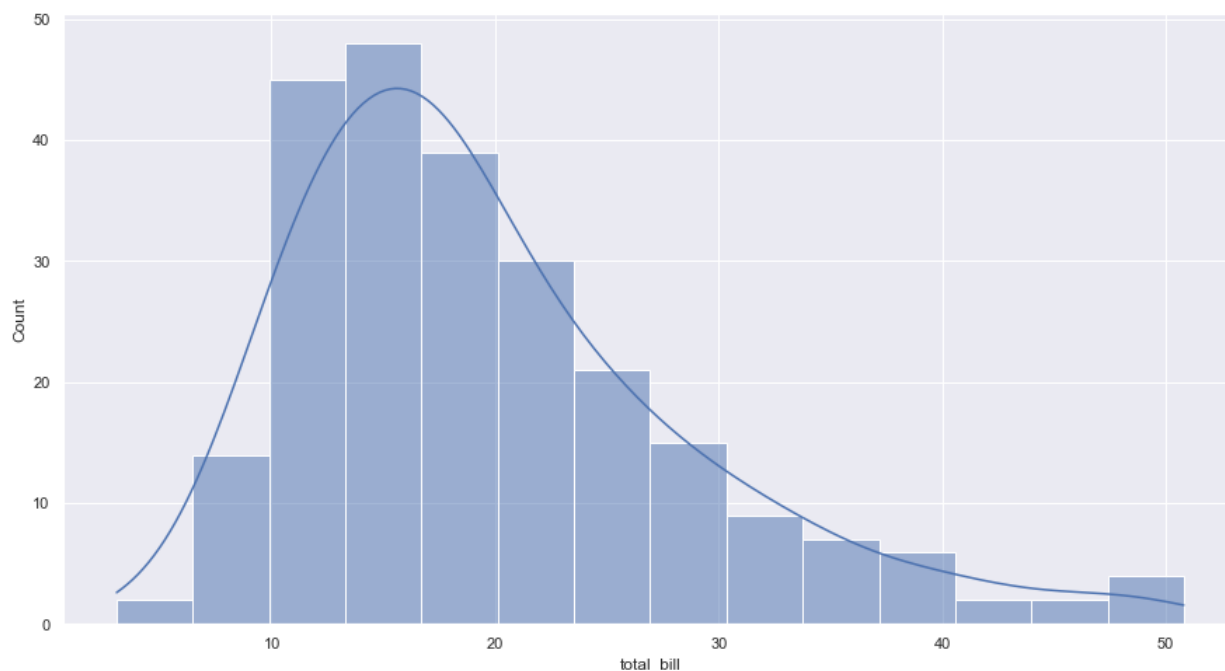


5.0 Histogram and Distribution

5.1 Histogram and distribution for total_bill

```
In [40]: sns.histplot(df_numeric_data['total_bill'], kde=True)
```

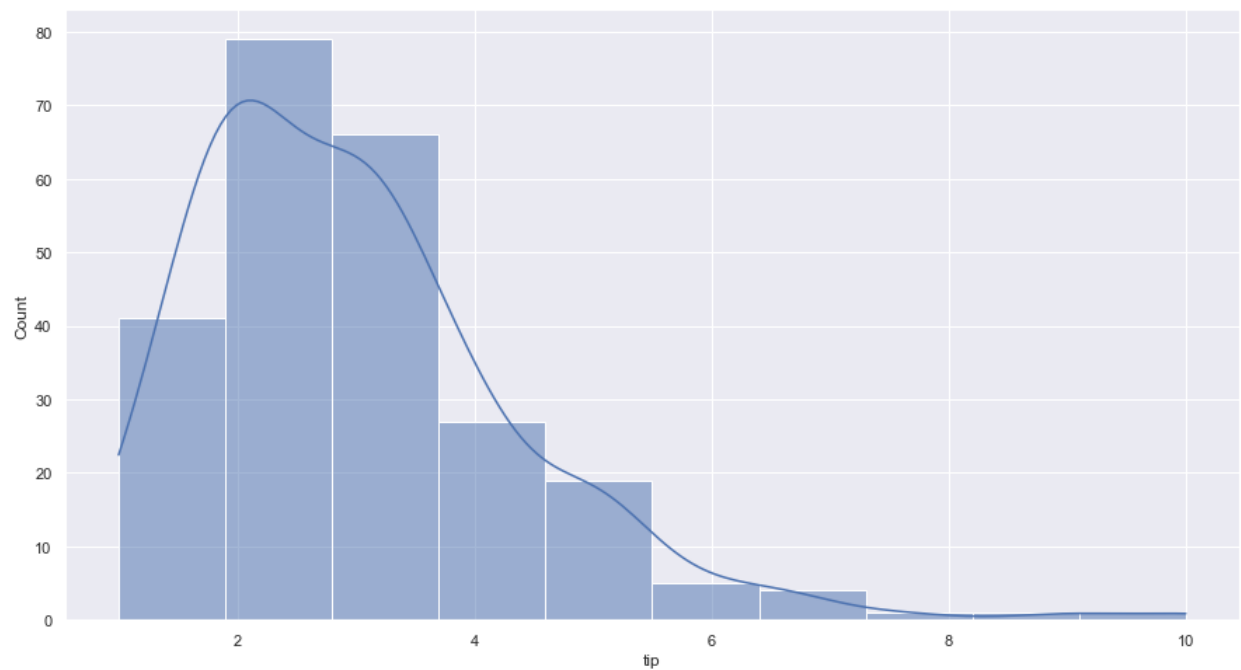
Out[40]: <AxesSubplot:xlabel='total_bill', ylabel='Count'>



