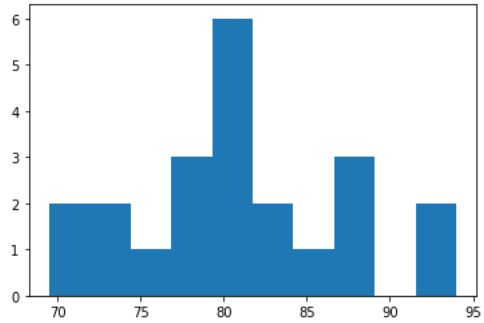
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import statsmodels.api as sm
import pylab
from statsmodels.stats.proportion import proportions_ztest
import statsmodels.api as sm
from statsmodels.formula.api import ols

data = np.array([94. , 84.9, 82.6, 69.5, 80.1, 79.6, 81.4, 77.8, 81.
```

```
##Check the normality
#H₀: The data is normally distributed.
#H₁: The data is not normally distributed.
import matplotlib.pyplot as plt
from scipy import stats
plt.hist(data)
plt.show()
```




```
stats.shapiro(data)   ##Reject the null hypothesis
```

```
ShapiroResult(statistic=0.9676006436347961, pvalue=0.6555852293968201)
```

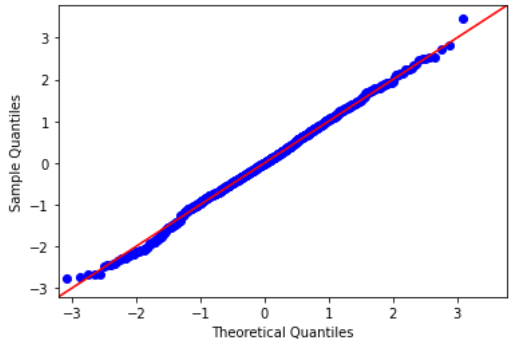
```
data1 = np.random.normal(size=1000)
plt.hist(data1)
plt.show()
#stats.shapiro(data1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-fd7201c8fc23> in <module>
----> 1 data1 = np.random.normal(size=1000)
      2 plt.hist(data1)
      3 plt.show()
      4 #stats.shapiro(data1)

NameError: name 'np' is not defined
```

SEARCH STACK OVERFLOW

```
import statsmodels.api as sm
import pylab
sm.qqplot(data1,line='45')
pylab.show()
```




```
##Homoginity of variance
##H₀: The variances of the samples are the same.
##H₁: The variances of the samples are different.

df = np.array([94. , 84.9, 82.6, 69.5, 80.1, 79.6, 81.4, 77.8, 81.7,
df1 = np.array([77.1, 71.7, 91. , 72.2, 74.8, 85.1, 67.6, 69.9, 75.3
stats.levene(df,df1)  ##Fail to reject null hypothesis
```

```
LeveneResult(statistic=0.05566544473596216, pvalue=0.8148984115435045)
```

```
##Testing of hypothesis for mean
##The null hypothesis is that μ ≥ 10000. We begin with computing the
xbar = 9900             # sample mean
mu0 = 10000             # hypothesized value
sigma = 120             # population standard deviation
n = 30                  # sample size
se = sigma/np.sqrt(n)
Z_cal = (xbar-mu0)/se
print(Z_cal)


alpha=0.05
stats.norm.ppf(1-alpha)
z_table=-(stats.norm.ppf(1-alpha))
print(z_table)  ##Reject Ho
##or
print(stats.norm.cdf(-4.564))  ## p value <0.05
t_table = stats.t.ppf(1-alpha,29)
print(-t_table)
```

```
-4.564354645876384
-1.6448536269514722
2.5094039266298674e-06
-1.6991270265334972
```

```
df = np.array([10,12,13,14,15,2,7,8])
print(stats.ttest_1samp(df,popmean=12,alternative='two-sided'))  ##{
```

```
##or
print(stats.t.cdf(-1.2264982650902285,7)*2)
print(stats.t.ppf(0.05,7))  ##accept H0
```

```
Ttest_1sampResult(statistic=-1.2264982650902285, pvalue=0.25966236017428784)
0.25966236017428784
-1.8945786050613054
```

