```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv(r"C:\Users\Arigala.Adarsh\Downloads\mtcars.csv")
df
```

```
                  model   mpg  cyl   disp   hp  drat     wt   qsec  vs  
am  \
0             Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   
1
1         Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   
1
2            Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   
1
3        Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   
0
4     Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   
0
5               Valiant  18.1    6  225.0  105  2.76  3.460  20.22   1   
0
6            Duster 360  14.3    8  360.0  245  3.21  3.570  15.84   0   
0
7             Merc 240D  24.4    4  146.7   62  3.69  3.190  20.00   1   
0
8              Merc 230  22.8    4  140.8   95  3.92  3.150  22.90   1   
0
9              Merc 280  19.2    6  167.6  123  3.92  3.440  18.30   1   
0
10            Merc 280C  17.8    6  167.6  123  3.92  3.440  18.90   1   
0
11           Merc 450SE  16.4    8  275.8  180  3.07  4.070  17.40   0   
0
12           Merc 450SL  17.3    8  275.8  180  3.07  3.730  17.60   0   
0
13          Merc 450SLC  15.2    8  275.8  180  3.07  3.780  18.00   0   
0
14   Cadillac Fleetwood  10.4    8  472.0  205  2.93  5.250  17.98   0   
0
15  Lincoln Continental  10.4    8  460.0  215  3.00  5.424  17.82   0   
0
16    Chrysler Imperial  14.7    8  440.0  230  3.23  5.345  17.42   0   
0
17             Fiat 128  32.4    4   78.7   66  4.08  2.200  19.47   1   
1
18          Honda Civic  30.4    4   75.7   52  4.93  1.615  18.52   1   
1
19       Toyota Corolla  33.9    4   71.1   65  4.22  1.835  19.90   1   
1
```

```
20       Toyota Corona  21.5    4  120.1   97  3.70  2.465  20.01   1
0
21    Dodge Challenger  15.5    8  318.0  150  2.76  3.520  16.87   0
0
22         AMC Javelin  15.2    8  304.0  150  3.15  3.435  17.30   0
0
23          Camaro Z28  13.3    8  350.0  245  3.73  3.840  15.41   0
0
24    Pontiac Firebird  19.2    8  400.0  175  3.08  3.845  17.05   0
0
25           Fiat X1-9  27.3    4   79.0   66  4.08  1.935  18.90   1
1
26       Porsche 914-2  26.0    4  120.3   91  4.43  2.140  16.70   0
1
27        Lotus Europa  30.4    4   95.1  113  3.77  1.513  16.90   1
1
28      Ford Pantera L  15.8    8  351.0  264  4.22  3.170  14.50   0
1
29        Ferrari Dino  19.7    6  145.0  175  3.62  2.770  15.50   0
1
30       Maserati Bora  15.0    8  301.0  335  3.54  3.570  14.60   0
1
31          Volvo 142E  21.4    4  121.0  109  4.11  2.780  18.60   1
1

    gear  carb
0      4     4
1      4     4
2      4     1
3      3     1
4      3     2
5      3     1
6      3     4
7      4     2
8      4     2
9      4     4
10     4     4
11     3     3
12     3     3
13     3     3
14     3     4
15     3     4
16     3     4
17     4     1
18     4     2
19     4     1
20     3     1
21     3     2
22     3     2
```

```
23     3     4
24     3     2
25     4     1
26     5     2
27     5     2
28     5     4
29     5     6
30     5     8
31     4     2
```

```
df.shape
```

```
(32, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   model   32 non-null     object
 1   mpg     32 non-null     float64
 2   cyl     32 non-null     int64
 3   disp    32 non-null     float64
 4   hp      32 non-null     int64
 5   drat    32 non-null     float64
 6   wt      32 non-null     float64
 7   qsec    32 non-null     float64
 8   vs      32 non-null     int64
 9   am      32 non-null     int64
 10  gear    32 non-null     int64
 11  carb    32 non-null     int64
dtypes: float64(5), int64(6), object(1)
memory usage: 3.1+ KB
```

```
df.describe()
```

```
             mpg        cyl        disp          hp       drat
wt  \
count  32.000000  32.000000   32.000000   32.000000  32.000000
32.000000
mean   20.090625   6.187500  230.721875  146.687500   3.596563
3.217250
std     6.026948   1.785922  123.938694   68.562868   0.534679
0.978457
min    10.400000   4.000000   71.100000   52.000000   2.760000
1.513000
25%    15.425000   4.000000  120.825000   96.500000   3.080000
2.581250
50%    19.200000   6.000000  196.300000  123.000000   3.695000
```

```
3.325000
75%    22.800000   8.000000  326.000000  180.000000   3.920000
3.610000
max    33.900000   8.000000  472.000000  335.000000   4.930000
5.424000

            qsec         vs         am       gear     carb
count  32.000000  32.000000  32.000000  32.000000  32.0000
mean   17.848750   0.437500   0.406250   3.687500   2.8125
std     1.786943   0.504016   0.498991   0.737804   1.6152
min    14.500000   0.000000   0.000000   3.000000   1.0000
25%    16.892500   0.000000   0.000000   3.000000   2.0000
50%    17.710000   0.000000   0.000000   4.000000   2.0000
75%    18.900000   1.000000   1.000000   4.000000   4.0000
max    22.900000   1.000000   1.000000   5.000000   8.0000
```

```
df.isnull().sum()
```

```
model    0
mpg      0
cyl      0
disp     0
hp       0
drat     0
wt       0
qsec     0
vs       0
am       0
gear     0
carb     0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

```
df.mean()
```

```
C:\Users\Arigala.Adarsh\AppData\Local\Temp\
ipykernel_24668\3698961737.py:1: FutureWarning: Dropping of nuisance
columns in DataFrame reductions (with 'numeric_only=None') is
deprecated; in a future version this will raise TypeError.  Select
only valid columns before calling the reduction.
  df.mean()
```

