PARAMETRIC HYPOTHESIS TESTS

In [1]:

```
import pandas as pd
import numpy as np
from datetime import date
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("Company_Data.csv",index_col=0)
```

One SAMPLE T-TEST

#The average age of Company customers is 52 at alpha = 0.05.

In [45]:

```
Sample = data.sample(n=29,replace=False)
x = 52
alpha = 0.05
H0 = " The average age of Customers is 52."
H1 = " The Average age of customers is not equal to 52 "
sample_mean = Sample["Age"]
print("Avg age in a sample is = ",Sample["Age"].mean())
from scipy import stats
t,p = stats.ttest_1samp(sample_mean,x)

print("Statistic test value is = ",t)
print("P-value is = ",p)

if p<alpha:
    print(H0)
else:
    print(H1)
```

```
Avg age in a sample is =  50.13793103448276
Statistic test value is =  -0.9465015739114689
P-value is =  0.3519901616647836
 The Average age of customers is not equal to 52
```

In [ ]:

```

```

INDEPENDENT TWO-SAMPLE T-TEST

Q) Does the product type hinders the purchasing of products between individuals?

In [36]:

```
df1 =data.loc[data["Marital_Status"]=="Single"]  # Population-1
df2 =data.loc[data["Marital_Status"]=="Married"]  #Population-2
Sample1 = df1.sample(n=10,replace=False)     # Sample data of size < 30 for 2 sample,T-test
Sample2 = df2.sample(n=10,replace=False)
def Independent_TT_test(a,b):

    Sq1 =a**2      #Square of a
    T1 = Sq1.sum()     #Total Sum of Square a
    Sq2 = b**2     #Square of b
    T2 = Sq2.sum()     #Total Sum of Square b

    S1 = a.sum()  # Total Sum of elements in a
    S2 = b.sum()  # Total Sum of elements in b

    X1 = a.mean()  #Mean of a
    X2 = b.mean()  #Mean of b
    n1 = len(a)    #Size of a
    n2 = len(b)    #Size of b
    dof = (n1-1) + (n2-1)

    num1 =(n1*T1)-(S1**2)  #Test statistic for sample-1
    den1 = n1*(n1-1)
    St1 =num1/den1

    num2 =(n2*T2)-(S2**2) #Test Statistic for sample-2
    den2 = n2*(n2-1)
    St2 =num2/den2

    num3 =  ((n1-1)*St1) + ((n2-1)*St2) # Test Statistic value calculation
    den3 = (n1-1)+(n2-1)
    Sp2 = num3/den3
    n = (1/n1) + (1/n2)
    Sp = np.sqrt(Sp2) *np.sqrt(n)
    t = X1-X2/Sp
    #Converting series to list to prepare dataframe
    a = a.tolist()
    Sq1 = Sq1.tolist()
    b = b.tolist()
    Sq2 =Sq2.tolist()
    data = {'a':a, 'Sq1':Sq1, 'b':b, 'Sq2':Sq2 }
    df = pd.DataFrame(data)
    print(df)
    print("Total: ", S1," ",T1," ",S2," ",T2,"\n")
    print(" Sample-1 test value S1-square is ",St1)
    print(" Sample-2 test value S2-square is ",St2)
    print("dof is = ",dof)
    return Sp,t;



if __name__ == "__main__":

    Sweets = Sample1['MntFishProducts'] # Sample-1
    Fruits = Sample2['MntMeatProducts']        #Sample-2

    H0 = "μ1 > μ2 Mean Amount spent on FISH products is less than or equal to MEAT."
    H1 = "μ1 <= μ2 Mean Amount spent on Fish products is greater than or equal to MEAT."
```

```
    Sp,t = Independent_TT_test(Sweets,Fruits)
    t_critical = 1.7341   # At n=18 1 Tail alpha =0.05
    print("\n Sp = ",Sp)
    print("Test Statistic Value is = ",t)
    if(t<=0):
        t =-t
    print("\n -CONCLUSION :")
    if(t>t_critical):
        print(H0)
    else:
        print(H1)
```

```
     a    Sq1    b     Sq2
0    0      0  413  170569
1   56   3136   19     361
2    3      9   44    1936
3   50   2500    6      36
4    2      4   15     225
5    0      0  134   17956
6    2      4  104   10816
7   94   8836   40    1600
8   42   1764   57    3249
9  110  12100  170   28900
Total:  359   28353   1002   235648

 Sample-1 test value S1-square is  1718.3222222222223
 Sample-2 test value S2-square is  15027.511111111111
dof is =  18

 Sp =  40.92167314924126
Test Statistic Value is =  33.45141976637732

 -CONCLUSION :
µ1 > µ2 Mean Amount spent on FISH products is less than or equal to MEAT.
```