5.2 Histogram and distribution for tip

```
In [41]: sns.histplot(df_numeric_data['tip'], kde=True, bins=10)
```

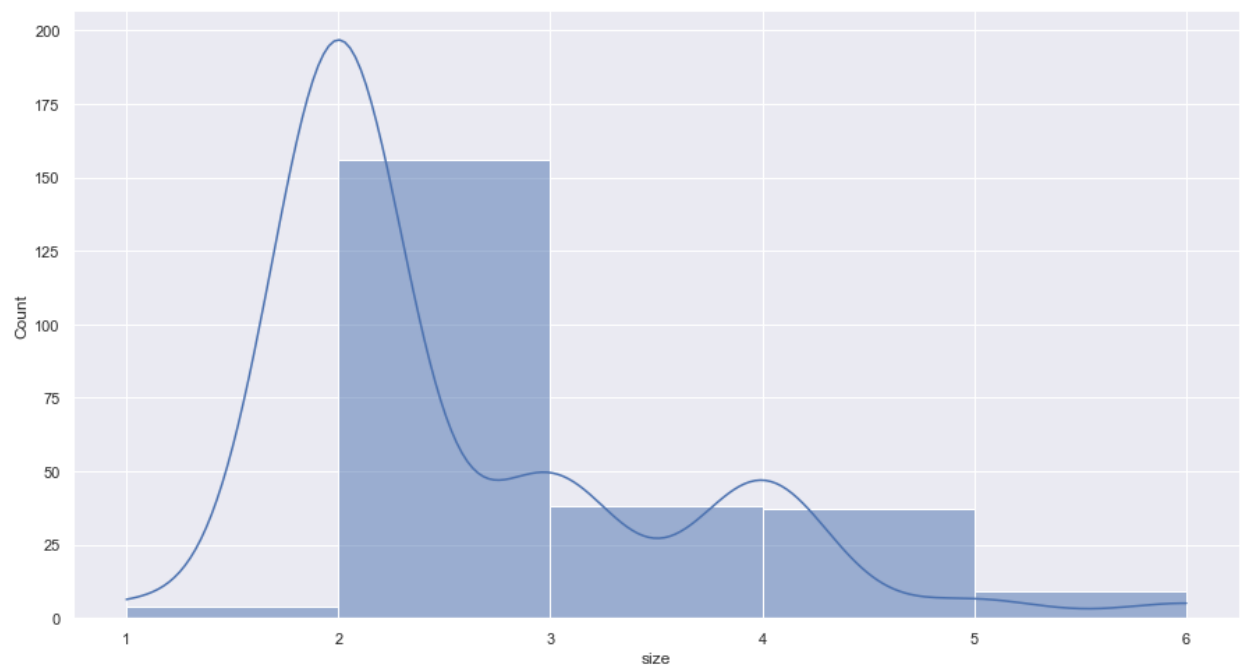
Out[41]: <AxesSubplot:xlabel='tip', ylabel='Count'>



5.3 Histogram and distribution for size

```
In [42]: sns.histplot(df_numeric_data['size'], kde=True, bins=5)
```

```
Out[42]: <AxesSubplot:xlabel='size', ylabel='Count'>
```



6.0 Standardization

```
In [43]: df_numeric_data.head()
```

```
Out[43]:
```

	total_bill	tip	size
0	16.99	1.01	2
1	10.34	1.66	3
2	21.01	3.50	3
3	23.68	3.31	2
4	24.59	3.61	4

6.1 standardization of dataset

```
In [44]: df_numeric_data_std = (df_numeric_data - df_numeric_data.mean())/df_numeric_data.std()
```

```
In [45]: df_numeric_data_std.head()
```

```
Out[45]:
```

	total_bill	tip	size
0	-0.314066	-1.436993	-0.598961
1	-1.061054	-0.967217	0.452453
2	0.137497	0.362610	0.452453
3	0.437416	0.225291	-0.598961
4	0.539635	0.442111	1.503867

6.2 mean and standard deviation of standardized dataset

```
In [46]: round(df_numeric_data_std.mean(), 2)
```

```
Out[46]:
```

total_bill	-0.0
tip	-0.0
size	-0.0

dtype: float64

```
In [47]: df_numeric_data_std.std()
```

```
Out[47]:
```

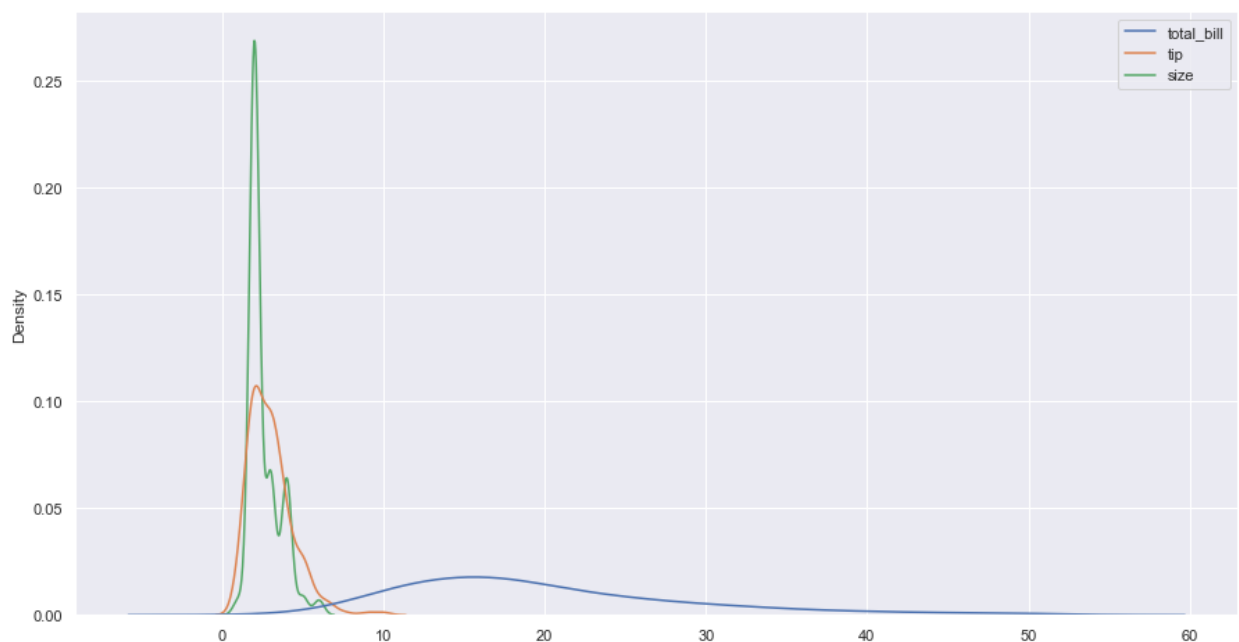
total_bill	1.0
tip	1.0
size	1.0

dtype: float64

6.3 kde plot of original dataset

```
In [48]: sns.kdeplot(data = df_numeric_data) # kde plot for numeric data
```