```
mpg      20.090625
cyl       6.187500
disp    230.721875
hp      146.687500
drat      3.596563
wt        3.217250
```

```
qsec     17.848750
vs        0.437500
am        0.406250
gear      3.687500
carb      2.812500
dtype: float64
```

Mean

```
df.mpg.mean()

20.090624999999996
```

Median

```
df.mpg.median()

19.2
```

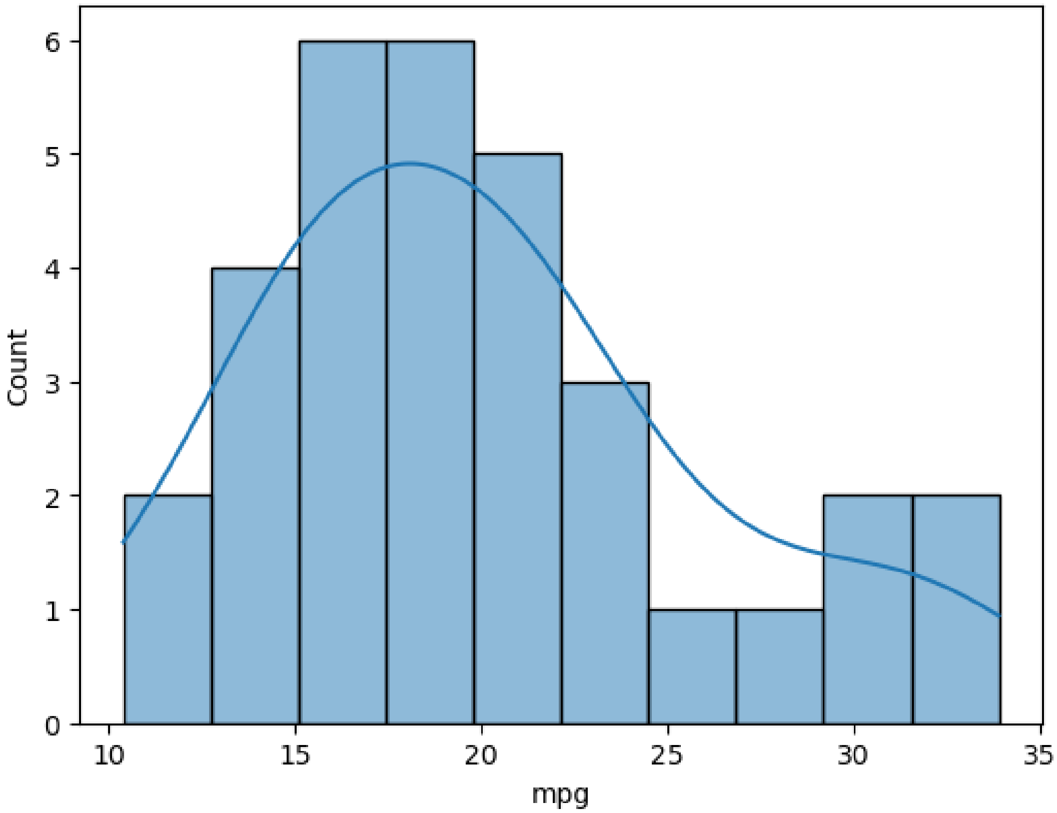
Mode

```
df.mpg.mode()

0    10.4
1    15.2
2    19.2
3    21.0
4    21.4
5    22.8
6    30.4
Name: mpg, dtype: float64

df[df.mpg==10.4]

                  model   mpg  cyl   disp   hp  drat     wt   qsec  vs
am  \
14   Cadillac Fleetwood  10.4    8  472.0  205  2.93  5.250  17.98   0
0
15  Lincoln Continental  10.4    8  460.0  215  3.00  5.424  17.82   0
0

    gear  carb
14     3     4
15     3     4

df[df.mpg==30.4]

           model   mpg  cyl  disp   hp  drat     wt   qsec  vs  am
gear  carb
18   Honda Civic  30.4    4  75.7   52  4.93  1.615  18.52   1   1
4     2
```

```
27  Lotus Europa  30.4    4  95.1  113  3.77  1.513  16.90   1   1
5     2
```

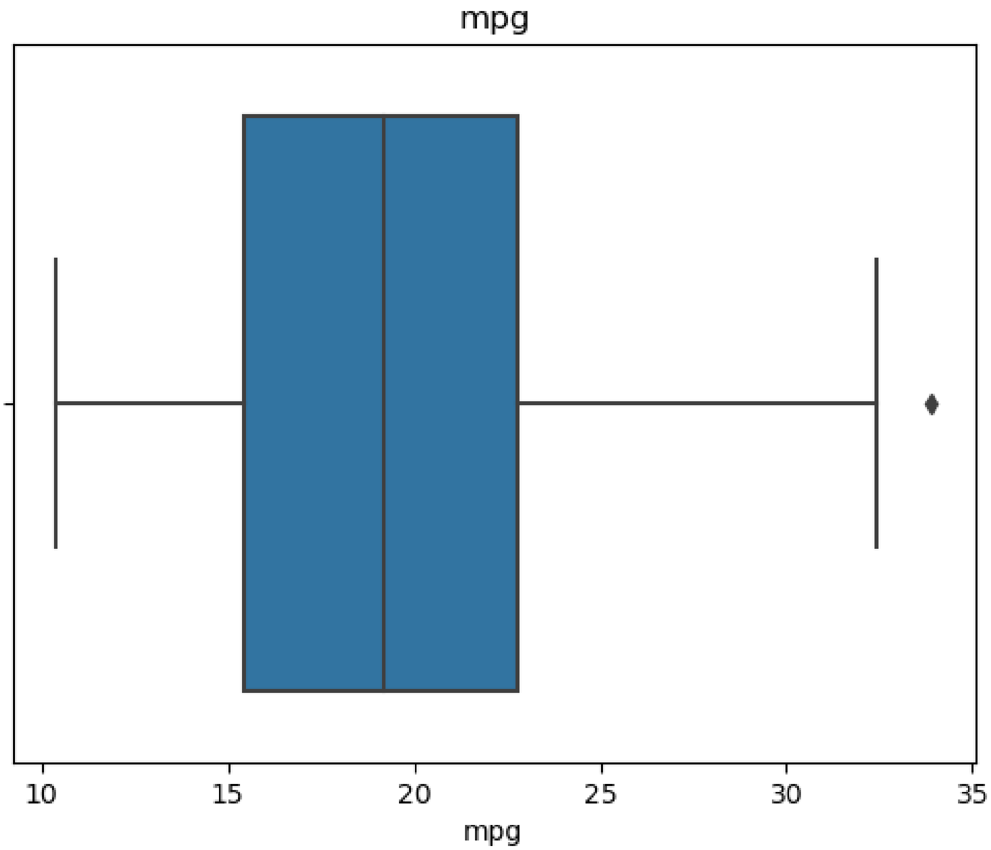
Range