In [ ]:

Dependent Two Sample T-TEST

In [ ]:

```
The Vegeterian people are Spending equal amount on Sweet and Fruits which are contibuting m
Does the amount spent on Fruits and Sweets are dependent.
```

In [37]:

```
Sample = data.sample(n=25,replace=False)

def Dependent_TT_test(a,b,mu):
    d=a-b
    D = d**2
    Td = d.sum()
    TD = D.sum()
    n=len(a)
    D_avg = Td/n

    num = (n*TD) - (Td**2)
    den =n*(n-1)
    Sd = np.sqrt(num/den)
    Sd_avg = Sd / np.sqrt(n)
    t = (D_avg - mu)/ Sd_avg

    return Sd,t


if __name__ == "__main__":

    Sweets = Sample['MntSweetProducts'] # Sample-1
    Fruits = Sample['MntFruits']        #Sample-2

    H0 = "µD = 0 No differnece in Amount spent on Sweet products and Fuits."
    H1 = "µD != 0 There is difference in Amount spent on Sweet products and Fuits."
    Mu_hyp = 0
    Sd,t = Dependent_TT_test(Sweets,Fruits,Mu_hyp)

    t_critical = 1.7341   # At n=18 1 Tail alpha =0.05
    print("\n Sp = ",Sp)
    print("Test Statistic Value is = ",t)

    print("\n -CONCLUSION :")
    if(t<=0):
        t =-t
    if(t>t_critical):
        print(H0)
    else:
        print(H1)
```

```
 Sp =  40.92167314924126
Test Statistic Value is =  -0.24801785651577193

 -CONCLUSION :
µD != 0 There is difference in Amount spent on Sweet products and Fuits.
```

In [ ]:

In [ ]:

```
```

In [16]:

```
lst =['MntWines', 'MntFruits','MntMeatProducts', 'MntFishProducts', 'MntSweetProducts','Mnt
data[lst].describe()
```

Out[16]:

	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	MntGo
count	2237.000000	2237.000000	2237.000000	2237.000000	2237.000000	2237
mean	303.995530	26.270451	166.916853	37.523022	27.068842	43
std	336.574382	39.715972	225.661158	54.639909	41.293949	52
min	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	24.000000	1.000000	16.000000	3.000000	1.000000	9
50%	174.000000	8.000000	67.000000	12.000000	8.000000	24
75%	504.000000	33.000000	232.000000	50.000000	33.000000	56
max	1493.000000	199.000000	1725.000000	259.000000	263.000000	362

In [ ]:

```
```

ONE SAMPLE Z-TEST

In [ ]:

```
# Avg number of web purchases is greater than 3.
```

In [82]:

```
#Hypothesis
H0 = "μ <= 3 Average number of web purchases is LESS than or equal to 3."
H1 = "μ > 3 Average number of web purchases is GREATER than or equal to 3."

# Data values
z_critic = 1.645 # AT alpha =0.05 One tailed test
mu = 3
std  = data['NumWebPurchases'].std()    #Standard deviation of sample
X =data['NumWebPurchases'].mean()       # Sample Mean
n = len(data['NumWebPurchases'])

#Calculation
standard_error = std/np.sqrt(n)
z = (X-mu)/standard_error
print("Z-Value is = ",z)

if(z<0):
    z=-z
if(z<z_critic):
    print(H0)
elif(z>z_critic):
    print("Reject ", H0)
```

```
Z-Value is =  18.4999262909087
Reject  μ <= 3 Average number of web purchases is LESS than or equal to 3.
```

In [ ]:

```

```

TWO SAMPLE Z TEST

#Average amount spent on wines is greater for Singles than Married.