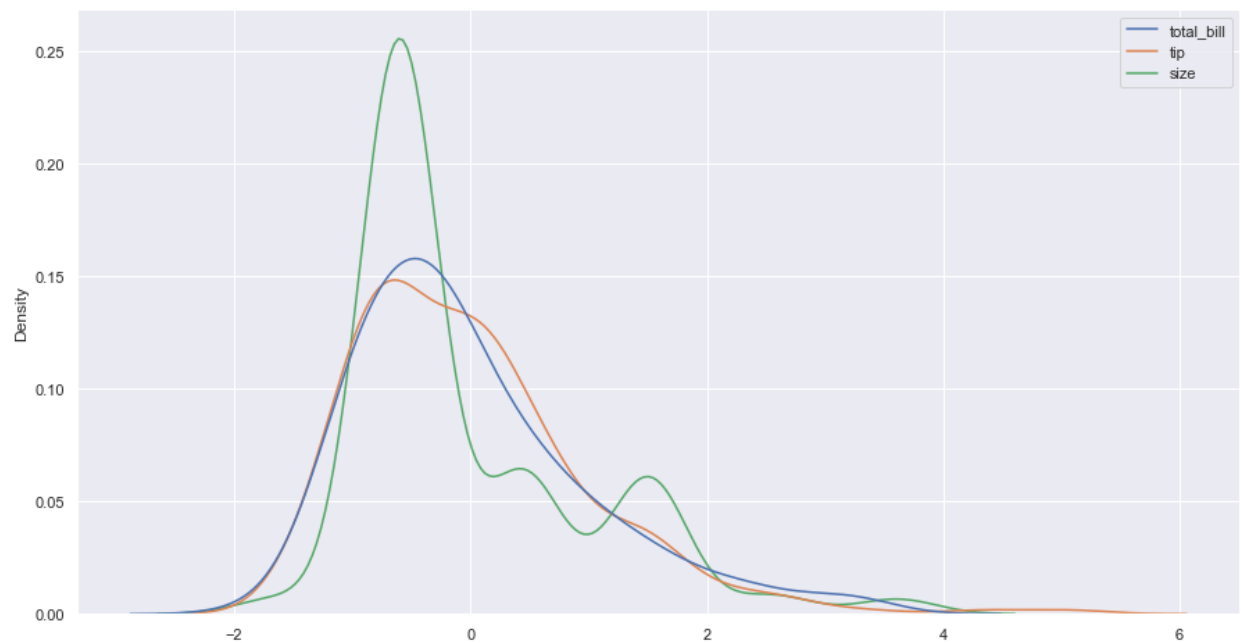
```
Out[48]: <AxesSubplot:ylabel='Density'>
```



6.4 kde plot of standardized dataset

```
In [49]: sns.kdeplot(data = df_numeric_data_std) # kde plot for numeric data in tips data set after standardization
```

```
Out[49]: <AxesSubplot:ylabel='Density'>
```



7.0 Normalization

Using MinMax Scalar

Note: using MinMax Scalar the data range will be 0 to 1.

```
In [50]: df_numeric_data.head()
```

```
Out[50]:
```

	total_bill	tip	size
0	16.99	1.01	2
1	10.34	1.66	3
2	21.01	3.50	3
3	23.68	3.31	2
4	24.59	3.61	4

7.1 normalization of dataset

```
In [51]: df_numeric_data_normal = (df_numeric_data - df_numeric_data.min()) / (df_numeric_data.max() - df_numeric_data
```

```
In [52]: df_numeric_data_normal.head()
```

```
Out[52]:
```

	total_bill	tip	size
0	0.291579	0.001111	0.2
1	0.152283	0.073333	0.4
2	0.375786	0.277778	0.4
3	0.431713	0.256667	0.2
4	0.450775	0.290000	0.6

7.2 min, max, mean and standard deviation of normalized dataset

```
In [53]: df_numeric_data_normal.min() # minimum value of data
```

```
Out[53]:
```

total_bill	0.0
tip	0.0
size	0.0
dtype:	float64

```
In [54]: df_numeric_data_normal.max() # maximum value of data
```



```
Out[54]: total_bill    1.0  
         tip          1.0  
         size         1.0  
         dtype: float64
```

```
In [55]: df_numeric_data_normal.mean()
```

```
Out[55]: total_bill    0.350145  
         tip          0.222031  
         size         0.313934  
         dtype: float64
```