```
max=df.mpg.max()
min=df.mpg.min()

range=max-min
print("Range of mpg:",range)

Range of mpg: 23.5

sns.histplot(df.mpg,bins=10,kde=True)
plt.show()
```

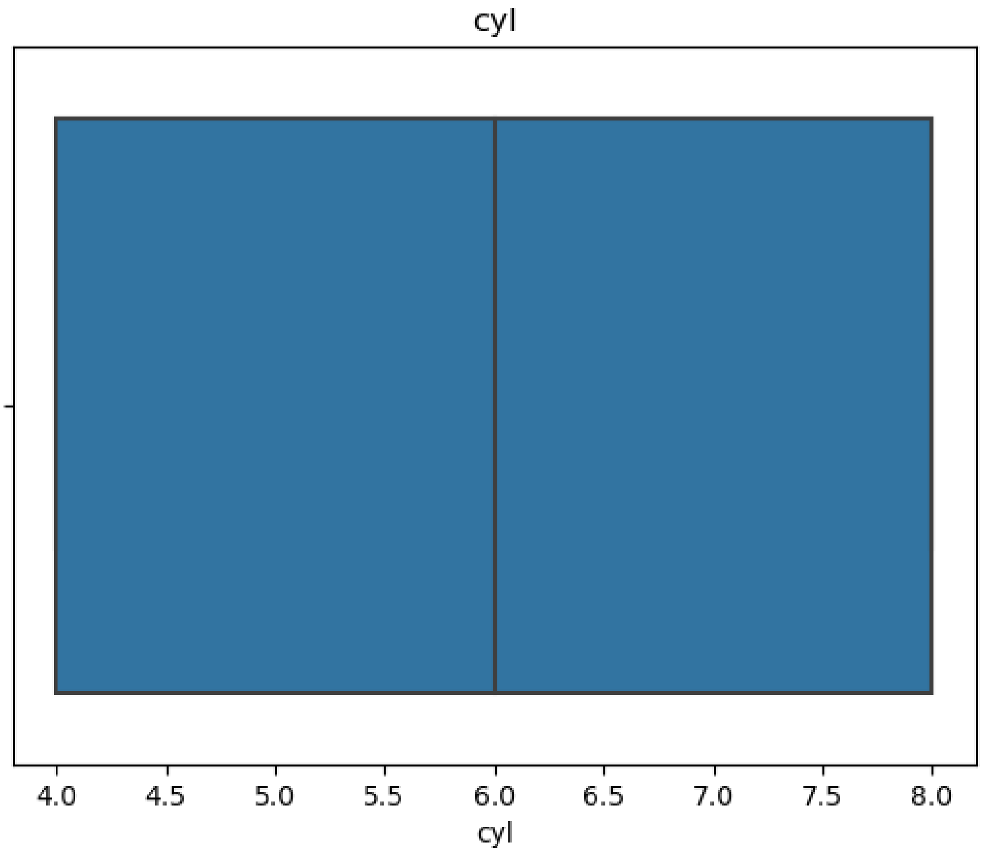


```
for i in df.columns:
    if(df[i].dtypes!='object'):
        sns.boxplot(df[i])
        plt.title(i)
        plt.show()

C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
```

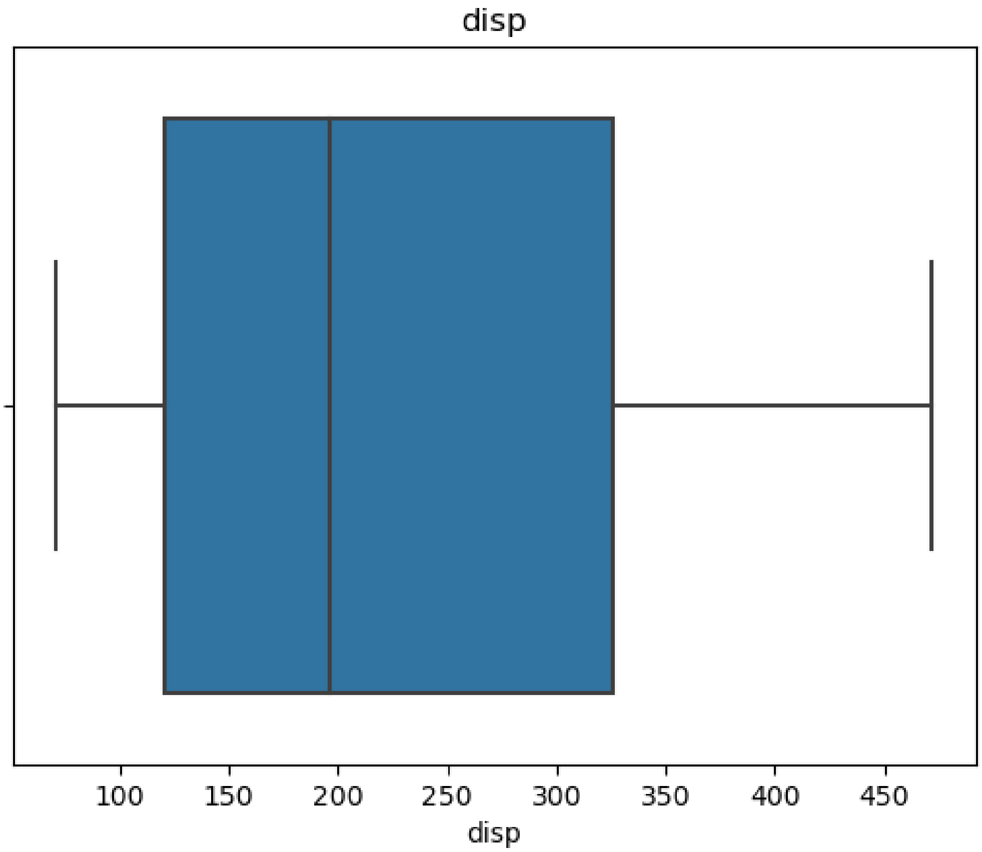
```
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