In [83]:

```
# SAMPLES
df1 = data.loc[data["Marital_Status"]=="Single"]   # Sample-1
df2 = data.loc[data["Marital_Status"]=="Married"]  #Sample-2

#Hypothesis
H0 = "μ1 <= μ2 Average amount spent on wines is less than Average Amount spent on Gold."
H1 = "μ1 > μ2 Average amount spent on wines is greater than Average Amount spent on Gold."

# Data values
z_critic = 1.645 # AT alpha =0.05 One tailed test

# mu1 - mu2 = 0

x1 = df1['MntWines'].mean()    #Sample Mean
x2 = df2['MntWines'].mean()
std1 = df1['MntWines'].std()   #Standard Deviations
std2 = df2['MntWines'].std()
n1 = len(df1['MntWines'])
n2 = len(df2['MntWines'])
e1 =(std1**2)/n1               #error
e2 =(std2**2)/n2

#CALCULATION
standard_error =np.sqrt(e1+e2)
z = (x1-x2)/standard_error
print("Z-Value is = ",z)

if(z<0):
    z=-z
if(z<z_critic):
    print(H0)
elif(z>z_critic):
    print("Reject ", H0)
```

```
Z-Value is =  0.35559057941777744
14.882854294937603
μ1 <= μ2 Average amount spent on wines is less than Average Amount spent on
Gold.
```

In [ ]:

```
```

One Sample Z-TEST for Proportions

#There are 20% of customers who accepted the offer atleast once.Checking if it is true

In [126]:

```
df =data
df["NumAccepted"] = df['AcceptedCmp1']+df['AcceptedCmp2']+df['AcceptedCmp3']+df['AcceptedCm
df["Accepted"] = np.where( df["NumAccepted"]>0,1,0)   #Accepted atleast 1 offer after all c

# Hypothesis
H0 = "p = 0.2 There are 20% of customers who accepted the offer atleast once"
H1 = "p != 0.2 The customers who accepted the offer atleast once is not equal to 0.2."

# Data values
z_critic = 1.645 # AT alpha =0.05 One tailed test

p=0.2
q=1-p
x = len(df.loc[df["Accepted"]==1])
n = len(df)
proportion = x/n
z = (proportion-p)/np.sqrt((p*q)/n)
print("Z-Value is = ",z)
if(z<0):
    z=-z
if(z<z_critic):
    print(H0)
elif(z>z_critic):
    print("Reject ", H0)
```

```
Z-Value is =  0.771720207508393
p = 0.2 There are 20% of customers who accepted the offer atleast once
```

In [ ]:

```

```

Two Sample Z-TEST for Proportions

#There is difference in proportions of accepting Campaign offers between singles and married

In [130]:

```
H0 = "P1-P2 = 0 There is no difference in proportions of accepting Campaign offers between
H1 = "P1-P2 != 0 There is difference in proportions of accepting Campaign offers between si
df =data
df["NumAccepted"] = df['AcceptedCmp1']+df['AcceptedCmp2']+df['AcceptedCmp3']+df['AcceptedCm
df["Accepted"] = np.where( df["NumAccepted"]>0,1,0)   #Accepted atleast 1 offer after all c

# Population-1
df1 =df.loc[data["Marital_Status"]=="Single"]
n1 = len(df1)          #Number of singles
x1 = len(df1.loc[df1["Accepted"]==1])          #Number of singles who accepted offer
p1 =x1/n1  # Proportion-1

#Population-2
df2 =df.loc[data["Marital_Status"]=="Married"]
n2 = len(df2)            #Number of Married Customers
x2 = len(df.loc[df["Accepted"]==1])        #Number of Married Customers who accepted offer
p2=x2/n2  #Proportion-2

p =(x1+x2)/(n1+n2)
q =1-p

# p1 ^-p2^ = 0

den = (p*q)*(1/n1 + 1/n2)
z = (p1-p2)/np.sqrt(den)
z_critical = 1.960    # AT alpha =0.05 Two tailed test
print("Z-Value is = ",z)
if(z<0):
    z=-z

print("\n -----> CONCLUSION :\n")
if(z<z_critic):
    print(H0)
elif(z>z_critic):
    print(H1)
```

```
Z-Value is =  -5.661516235110674

 -----> CONCLUSION :

P1-P2 != 0 There is difference in proportions of accepting Campaign offers b
etween singles and married
```

In [ ]:

In [ ]: