In [1]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
from scipy.stats import ttest_ind
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
import random
vol=np.random.randn(400)
#randn will generate values in standard normal scale
vol[:50]
```

Out[3]:
```
array([-3.26328375e-01, -3.75551486e-01, -1.22148074e+00,  4.75539527e-01,
        2.07935050e+00, -1.93233821e+00,  3.03882745e-01,  4.65615859e-01,
       -2.92467949e-01, -2.77921169e+00, -3.71255142e-02,  4.31287181e-01,
        2.54679137e+00,  2.47497081e-01, -6.81927992e-01,  2.20881442e+00,
       -1.09452715e+00, -3.14261720e-01, -1.09431824e+00,  1.46915824e+00,
       -5.21034048e-01, -1.16939168e-01, -1.25526832e+00, -1.87880152e+00,
       -3.29281416e-01, -2.34937478e-01, -3.47567045e-01,  6.20307171e-01,
        2.01675941e-01,  1.45150895e-03,  5.43461913e-01, -1.20237380e-01,
        1.77981442e+00, -7.93108750e-01,  7.81308358e-01,  1.31609768e+00,
        9.05585817e-01,  7.83014183e-01,  3.05764448e-01, -2.63191548e+00,
       -2.29322771e+00,  5.15631126e-01, -2.24448697e-01,  7.62143653e-01,
        2.60881292e-01,  1.95498272e+00,  9.36165787e-01, -1.96664580e-01,
        1.82994294e+00,  5.20215195e-01])
```

In [4]:
```
samp_vol=vol*1.5+298.56
samp_vol[:50]
```

Out[4]:
```
array([298.07050744, 297.99667277, 296.7277789 , 299.27330929,
       301.67902575, 295.66149269, 299.01582412, 299.25842379,
       298.12129808, 294.39118247, 298.50431173, 299.20693077,
       302.38018705, 298.93124562, 297.53710801, 301.87322164,
       296.91820928, 298.08860742, 296.91852263, 300.76373735,
       297.77844893, 298.38459125, 296.67709752, 295.74179772,
       298.06607788, 298.20759378, 298.03864943, 299.49046076,
       298.86251391, 298.56217726, 299.37519287, 298.37964393,
       301.22972163, 297.37033688, 299.73196254, 300.53414652,
       299.91837873, 299.73452127, 299.01864667, 294.61212678,
       295.12015844, 299.33344669, 298.22332695, 299.70321548,
       298.95132194, 301.49247407, 299.96424868, 298.26500313,
       301.30491442, 299.34032279])
```

In [5]:
```
#To verify fill volume of softdrink follows 300ml spec
#Random 400 samples were collected from the factory
#The statistics is found to be
x_bar=np.mean(samp_vol)
s=np.std(samp_vol,ddof=1)
x_bar,s
```

Out[5]:
```
(298.51331860969555, 1.549862841816424)
```

One sample t-Test (two tailed)

- Ho: pop_mean=300 ml
- Ha: pop_mean != 300 ml

In [6]:
```
t_stat=(x_bar-300)/(s/np.sqrt(400))
t_stat
```

Out[6]:
```
-19.184683317680918
```

In [ ]:
```
# we can verify the score with the built in function
```

In [7]:
```
from scipy.stats import ttest_1samp
```

In [9]:
```
stats.norm.isf(0.05)
```

Out[9]:
```
1.6448536269514729
```

In [8]:
```
ttest_1samp(samp_vol,300)
```

Out[8]:
```
Ttest_1sampResult(statistic=-19.184683317680918, pvalue=1.3520986667089086e-58)
```

since p-val <0.05, we reject Ho,which implies Ha holds good, (ie) sample doesn't represent 300 ml

In [40]:
```
#Courier Company Example (One-tail test)
#n=50
dtime=np.random.randn(50)
samp_dtime=dtime*0.6+2.88
```

In [41]:
```
x_bar=np.mean(samp_dtime)
s=np.std(samp_dtime,ddof=1)
x_bar,s
```

Out[41]:
```
(2.9690997558267496, 0.6322643868481541)
```

In [48]:
```
#compute 95% CI range for x_bar
x_bar-1.96*(s/np.sqrt(50))
```

Out[48]:
```
2.7938450091330607
```

In [49]:
```
x_bar+1.96*(s/np.sqrt(50))
```

Out[49]:
```
3.1443545025204385
```

In [50]:
```
t_stat=(x_bar-3)/(s/np.sqrt(50))
t_stat
```

Out[50]: -0.3455796759983094

In [51]:
```
ttest_1samp(samp_dtime,3)
```

Out[51]: Ttest_1sampResult(statistic=-0.3455796759983094, pvalue=0.7311378981078815)

In [ ]:
```
s
```

Out[16]: 0.5627626509636148

In [ ]:
```
t_stat=(x_bar-3)/(s/np.sqrt(500))
t_stat
```

Out[31]: -7.5972214726259075

In [ ]:
```
stats.norm.isf(0.05)
```

Out[13]: 1.6448536269514729

In [ ]:
```
np.mean(samp_dtime),np.std(samp_dtime,ddof=1)
```

Out[30]: (2.791987829101012, 0.6122361391640108)

In [ ]:
```
ttest_1samp(samp_dtime,3)
```

Out[28]: Ttest_1sampResult(statistic=-7.597221472625907, pvalue=1.505809226022567e-13)

since,p-val<0.05, we reject Ho, which implies Ha holds good, ie (courier company claim is correct)

In [ ]:
```
x_bar-1.96*(s/np.sqrt(50))
```

Out[25]: 2.6222846294303976

In [ ]:
```
x_bar+1.96*(s/np.sqrt(50))
```
Out[26]:
```
2.9616910287716265
```

Soyabean yield example (one-tailed-Right tailed)

- Ha: pop_mean >520