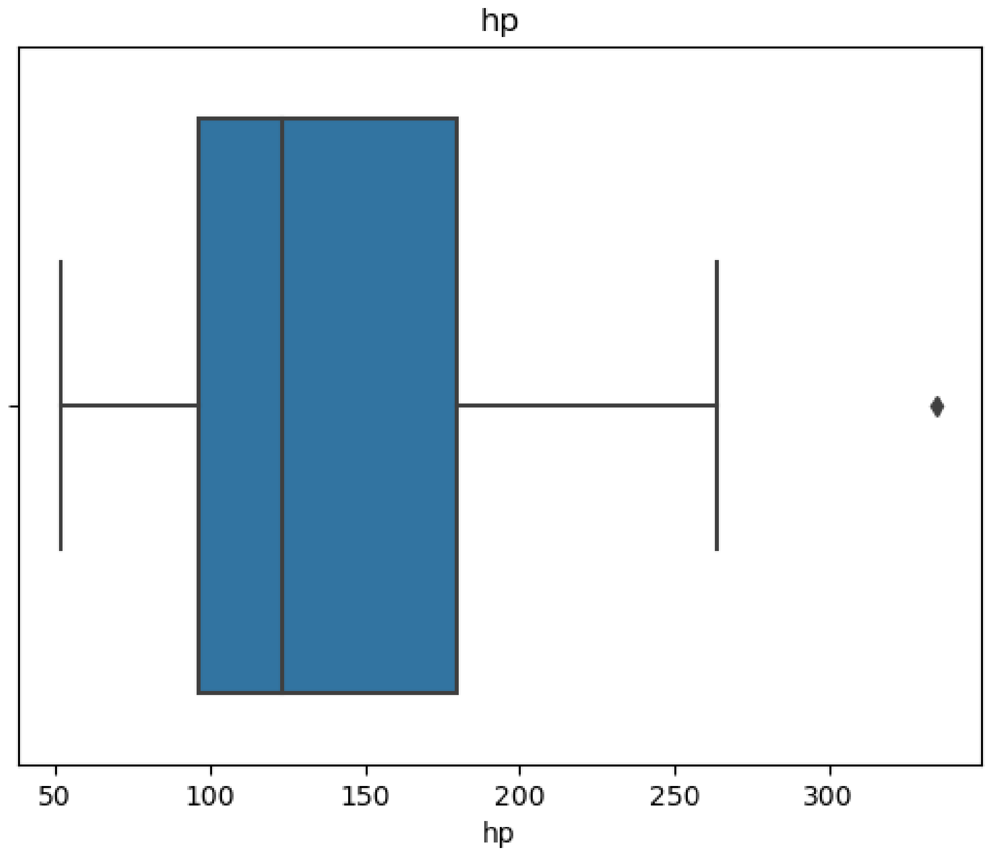
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



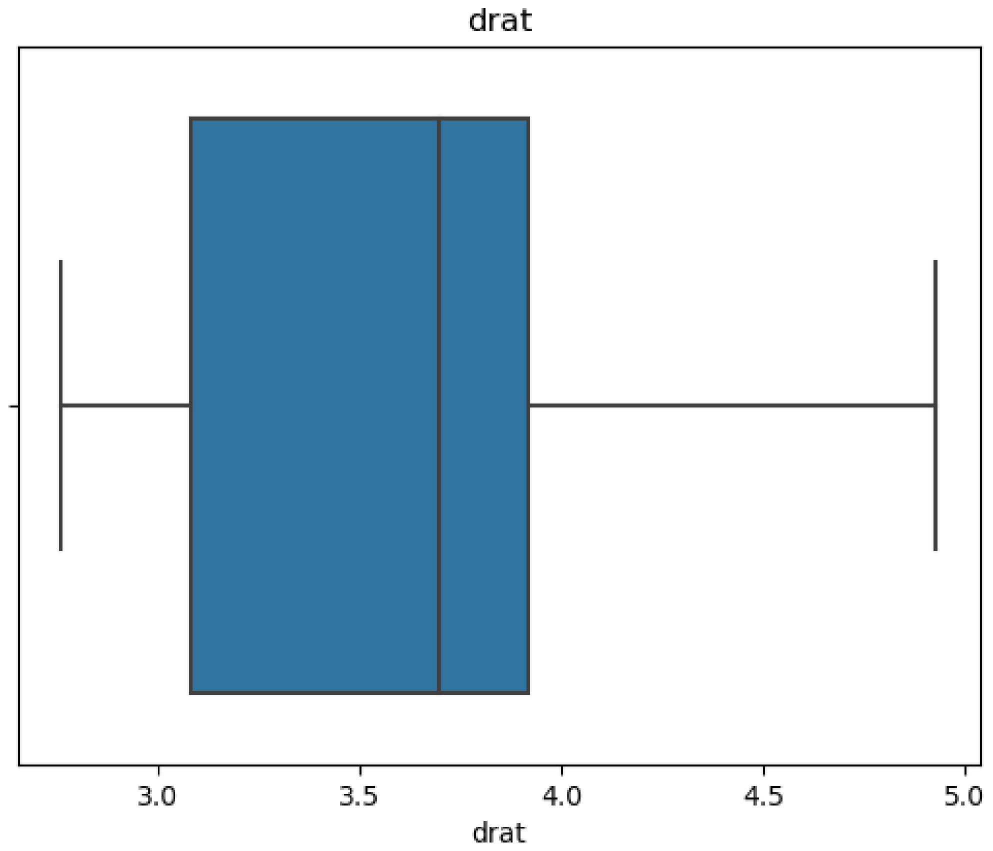
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