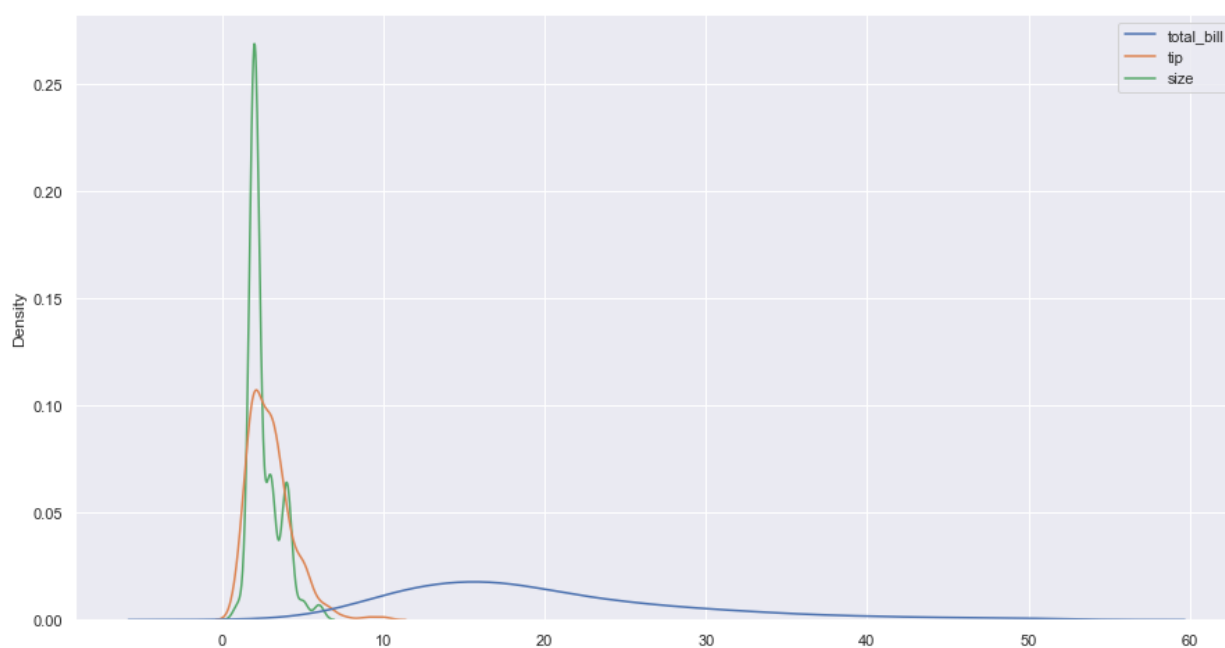
```
In [56]: df_numeric_data_normal.std()
```

```
Out[56]: total_bill    0.186477  
         tip          0.153738  
         size         0.190220  
         dtype: float64
```

7.3 kde plot of original dataset

```
In [57]: sns.kdeplot(data = df_numeric_data) # kde plot for numeric data
```

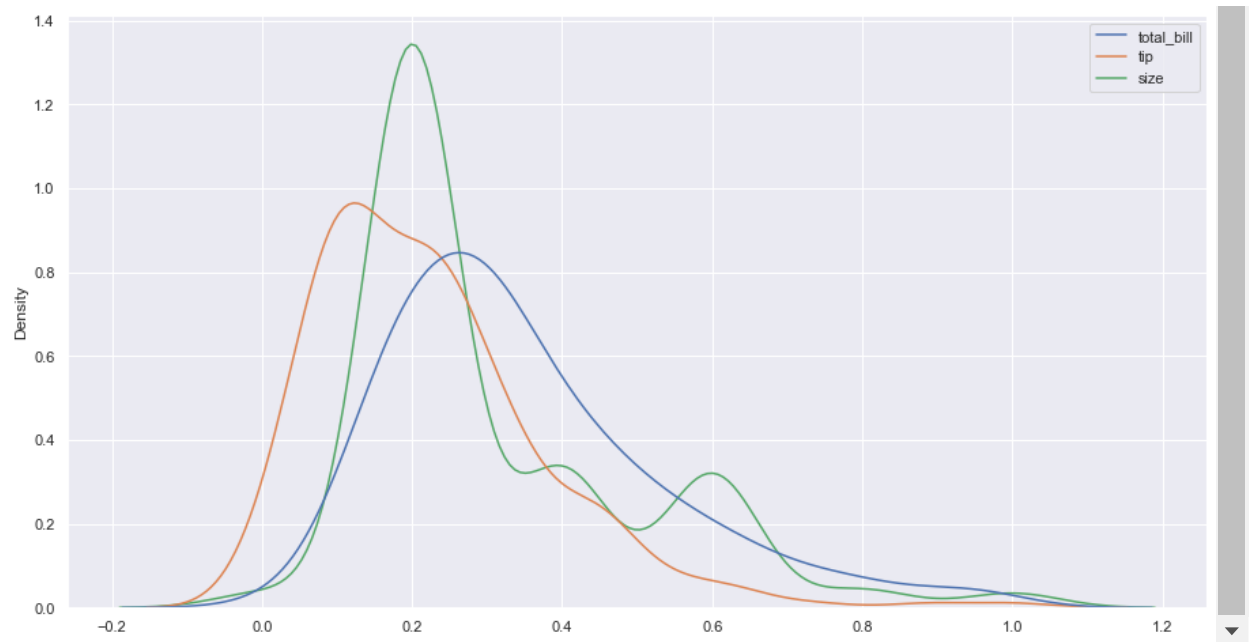
```
Out[57]: <AxesSubplot:ylabel='Density'>
```



7.4 kde plot of normalized dataset

```
In [58]: sns.kdeplot(data = df_numeric_data_normal) # kde plot for numeric data in tips data set after normalization
```

```
Out[58]: <AxesSubplot:ylabel='Density'>
```



8.0 Central Limit Theorem

In [59]: `df_numeric_data.head()`

Out[59]:

	total_bill	tip	size
0	16.99	1.01	2
1	10.34	1.66	3
2	21.01	3.50	3
3	23.68	3.31	2
4	24.59	3.61	4

8.1 for total_bill

In [60]:

```
mean_pop_total_bill = df_numeric_data['total_bill'].mean()
std_pop_total_bill = df_numeric_data['total_bill'].std()

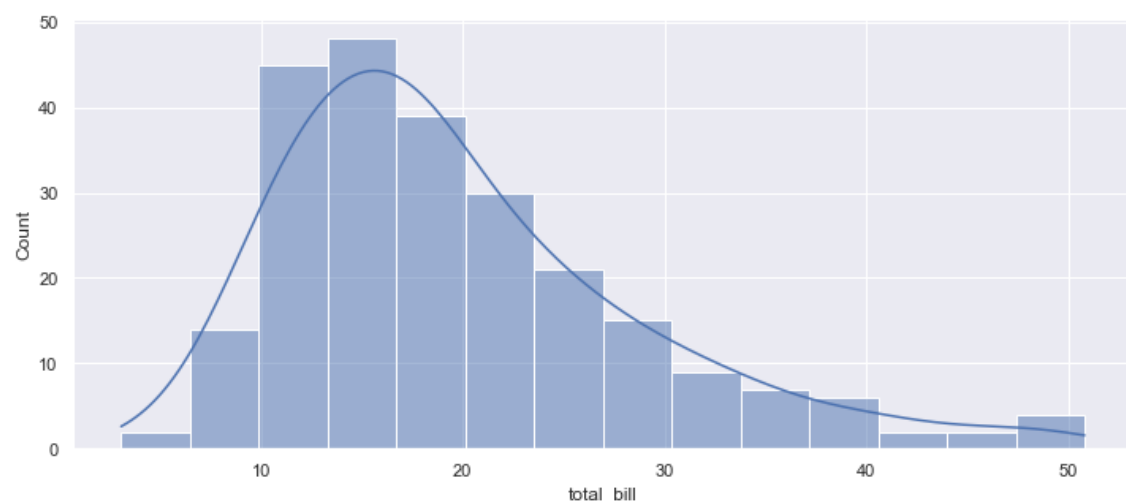
print("population mean ( $\mu$ ): {} \ npopulation standard deviation ( $\sigma$ ): {}".format(mean_pop_total_bill, std_pop_tot
```