- Ho: pop_mean <=520

In [52]:
```
yd=np.random.randn(400)
samp_yd=yd*124+573
```

In [55]:
```
x_bar=np.mean(samp_yd)
s=np.std(samp_yd,ddof=1)
x_bar,s
```
Out[55]:
```
(572.8865052001379, 123.7503003973732)
```

In [56]:
```
t_stat=(x_bar-520)/(s/np.sqrt(400))
t_stat
```
Out[56]:
```
8.547293223582429
```

In [57]:
```
ttest_1samp(samp_yd,520)
```
Out[57]:
```
Ttest_1sampResult(statistic=8.547293223582429, pvalue=2.7099868415520073e-16)
```

In [60]:
```
#HYD DSE Data
dse_age_hyd=np.random.randn(800)
DH_age=dse_age_hyd*1.8+24.3
np.mean(DH_age),np.std(DH_age,ddof=1)
```
Out[60]:
```
(24.331701058427495, 1.7577885499753965)
```

In [61]:
```
#BLR DSE Data
dse_age_blr=np.random.randn(1000)
DB_age=dse_age_blr*2.3+26.5
np.mean(DB_age),np.std(DB_age,ddof=1)
```

Out[61]:
```
(26.505777478090163, 2.321083438150285)
```

In [63]:
```
ttest_ind(DB_age,DH_age)
```

Out[63]:
```
Ttest_indResult(statistic=21.93417098536577, pvalue=1.088058828703349e-94)
```

In [64]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Two Sample t Test

- H_0: $\mu 1 = \mu 2$
- H_a: $\mu 1 \neq \mu 2$

Test statistic T = $\frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{{s_1}^2}{n1} + \frac{{s_2}^2}{n2}}}$

- where n1 and n2 are the sample sizes and X1 and X2 are the sample means
- ${S_1}^2$ and ${S_2}^2$ are sample variances

In [67]:
```
A=pd.read_table('/content/drive/My Drive/Statistics Mahesh Anand/HR.txt',index_col=0)
A.head()
```

Out[67]:

	Attrition	Age	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Employ
Individual										
Ind1	Yes	41	1	1102	1	1	2	1	1	
Ind2	No	49	2	279	2	8	1	1	1	
Ind3	Yes	37	1	1373	2	2	2	6	1	
Ind4	No	33	2	1392	2	3	4	1	1	
Ind5	No	27	1	591	2	2	1	2	1	

In [68]:
```
A.loc['Ind400']
```

Out[68]:
```
Attrition                         No
Age                               31
BusinessTravel                     1
DailyRate                        329
Department                         2
DistanceFromHome                   1
Education                          2
EducationField                     1
EmployeeCount                      1
EmployeeNumber                   530
EnvironmentSatisfaction            4
Gender                             1
HourlyRate                        98
JobInvolvement                     2
JobLevel                           1
JobRole                            3
JobSatisfaction                    1
MaritalStatus                      2
MonthlyIncome                   2218
MonthlyRate                    16193
NumCompaniesWorked                 1
OverTime                           2
PercentSalaryHike                 12
PerformanceRating                  3
RelationshipSatisfaction           3
StandardHours                     80
StockOptionLevel                   1
TotalWorkingYears                  4
TrainingTimesLastYear              3
WorkLifeBalance                    3
YearsAtCompany                     4
YearsInCurrentRole                 2
YearsSinceLastPromotion            3
YearsWithCurrManager               2
Name: Ind400, dtype: object
```

In [69]:
```
A.shape
```

Out[69]:
```
(1470, 34)
```

In [83]:
```
A['Gender'].value_counts()
```

Out[83]:
```
1    882
2    588
Name: Gender, dtype: int64
```

- Verify the avg sal of employee who left the org = avg sal of emp who currently working in the org
- Ho: pop_mean_sal (left)=pop_mean_sal (currently working)
- Ha: not equal

In [77]:
```
df_yes=A[A['Attrition']=='Yes']
df_no=A[A['Attrition']=='No']
```

In [84]:
```
df=A.groupby('Gender')
df1=df.get_group(1)
df2=df.get_group(2)
df1.shape,df2.shape
```

Out[84]:
```
((882, 34), (588, 34))
```

In [78]:
```
ttest_ind(df_yes['MonthlyIncome'],df_no['MonthlyIncome'])
```

Out[78]:
```
Ttest_indResult(statistic=-6.203935765608938, pvalue=7.14736398535381e-10)
```

In [85]:
```
ttest_ind(df1['MonthlyIncome'],df2['MonthlyIncome'])
```

Out[85]:
```
Ttest_indResult(statistic=-1.2212617308870655, pvalue=0.22218303455087898)
```

In [86]:
```
df1['MonthlyIncome'].mean()
```

Out[86]:
```
6380.507936507936
```

In [87]:
```
df2['MonthlyIncome'].mean()
```

Out[87]:
```
6686.566326530612
```

In [79]:
```
df_yes['MonthlyIncome'].mean()
```

Out[79]:
```
4787.0928270042195
```

In [80]:
```
df_no['MonthlyIncome'].mean()
```

Out[80]:
```
6832.739659367397
```

In [ ]:
```
A.columns
```

Out[37]:
```
Index(['Attrition', 'Age', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [ ]:
```
A.shape
```