```
##Testing of hypothesis for Proportion
##The null hypothesis is that p ≥ 0.6. We begin with computing the te

pbar = 85/148          # sample proportion
p = 0.6                # hypothesized value
n = 148                # sample size
se = np.sqrt((p*(1-p))/n)
Z_cal = (pbar-p)/se
print(Z_cal)
alpha=0.05
stats.norm.ppf(1-alpha)
z_table=-(stats.norm.ppf(1-alpha))
print(z_table)  ##Reject Ho
t_table = stats.t.ppf(1-alpha,147)
print(-t_table)  ##accept ho
```

```
-0.6375982524533221
-1.6448536269514722
-1.6552854366066903
```

```
from statsmodels.stats.proportion import proportions_ztest
proportions_ztest(count=85, nobs=148,value=0.6,alternative = "smaller
```

```
(-0.6317346770183859, 0.2637801323087673)
```

```
##Testing of hypothesis for two mean with SD known/ unequal var
##The null hypothesis is that μ1-μ1 = 0. We begin with computing the
xbar1 = 121            # sample mean
xbar2 = 112
#mu1-mu2 = 0            # hypothesized value
sigma1 = 8           # population standard deviation
sigma2 = 8
n1 = 10                # sample size
n2 = 10
se = np.sqrt((sigma1**2/n1)+(sigma2**2/n2))
Z_cal = (xbar1-xbar2)/se
print(Z_cal)
alpha=0.05
stats.norm.ppf(1-alpha)
z_table=(stats.norm.ppf(1-alpha))
print(z_table)  ##Reject Ho
##or
print(1-stats.norm.cdf(2.515))  ## p value <0.05
```

```
t_table = stats.t.ppf(1-(alpha/2),18)  ##two sided
print(t_table)
```

```
2.5155764746872635
1.6448536269514722
0.005951619189523916
2.10092204024096
```

```
##Testing of hypothesis for two mean with SD uknown
## Lab assignement

a = np.array([56, 128.6, 12, 123.8, 64.34, 78, 763.3])
b = np.array([1.1, 2.9, 4.2])
print(stats.ttest_ind(a,b,equal_var=True))
print(stats.ttest_ind(a,b,equal_var=False))
stats.t.ppf(0.025,8)  ##two sided that's why 0.025
```

```
Ttest_indResult(statistic=1.099305186099593, pvalue=0.30361296704535845)
Ttest_indResult(statistic=1.7380929645474241, pvalue=0.13285209620970656)
-2.306004135033371
```

```
##Type I error and Type II error
##The null hypothesis is that μ = 50. We begin with computing the tes
xbar = 48.5            # sample mean
mu0 = 50           # hypothesized value
se = 0.79            # standard error
n = 30               # sample size
Z_cal = (xbar-mu0)/se
if Z_cal<0:
  print(stats.norm.cdf(Z_cal))
else:
  print(1-stats.norm.cdf(Z_cal))
```

```
0.02879971774715278
```

```
##Testing of hypothesis for two mean of depdent sample(Paired Student
a = np.array([56, 128, 12, 123, 64, 78, 763])
b = np.array([46,100,5,121,54,80,700])
print(stats.ttest_rel(a,b))

## t test for two independent sample
stats.ttest_ind(df,df1)  ##Rject null hypothesis
```

```
Ttest_indResult(statistic=2.8414882345796917, pvalue=0.007535984340826129)
```

```
##One_Way ANOVA
import pandas as pd
df = {"Drink1": [2,3,7,2,6],"Drink2":[10,8,7,5,10],"Drink3":[10,13,14
```

```
df=pd.DataFrame(data=df)
df
```

| | Drink1 | Drink2 | Drink3 |
|---|---|---|---|
| **0** | 2 | 10 | 10 |
| **1** | 3 | 8 | 13 |
| **2** | 7 | 7 | 14 |
| **3** | 2 | 5 | 13 |
| **4** | 6 | 10 | 15 |

```
df1 = pd.melt(df.reset_index(),id_vars = ["index"],value_vars=["Drinl
df1.columns=["index","Treatments","Value"]
print(df1)
```

```
    index Treatments  Value
0       0     Drink1      2
1       1     Drink1      3
2       2     Drink1      7
3       3     Drink1      2
4       4     Drink1      6
5       0     Drink2     10
6       1     Drink2      8
7       2     Drink2      7
8       3     Drink2      5
9       4     Drink2     10
10      0     Drink3     10
11      1     Drink3     13
12      2     Drink3     14
13      3     Drink3     13
14      4     Drink3     15
```

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

fit1 = ols('Value ~ Treatments',data=df1).fit()
anova1 = sm.stats.anova_lm(fit1,typ=1)
anova1
```