population mean (μ): 19.785942622950824
 population standard deviation (σ): 8.902411954856856

In [61]:

```
sns.set(rc={'figure.figsize':(12,5)})
sns.histplot(df_numeric_data['total_bill'], kde=True)
```

Out[61]: <AxesSubplot:xlabel='total_bill', ylabel='Count'>



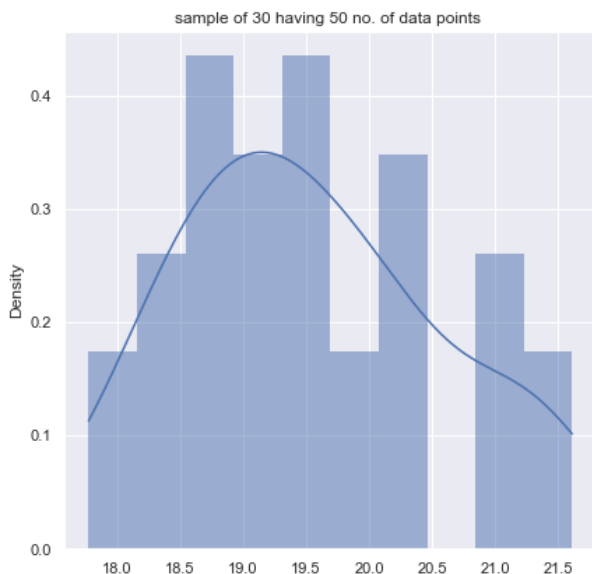
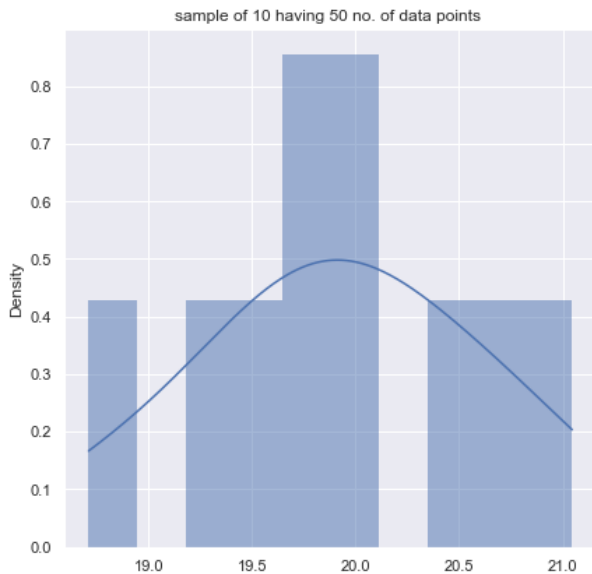
In [62]: `def mean_distribution(data, samples_count, data_points_count):`

```

list_sample = list()
data = np.array(data.values)
for i in range(0, samples_count):
    samples = random.sample(range(0, data.shape[0]), data_points_count)
    list_sample.append(data[samples].mean())
return np.array(list_sample)

count = 0
mean_list = list()
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(15, 15))
lst = [(10,50),(20,50),(30,50),(50,50)]
for i in (0,1):
    for j in (0,1):
        ax[i,j].set_title("sample of " + str(lst[count][0]) + " having " + str(lst[count][1])+" no. of data p
        sns.histplot(mean_distribution(df_numeric_data['total_bill'], lst[count][0],lst[count][1]),ax = ax[i,
        mean_list.append(mean_distribution(df_numeric_data['total_bill'], lst[count][0],lst[count][1]))
        count +=1

```



8.2 for tip

```

In [63]: mean_pop_tip = df_numeric_data['tip'].mean()
std_pop_tip = df_numeric_data['tip'].std()

print("population mean ( $\mu$ ): {} \n population standard deviation ( $\sigma$ ): {}".format(mean_pop_tip, std_pop_tip))

population mean ( $\mu$ ): 2.9982786885245902
population standard deviation ( $\sigma$ ): 1.3836381890011826

```

```

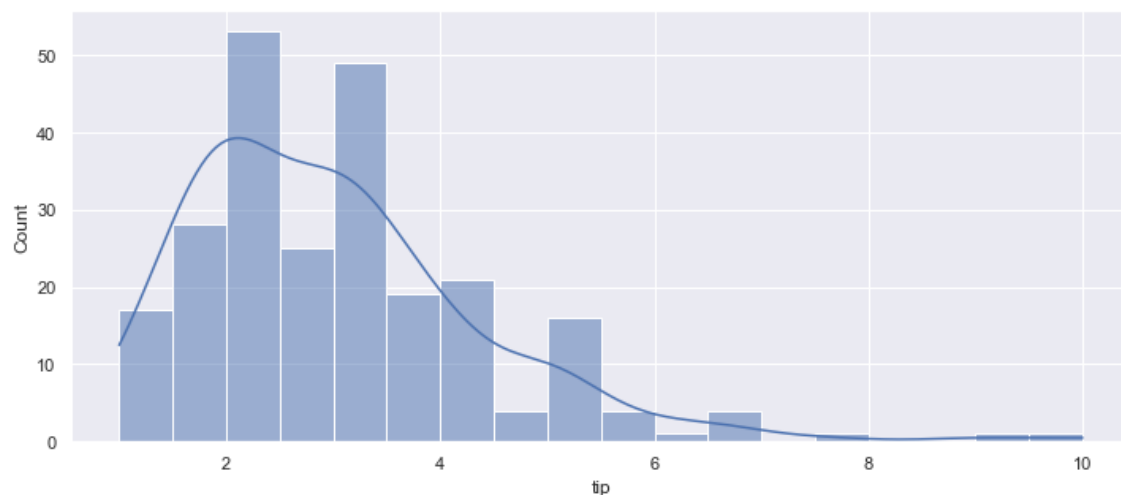
In [64]: sns.set(rc={'figure.figsize':(12,5)})
sns.histplot(df_numeric_data['tip'], kde=True)

```

```

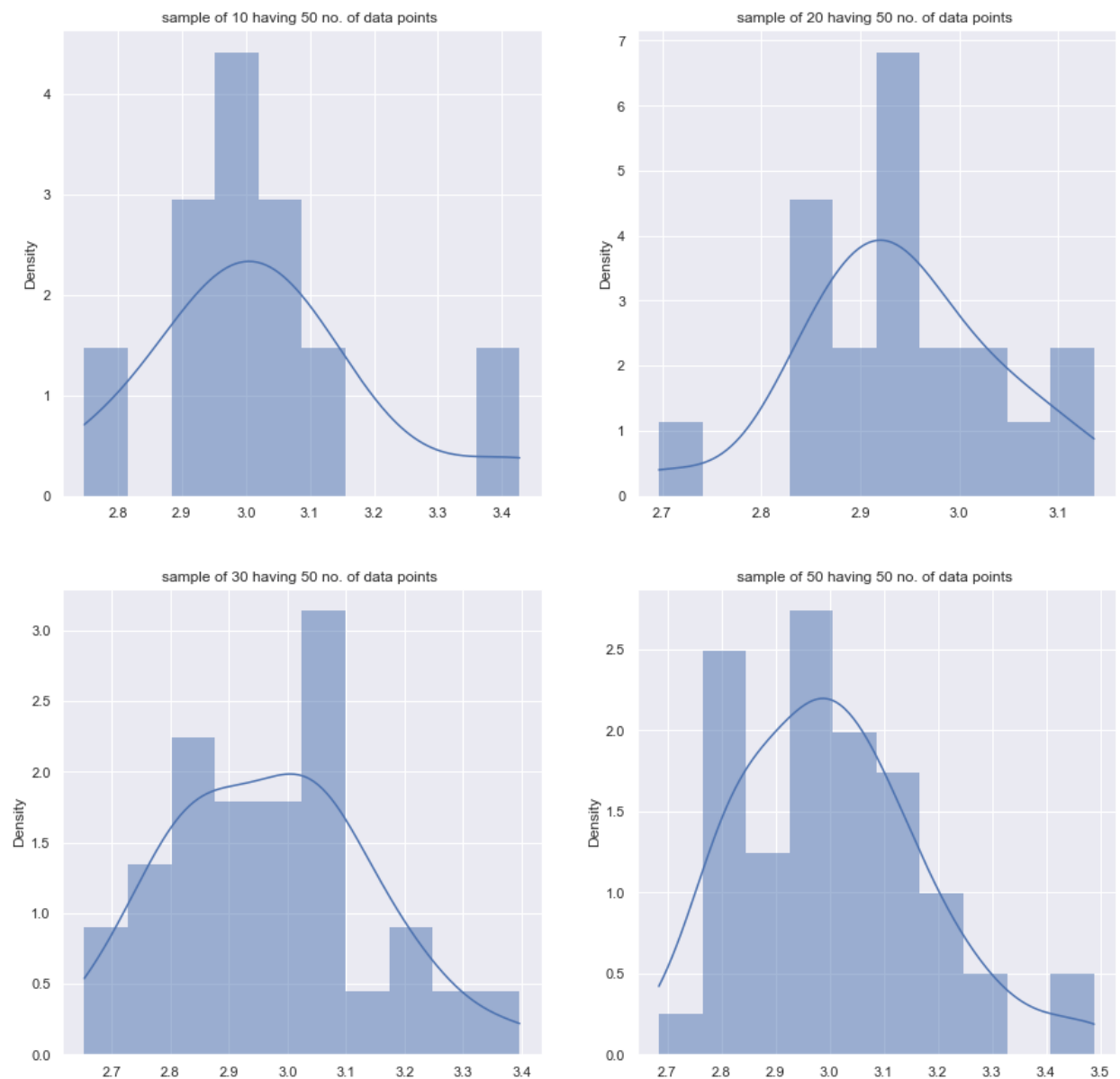
Out[64]: <AxesSubplot:xlabel='tip', ylabel='Count'>

```



```
In [65]: def mean_distribution(data, samples_count, data_points_count):
    list_sample = list()
    data = np.array(data.values)
    for i in range(0, samples_count):
        samples = random.sample(range(0, data.shape[0]), data_points_count)
        list_sample.append(data[samples].mean())
    return np.array(list_sample)

count = 0
mean_list = list()
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(15, 15))
lst = [(10,50),(20,50),(30,50),(50,50)]
for i in (0,1):
    for j in (0,1):
        ax[i,j].set_title("sample of " + str(lst[count][0]) + " having " + str(lst[count][1])+" no. of data p
        sns.histplot(mean_distribution(df_numeric_data['tip'], lst[count][0],lst[count][1]),ax = ax[i,j],bins
        mean_list.append(mean_distribution(df_numeric_data['tip'], lst[count][0],lst[count][1]))
        count +=1
```