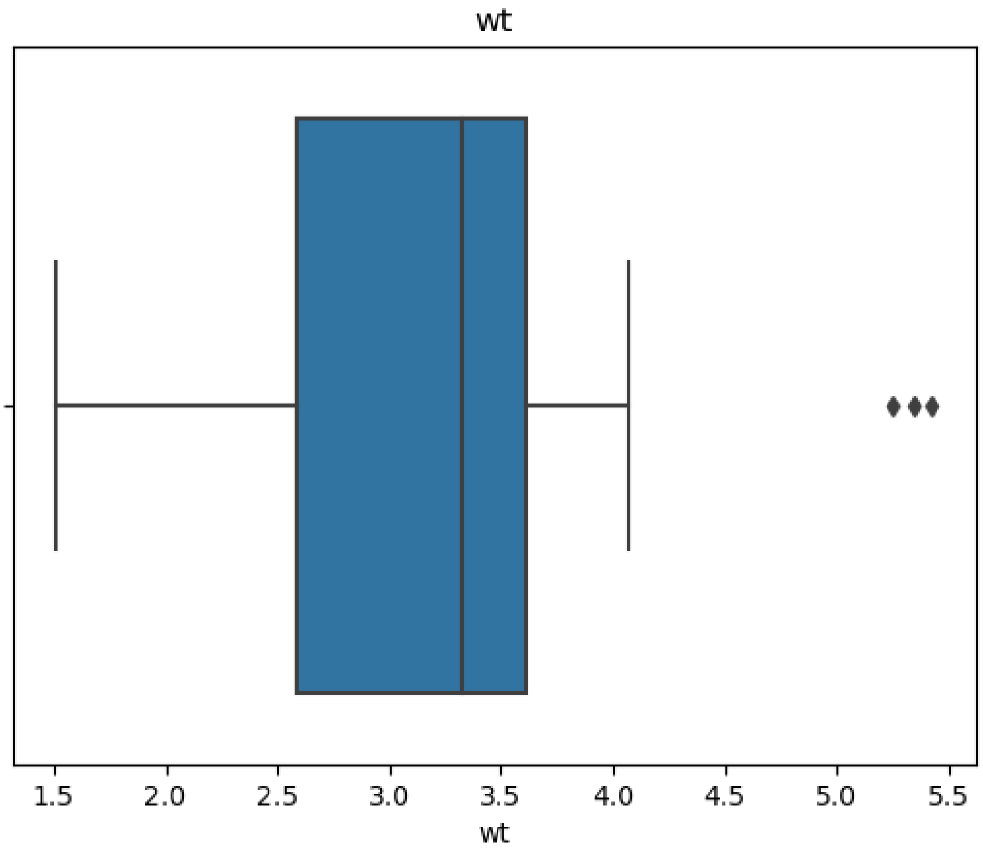
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



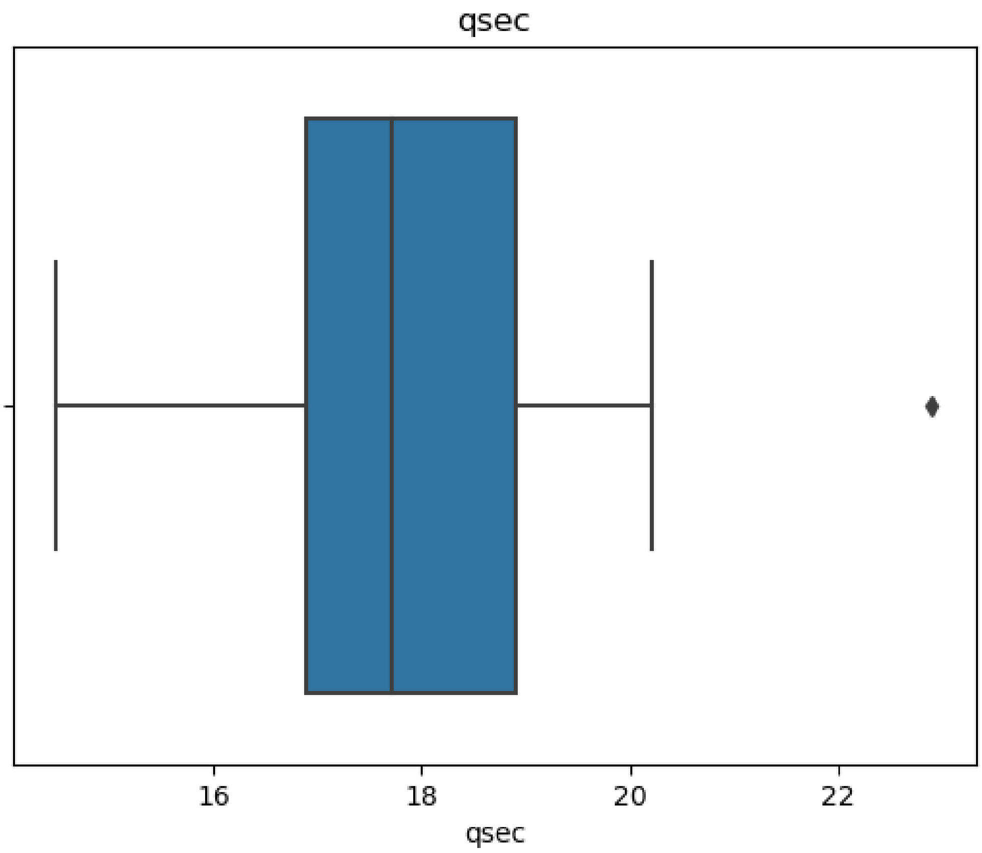
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



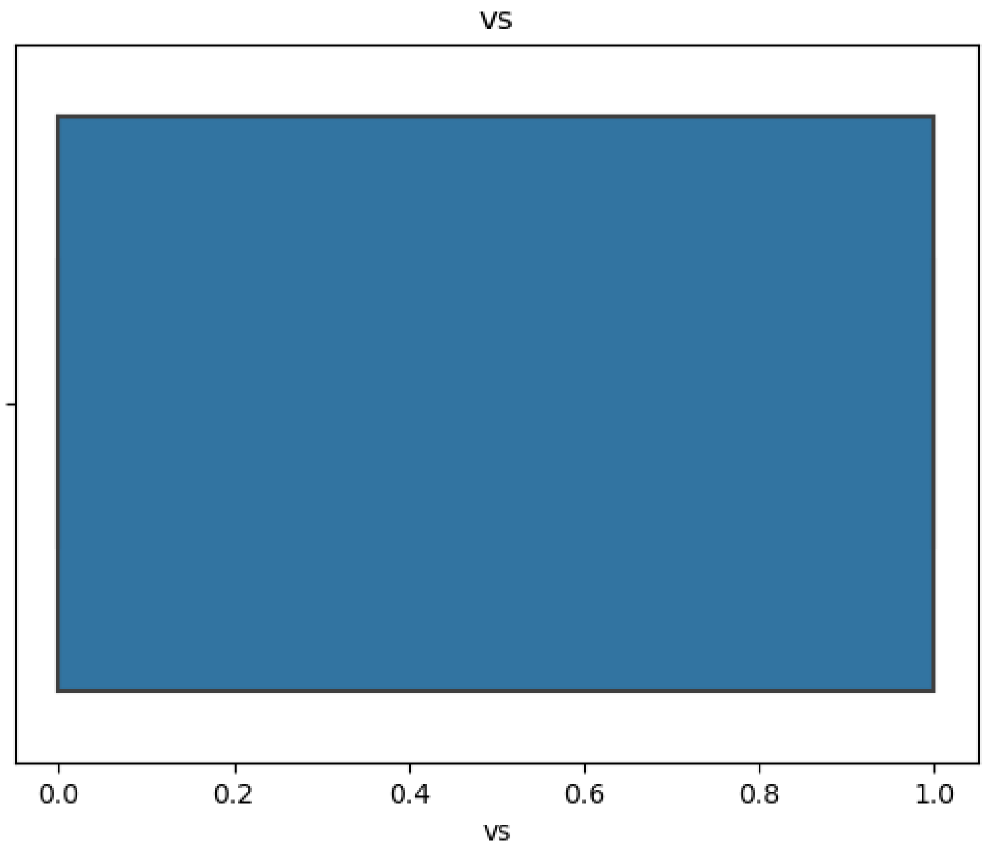
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



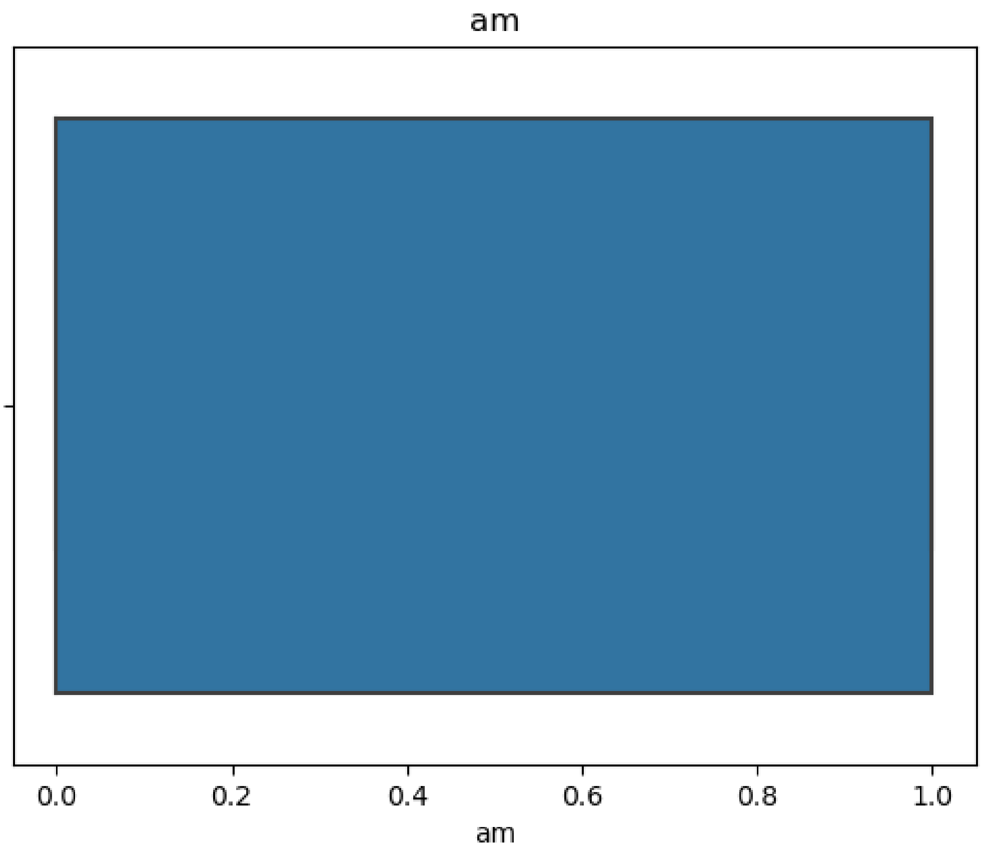
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