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| **Treatments** | 2.0 | 203.333333 | 101.666667 | 22.592593 | 0.000085 |
| **Residual** | 12.0 | 54.000000 | 4.500000 | NaN | NaN |

```
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.formula.api import ols
from statsmodels.base.model import Model
from statsmodels.stats.multicomp import pairwise_tukeyhsd
tukey = pairwise_tukeyhsd(df1["Value"],groups=df1["Treatments"])
tukey._results_table
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

| group1 | group2 | meandiff | p-adj | lower | upper | reject |
|---|---|---|---|---|---|---|
| Drink1 | Drink2 | 4.0 | 0.0286 | 0.4224 | 7.5776 | True |
| Drink1 | Drink3 | 9.0 | 0.001 | 5.4224 | 12.5776 | True |
| Drink2 | Drink3 | 5.0 | 0.0075 | 1.4224 | 8.5776 | True |

```
##Two Way ANOVA
df = pd.DataFrame({'water': np.repeat(['daily', 'weekly'], 15),
                   'sun': np.tile(np.repeat(['low', 'med', 'high'], !
                   'height': [6, 6, 6, 5, 6, 5, 5, 6, 4, 5,
```

```
    6, 6, 7, 8, 7, 3, 4, 4, 4, 5,
    4, 4, 4, 4, 4, 5, 6, 6, 7, 8]})
```

df

| | water | sun | height |
|---|---|---|---|
| **0** | daily | low | 6 |
| **1** | daily | low | 6 |
| **2** | daily | low | 6 |
| **3** | daily | low | 5 |
| **4** | daily | low | 6 |
| **5** | daily | med | 5 |
| **6** | daily | med | 5 |
| **7** | daily | med | 6 |
| **8** | daily | med | 4 |
| **9** | daily | med | 5 |
| **10** | daily | high | 6 |
| **11** | daily | high | 6 |
| **12** | daily | high | 7 |
| **13** | daily | high | 8 |
| **14** | daily | high | 7 |
| **15** | weekly | low | 3 |
| **16** | weekly | low | 4 |
| **17** | weekly | low | 4 |
| **18** | weekly | low | 4 |
| **19** | weekly | low | 5 |
| **20** | weekly | med | 4 |
| **21** | weekly | med | 4 |
| **22** | weekly | med | 4 |
| **23** | weekly | med | 4 |
| **24** | weekly | med | 4 |
| **25** | weekly | high | 5 |
| **26** | weekly | high | 6 |
| **27** | weekly | high | 6 |
| **28** | weekly | high | 7 |
| **29** | weekly | high | 8 |

```
fit2 = ols('height ~ C(water) + C(sun) + C(water):C(sun)', data=df).
anova2 = sm.stats.anova_lm(fit2,typ=2)
print(anova2)
```

```
                   sum_sq    df        F    PR(>F)
C(water)         8.533333   1.0  16.0000  0.000527
C(sun)          24.866667   2.0  23.3125  0.000002
C(water):C(sun)  2.466667   2.0   2.3125  0.120667
Residual        12.800000  24.0      NaN       NaN
```

```
fit2 = ols('height ~ sun', data=df).fit()
pw = fit2.t_test_pairwise("sun",method="sh")
pw.result_frame
```

| | coef | std err | t | P>\|t\| | Conf. Int. Low | Conf. Int. Upp. | pvalue-sh | reject-sh |
|---|---|---|---|---|---|---|---|---|
| low-high | -1.7 | 0.419877 | -4.048809 | 0.000389 | -2.561515 | -0.838485 | 0.000778 | True |
| mod | | | | | | | | |

```
##one way MANOVA
import pandas as pd
from statsmodels.multivariate.manova import MANOVA
url = 'https://vincentarelbundock.github.io/Rdatasets/csv/datasets/ir
df = pd.read_csv(url, index_col=0)
df.columns = df.columns.str.replace(".", "_")
df.head()
```