8.3 for size

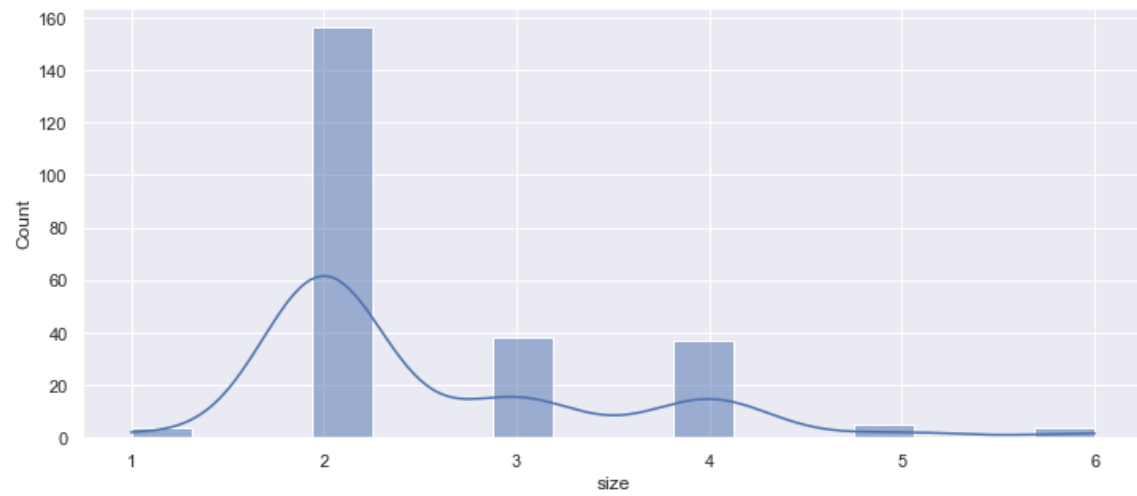
```
In [66]: mean_pop_size = df_numeric_data['size'].mean()
std_pop_size = df_numeric_data['size'].std()

print("population mean ( $\mu$ ): {}\npopulation standard deviation ( $\sigma$ ): {}".format(mean_pop_size, std_pop_size))

population mean ( $\mu$ ): 2.569672131147541
population standard deviation ( $\sigma$ ): 0.9510998047322332
```

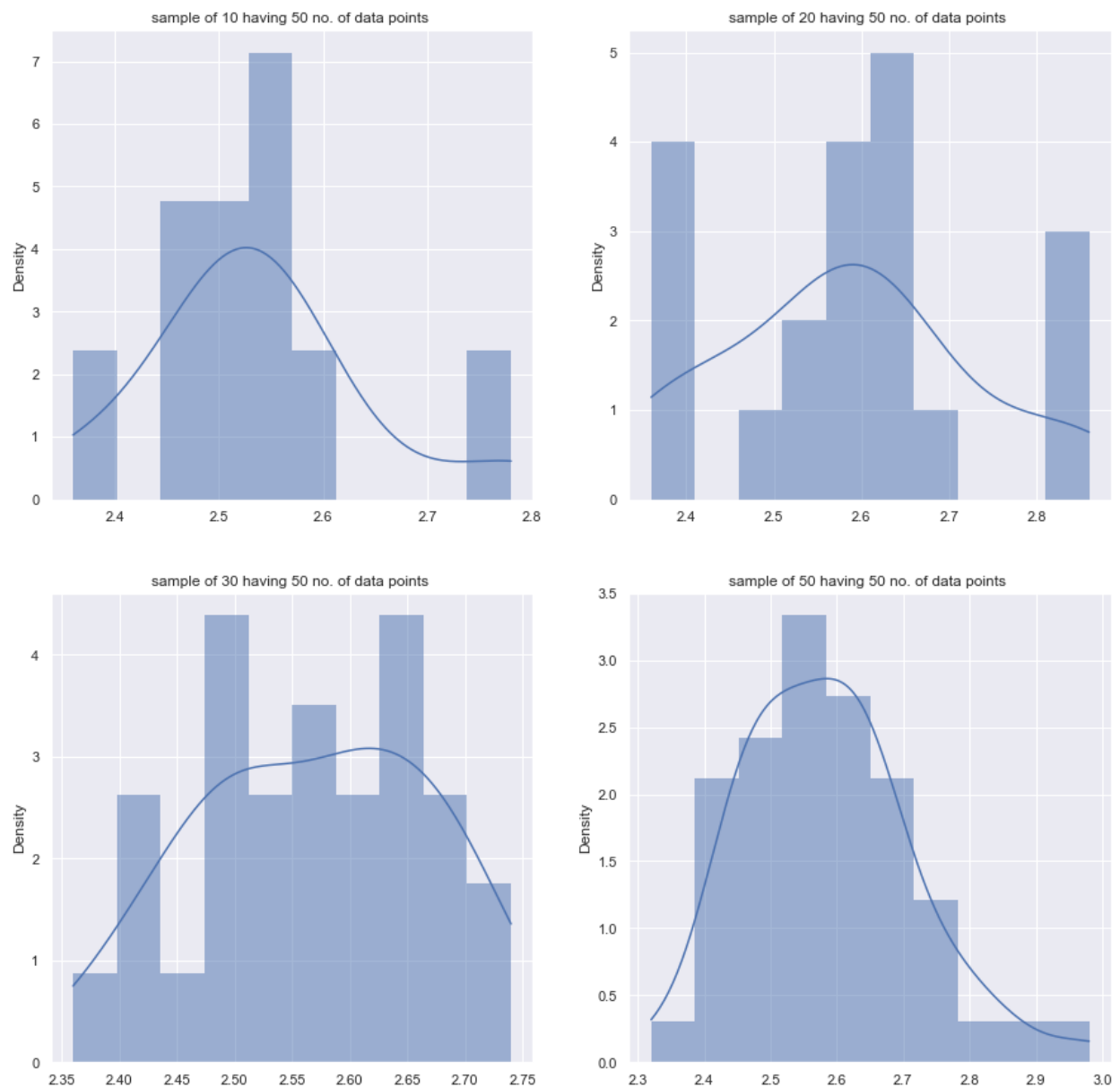
```
In [67]: sns.set(rc={'figure.figsize':(12,5)})
sns.histplot(df_numeric_data['size'], kde=True)
```

```
Out[67]: <AxesSubplot:xlabel='size', ylabel='Count'>
```



```
In [68]: def mean_distribution(data, samples_count, data_points_count):
    list_sample = list()
    data = np.array(data.values)
    for i in range(0, samples_count):
        samples = random.sample(range(0, data.shape[0]), data_points_count)
        list_sample.append(data[samples].mean())
    return np.array(list_sample)

count = 0
mean_list = list()
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(15, 15))
lst = [(10,50),(20,50),(30,50),(50,50)]
for i in (0,1):
    for j in (0,1):
        ax[i,j].set_title("sample of " + str(lst[count][0]) + " having " + str(lst[count][1])+" no. of data p
        sns.histplot(mean_distribution(df_numeric_data['size'], lst[count][0],lst[count][1]),ax = ax[i,j],bin
        mean_list.append(mean_distribution(df_numeric_data['size'], lst[count][0],lst[count][1]))
        count +=1
```