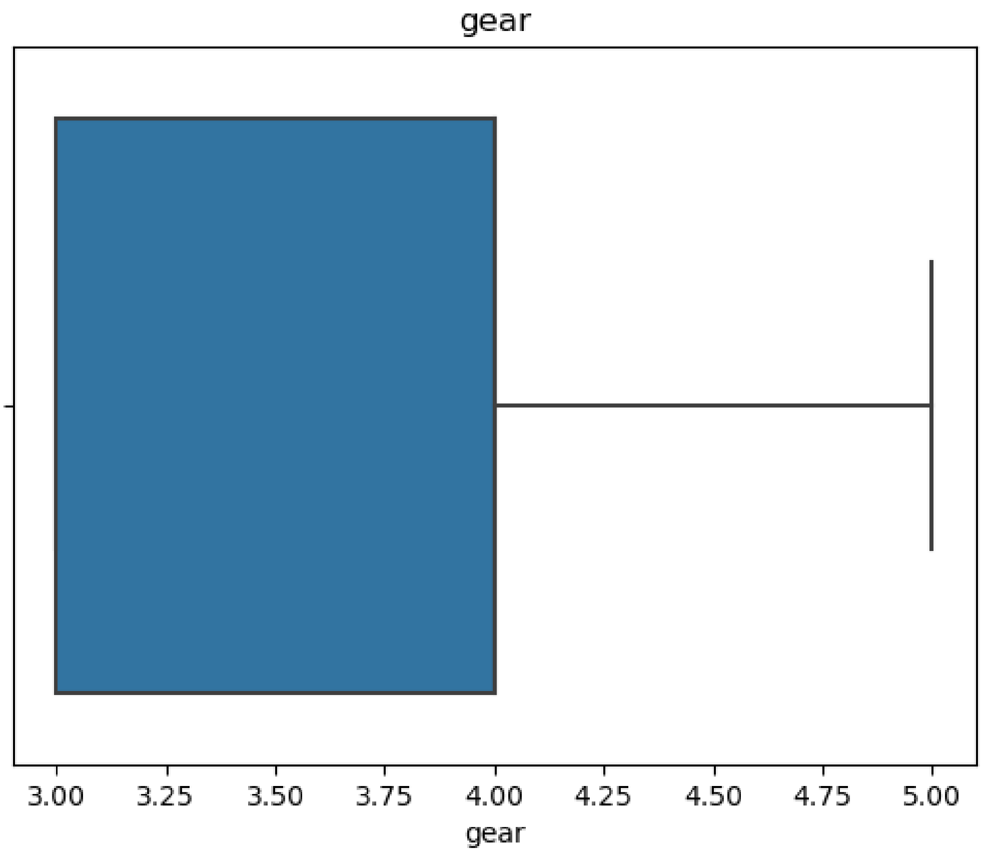
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



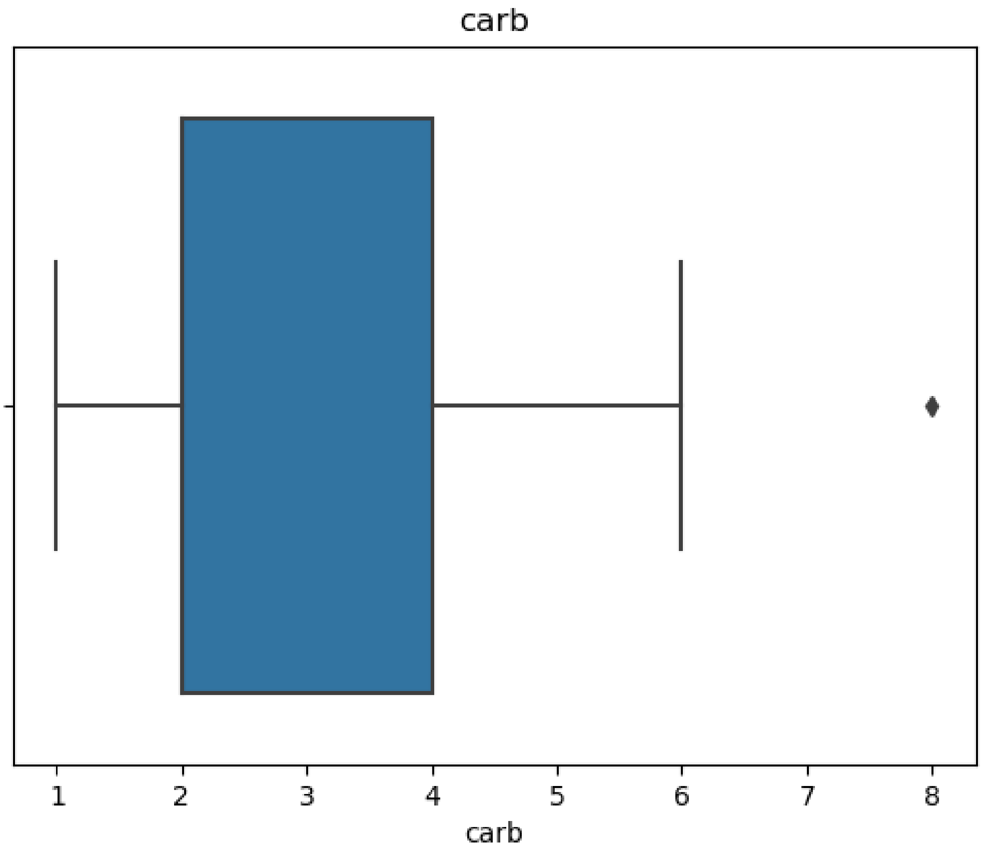
```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\seaborn\
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



# Variance and Standard Deviation

```
print("Variance of mpg is :",df.mpg.var())
print("Standard deviation of mpg is :",df.mpg.std())
```

```
Variance of mpg is : 36.32410282258065
Standard deviation of mpg is : 6.026948052089105
```

# Inferential Statistics

```
from scipy import stats
```

#pip install bioinfokit

```
from bioinfokit.analys import get_data
```

# One Sample Z-test

One Sample Z-test checks whether the sample comes from a `known population` where `population mean and standard deviation (σ)` should be known.