```
<ipython-input-2-77fb22b7d9cb>:6: FutureWarning: The default value of regex will char
  df.columns = df.columns.str.replace(".", "_")
```

| | Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Species |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 1 to 150
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Sepal_Length  150 non-null    float64
 1   Sepal_Width   150 non-null    float64
 2   Petal_Length  150 non-null    float64
 3   Petal_Width   150 non-null    float64
 4   Species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 7.0+ KB
```

```
maov = MANOVA.from_formula('Sepal_Length + Sepal_Width + Petal_Length
print(maov.mv_test())
```

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0170 4.0000 144.0000 2086.7720 0.0000
         Pillai's trace  0.9830 4.0000 144.0000 2086.7720 0.0000
 Hotelling-Lawley trace 57.9659 4.0000 144.0000 2086.7720 0.0000
    Roy's greatest root 57.9659 4.0000 144.0000 2086.7720 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
        Species          Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0234 8.0000 288.0000  199.1453 0.0000
         Pillai's trace  1.1919 8.0000 290.0000   53.4665 0.0000
 Hotelling-Lawley trace 32.4773 8.0000 203.4024  582.1970 0.0000
    Roy's greatest root 32.1919 4.0000 145.0000 1166.9574 0.0000
================================================================
```

```
import pandas as pd
df = pd.read_csv("https://raw.githubusercontent.com/researchpy/Data-s

df.drop('person', axis= 1, inplace= True)
```

```
# Recoding value from numeric to string
df['dose'].replace({1: 'placebo', 2: 'low', 3: 'high'}, inplace= Tru

df.head()
```

| | dose | libido |
|---|---|---|
| **0** | placebo | 3 |
| **1** | placebo | 2 |
| **2** | placebo | 1 |
| **3** | placebo | 1 |
| **4** | placebo | 4 |

```
import statsmodels.api as sm
from statsmodels.formula.api import ols

model = ols('libido ~ dose', data=df).fit()
aov_table = sm.stats.anova_lm(model, typ=1)
aov_table
```

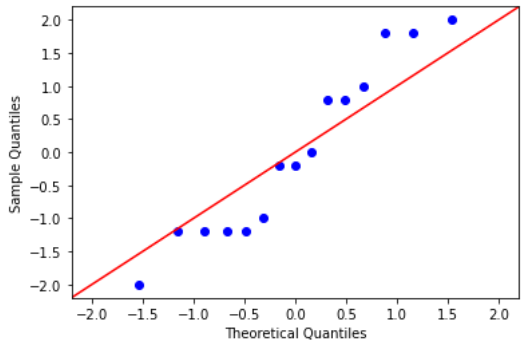
| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| **dose** | 2.0 | 20.133333 | 10.066667 | 5.118644 | 0.024694 |
| **Residual** | 12.0 | 23.600000 | 1.966667 | NaN | NaN |

```
import scipy.stats as stats

stats.shapiro(model.resid)   ##normaly distributed
```

```
ShapiroResult(statistic=0.916691780090332, pvalue=0.1714704930782318)
```

```
import statsmodels.api as sm
import pylab
sm.qqplot(model.resid,line='45')
pylab.show()
```




```
stats.levene(df['libido'][df['dose'] == 'high'],
             df['libido'][df['dose'] == 'low'],
             df['libido'][df['dose'] == 'placebo'])   ##homogenity te
```

```
LeveneResult(statistic=0.11764705882352934, pvalue=0.8900225182757423)
```

```
import statsmodels.stats.multicomp as mc

comp = mc.MultiComparison(df['libido'], df['dose'])
post_hoc_res = comp.tukeyhsd()
post_hoc_res.summary()
```

Multiple Comparison of Means - Tukey HSD, FWER=0.05

| group1 | group2 | meandiff | p-adj | lower | upper | reject |
|---|---|---|---|---|---|---|
| high | low | -1.8 | 0.1472 | -4.1651 | 0.5651 | False |
| high | placebo | -2.8 | 0.0209 | -5.1651 | -0.4349 | True |
| low | placebo | -1.0 | 0.5171 | -3.3651 | 1.3651 | False |

```
a = np.array([56, 128.6, 12, 123.8, 64.34, 78, 763.3])
b = np.array([1.1, 2.9, 4.2])
print(stats.ttest_ind(IND,PAK,equal_var=FALSE,popmean=0,alternative =
```