9.0 Covariance, Pearson correlation coefficient, Spearman's rank correlation coefficient

9.1 Covariance

In [69]: `df_numeric_data.cov()`

Out[69]:

	total_bill	tip	size
total_bill	79.252939	8.323502	5.065983
tip	8.323502	1.914455	0.643906
size	5.065983	0.643906	0.904591

9.2 Pearson correlation coefficient

In [70]: `df_numeric_data.corr("pearson")`

Out[70]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

9.3 Spearman's rank correlation coefficient

```
In [75]: df_numeric_data.corr("spearman")
```

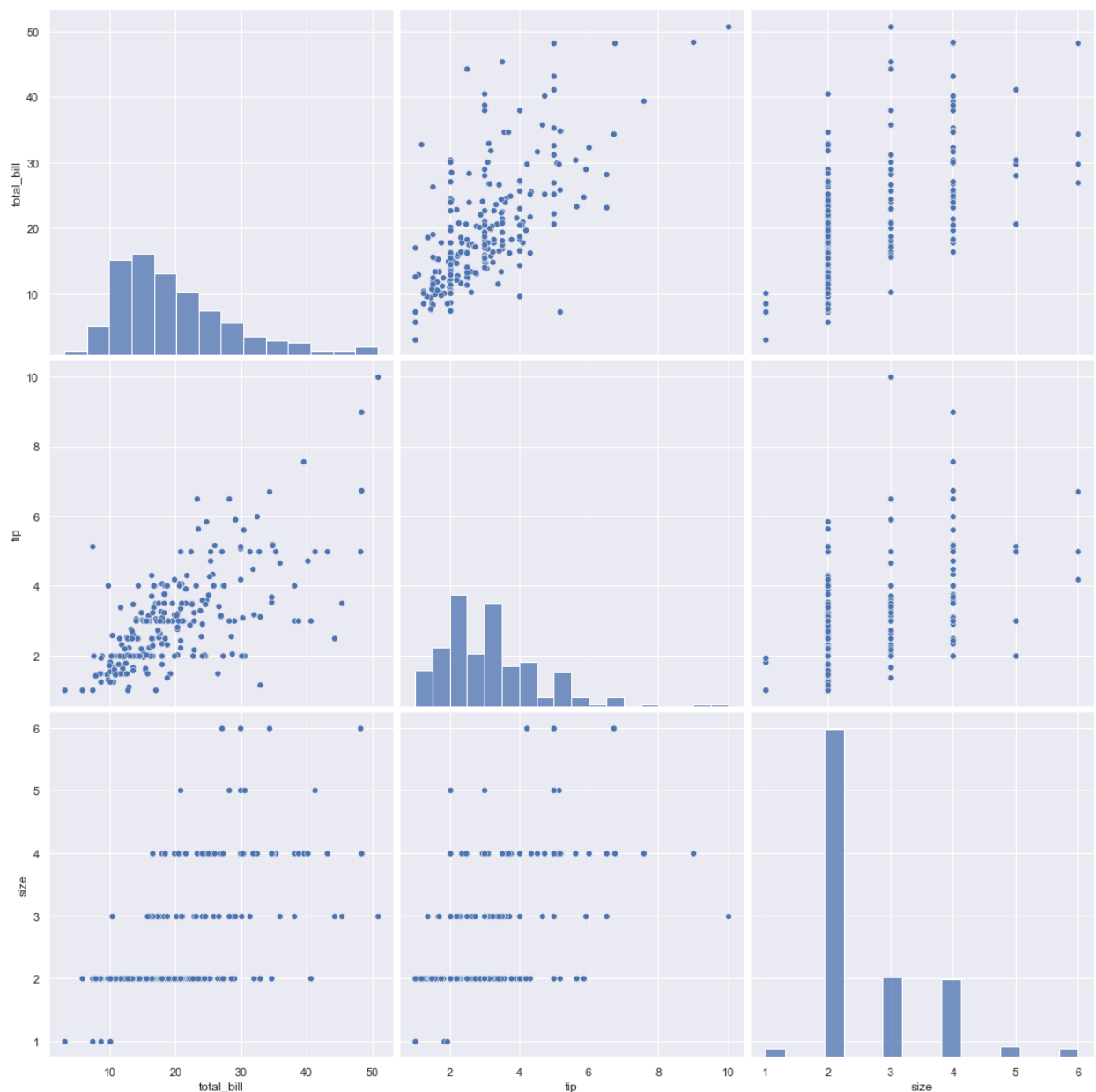
```
Out[75]:
```

	total_bill	tip	size
total_bill	1.000000	0.678968	0.604791
tip	0.678968	1.000000	0.468268
size	0.604791	0.468268	1.000000

9.4 Pairplot for tips dataset

```
In [87]: sns.set(rc={'figure.figsize':(15,15)})
sns.pairplot(df_numeric_data, height=5)
```

```
Out[87]: <seaborn.axisgrid.PairGrid at 0x20b5812e550>
```



9.4 Pairplot for tips dataset with kde

```
In [83]: sns.set(rc={'figure.figsize':(15,15)})
sns.pairplot(df_numeric_data, diag_kind="kde", height=5)
```

```
Out[83]: <seaborn.axisgrid.PairGrid at 0x20b542afee0>
```