```
df1=get_data('z_one_samp').data
df1
```

```
        sizes
0   4.819289
1   3.569358
2   5.346402
3   5.950908
4   5.871183
5   5.590780
6   3.591993
7   5.137837
8   3.870271
9   4.624155
10  5.796371
11  6.194647
12  4.167342
13  5.168888
14  6.646000
15  4.327118
16  4.609097
17  6.828167
18  3.407583
19  4.218922
20  4.413460
21  4.034847
22  5.979468
23  4.470525
24  4.146668
25  4.711762
26  4.918706
27  5.297779
28  5.473580
29  5.813743
30  4.189742
31  4.522986
32  6.333137
33  5.961645
34  5.606989
35  4.559243
36  4.135710
37  6.597137
```

```
38   4.261241
39   2.776458
40   6.507718
41   4.905230
42   4.662474
43   4.829080
44   6.308754
45   5.198246
46   5.222804
47   6.362502
48   2.906932
49   6.053046
```

- Factory produces balls of a diameter of 5 cm
- Std. deviation = **0.4**
- Quality officer wants to test whether the ball diameter is significantly different from 5 cm or not
- Null Hypothesis - Sample Mean = 5 cm |||| Alternate hypothesis - Sample Mean not equal to 5

```
from bioinfokit.analys import stat
res=stat()
res.ztest(df=df1,x="sizes",mu=5,x_std=0.4,test_type=1)

print(res.summary)


One Sample Z-test

------------------  ---------
Sample size         50
Mean                 5.01796
Z value              0.317465
p value (one-tail)   0.375446
p value (two-tail)   0.750891
Lower 95.0%          4.90709
Upper 95.0%          5.12883
------------------  ---------
```

- we can conclude that z_test is not significant. The mean diameter of the balls in random sample is equal to the population mean is 5cm.Accept the null hypothesis.

# Two Sample Z-Test

Null hypothesis: Two group means are equal Alternate hypothesis: Two group means are different (not equal)

```
df2 = get_data('z_two_samp').data
df2.head()
```

```
     fact_A    fact_B
0  4.977904  5.887947
1  5.166254  5.990616
2  4.991749  6.110116
3  4.901557  5.936784
4  4.713866  6.227506
```

Std. deviation of two population is fixed at 0.1

```
res = stat()
res.ztest(df=df2, x='fact_A', y='fact_B', x_std=0.1, y_std=0.1,
test_type=2)

print(res.summary)
```

```
Two Sample Z-test

------------------  ----------
Sample size for x    50
Sample size for y    50
Mean of x             5.01284
Mean of y             5.99015
Z value             -48.8656
p value (one-tail)    0
p value (two-tail)    0
Lower 95.0%          -1.01651
Upper 95.0%          -0.938113
------------------  ----------
```

- Reject the null hypothesis and can conclude that there is a significant difference in the ball size produced in factories A and B
- Factory A is closer to mean 5 however Factory B is not

## T Test

### One Sample Test

- used for comparing the sample mean (a random sample from a population) with the specific value
- In t-test, the population variance is unknown and it is estimated from sample variance
- If n < 30, Z-test cant be applied
- Null Hypothesis: Sample mean is equal to the hypothesized or known population mean
- Alternate Hypothesis: Sample mean is not equal to the hypothesized or known population mean

```
t1 = get_data('t_one_samp').data
t1.head(2)
```

```
       size
0  5.739987
1  5.254042
```

```
res=stat()
res.ttest(df=t1, test_type=1, res='size', mu=5)

print(res.summary)
```

```

One Sample t-test

------------------  --------
Sample size         50
Mean                 5.05128
t                    0.36789
Df                  49
p value (one-tail)   0.35727
p value (two-tail)   0.71454
Lower 95.0%          4.77116
Upper 95.0%          5.3314
------------------  --------
```

## Two Sample T-Test

t-test compares the means of two independent groups, determining whether they are equal or significantly different

```
### load dataset as pandas dataframe

t2 = get_data('t_ind_samp').data
t2
```

```
   Genotype  yield
0         A   78.0
1         A   84.3
2         A   81.0
3         B   88.0
4         B   92.0
5         B   84.1
6         A   74.5
7         A   77.8
8         A   79.0
9         B   88.0
10        B   92.5
11        B   91.8
```

```
res = stat()
res.ttest(df=t2, xfac='Genotype', res='yield', test_type=2)

print(res.summary)


Two sample t-test with equal variance

------------------  -------------
Mean diff           -10.3
t                    -5.40709
Std Error             1.90491
df                   10
p value (one-tail)    0.000149204
p value (two-tail)    0.000298408
Lower 95.0%         -14.5444
Upper 95.0%          -6.05561
------------------  -------------

Parameter estimates

Level      Number     Mean     Std Dev     Std Error     Lower 95.0%
Upper 95.0%
-------  --------  ------  ---------  -----------  -------------
-------------
A                6     79.1     3.30817       1.35056        75.6283
82.5717
B                6     89.4     3.29059       1.34338        85.9467
92.8533
```

- statistically significant and hence reject the null hypothesis

# Chi-Square Test

- Test for independence
- Null Hypothesis: there is no association between the two categorical variables
- Alternate Hypothesis: there is an association between the two categorical variables

```
chi = get_data('drugdata').data
chi

   treatments  cured  noncured
0     treated     60        10
1  nontreated     30        25

### Set treatment column as your index
chi= chi.set_index('treatments')

chi
```

```
            cured  noncured
treatments
treated        60        10
nontreated     30        25

res = stat()
res.chisq(df=chi)

print(res.summary)


Chi-squared test for independence

Test              Df    Chi-square      P-value
--------------  ----  ------------  -----------
Pearson            1       13.3365  0.000260291
Log-likelihood     1       13.4687  0.000242574
```