Out[38]:
```
(1470, 34)
```

In [ ]:
```
A['Attrition'].value_counts()
```

Out[39]:
```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [ ]:
```
df_yes=A[A['Attrition']=='Yes']
df_no=A[A['Attrition']=='No']
df_yes.shape,df_no.shape
```

Out[68]:
```
((237, 34), (1233, 34))
```

In [ ]:
```
#OR
DF=A.groupby('Attrition')
df1=DF.get_group('Yes')
df2=DF.get_group('No')
df1.shape,df2.shape
```

Out[67]:
```
((237, 34), (1233, 34))
```

In [ ]:
```
sal_yes=df_yes['MonthlyIncome']
sal_no=df_no['MonthlyIncome']
```

In [ ]:
```
np.mean(sal_yes),np.mean(sal_no)
```

Out[72]:
```
(4787.0928270042195, 6832.739659367397)
```

In [ ]:
```
ttest_ind(sal_no,sal_yes)
```

Out[71]:
```
Ttest_indResult(statistic=6.203935765608938, pvalue=7.14736398535381e-10)
```

Verify the average sal of employees who left the organisation is same as avg sal of emp currently working in organisation

In [ ]:
```
sal_yes=A['MonthlyIncome'][A['Attrition']=='Yes']
sal_no=A['MonthlyIncome'][A['Attrition']=='No']
```

In [ ]:
```
ttest_ind(sal_yes,sal_no)
```

Out[42]:
```
Ttest_indResult(statistic=-6.203935765608938, pvalue=7.14736398535381e-10)
```

p-val<5%(0.05), which implies,we are rejecting Ho, ie,avg sal is not equal (significant) difference

In [ ]:
```
np.mean(sal_yes),np.mean(sal_no)
```

Out[43]:
```
(4787.0928270042195, 6832.739659367397)
```

Since,p-val <0.05 (5%) which falls in the acceptence zone of Ha. Hence we reject Ho with 95% confidence. There is a statistical evidence to say the two means are significantly different

In [ ]:
```
A.columns
```

Out[9]:
```
Index(['Attrition', 'Age', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

In [ ]:
```
sal_male=A['MonthlyIncome'][A['Attrition']=='Yes']
sal_female=A['MonthlyIncome'][A['Attrition']=='No']
```

In [ ]:
```
A['Gender'].value_counts()
```

Out[10]:
```
1    882
2    588
Name: Gender, dtype: int64
```

In [ ]:
```
sal_male=A['MonthlyIncome'][A['Gender']==1]
sal_female=A['MonthlyIncome'][A['Gender']==2]
```

In [ ]:
```
ttest_ind(sal_male,sal_female)
```

Out[45]:
```
Ttest_indResult(statistic=-1.2212617308870655, pvalue=0.22218303455087898)
```

p-val>5%(0.05) infact it is 22%, we have strong evidence to accept Ho (Sal_male_emp = Sal_female_emp)

In [ ]:
```
np.mean(sal_male),np.mean(sal_female)
```

Out[16]:
```
(6380.507936507936, 6686.566326530612)
```

Since p-value >0.05 (5%) it falls in acceptence zone of Ho, ie., average salary of male and female are nearly same.