Hypothesis testing

Hypothesis testing is a statistical analysis that uses sample data to assess two mutually exclusive theories about the properties of a population. Statisticians call these theories the null hypothesis and the alternative hypothesis.

What Are T Values and P Values in Statistics?

The p-value explained

The p-value is a number, calculated from a statistical test, that describes how likely you are to have found a particular set of observations if the null hypothesis were true.

P-values are used in hypothesis testing to help decide whether to reject the null hypothesis. The smaller the p-value, the more likely you are to reject the null hypothesis.

T Values

The t-value measures the size of the difference relative to the variation in your sample data. Put another way, T is simply the calculated difference represented in units of standard error

Remember, the t-value in your output is calculated from only one sample from the entire population. It you took repeated random samples of data from the same population, you'd get slightly different t-values each time, due to random sampling error

One tail test

In [19]:

```
data=pd.Series([0.459,0.193, 0.142, 0.329, 0.691, 0.231, 0.793, 0.519, 0.392, 0.218])
```

In [22]:

```
#This gives two tail test p value so divide by 2 to get final p-value
import scipy
from scipy import stats
p=scipy.stats.ttest_1samp(data,0.3)[1]
p_value= p/2
p_value
```

Out[22]:

```
0.09833627738893307
```

SciPy in Python. SciPy in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to

manipulate the data and visualize the data using a wide range of high-level Python commands.

Two Tail test

In [25]:

```
Control=pd.Series([25,66,91, 87, 99, 77, 88, 91])
Treat =pd.Series([99,105,101, 110, 103, 93, 99, 103])
stats.ttest_ind(Control,Treat)
```

Out[25]:

```
Ttest_indResult(statistic=-2.7651393780790965, pvalue=0.015186589450336106)
```

In [ ]:

```

```

Correlation Tests

```
Pearson’s Correlation Coefficient
1] two samples have a linear relationship.

## Assumptions

1]each sample are independent and identically distributed.
2]each sample are normally distributed.
3]each sample have the same variance.
```

In [26]:

```
# Example of the Pearson's Correlation test
from scipy.stats import pearsonr
D1 = [0.893, 2.817, 0.121, 0.945, 0.055, -1.436, 0.360, -1.478]
D2 = [0.399, 3.517, 0.125, 7.545, 0.555, -1.536, 3.350, -1.578]
stat, p = pearsonr(D1,D2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Null Hypothesis[Probably independent]')
else:
    print('Alternate Hypothesis[Probably dependent]')
```

```
stat=0.658, p=0.076
Null Hypothesis[Probably independent]
```

Stationary Tests

In [ ]:

```
This section lists statistical tests that you can use to check if a time series is stationa
Augmented Dickey-Fuller Unit Root Test
```

In [ ]:

```
Assumptions

Observations in are temporally ordered.
```

In [2]:

```
# Example of the Augmented Dickey-Fuller unit root test
from statsmodels.tsa.stattools import adfuller
data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
stat, p, lags, obs, crit, t = adfuller(data)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Probably not Stationary')
else:
    print('Probably Stationary')
```

```
stat=1.496, p=0.998
Probably not Stationary
```

Parametric Statistical Hypothesis Tests

Student’s t-test

```
This section lists statistical tests that you can use to compare data samples.

Tests whether the means of two independent samples are significantly different.
two sample test.

Assumptions

1] sample are independent and population distribution is normal.
2] sample size is less than 30.
```

In [7]:

```
# Example of the Student's t-test
from scipy.stats import ttest_ind
D1 = [0.87, 2.81, 0.121, -0.94, -0.055, -1.46, 0.360, -1.478]
D2 = [1.14, -0.43, -0.938, -0.72, -0.86, -0.157, 0.500, 1.183]
stat, p = ttest_ind(D1,D2)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Null Hypothesis[Probably independent]')
else:
    print('Alternate Hypothesis[Probably dependent]')
```

```
stat=0.109, p=0.915
Null Hypothesis[Probably independent]
```

Analysis of Variance Test (ANOVA)

```
Tests whether the two or more independent samples are significantly different.
```

```
it is advance of T-test .
##Assumptions

1] sample are independent and identically distributed .
2] sample are normally distributed.
3] sample have the same variance
```

In [10]:

```
# Example of the Analysis of Variance Test
from scipy.stats import f_oneway
D1 = [ 3.81, 0.12, -0.94, -0.05, -1.43, 0.360, -1.478, -1.637, -1.99]
D2 = [2.42, -0.232, -0.938, -0.729, -0.846, -0.157, 0.500, 1.183, -1.055]
D3 = [-1.08, 0.296, 0.928, -1.148, -0.213, 0.229, 0.137, 0.229, -0.596]
stat, p = f_oneway(D1, D2, D3)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Null Hypothesis[Probably independent]')
else:
    print('Alternate Hypothesis[Probably dependent]')
```

```
stat=0.194, p=0.825
Null Hypothesis[Probably independent]
```

Chi-Squared Test

```
# Non-parametric Statistical Hypothesis Test.
1]two categorical variables are related or independent.

Assumptions
1] 50 or more examples in each cell of the contingency table.
```

In [9]:

```
# Example of the Chi-Squared Test
from scipy.stats import chi2_contingency
table1 = [[11, 50, 30,55,72],[8,  5, 44 ,45, 88]]
stat, p, dof, expected = chi2_contingency(table1)
print('stat=%.3f, p=%.3f' % (stat, p))
if p > 0.05:
    print('Null Hypothesis[Probably independent]')
else:
    print('Alternate Hypothesis[Probably dependent]')
```

```
stat=40.811, p=0.000
Alternate Hypothesis[Probably dependent]
```

In [12]:

```
import pandas as pd
df1=pd.read_csv("E:/HYPOSYNTHISIS/LabTAT.csv")
df1.head()
```

Out[12]:

	Laboratory 1	Laboratory 2	Laboratory 3	Laboratory 4
0	185.35	165.53	176.70	166.13
1	170.49	185.91	198.45	160.79
2	192.77	194.92	201.23	185.18
3	177.33	183.00	199.61	176.42
4	193.41	169.57	204.63	152.60

In [13]:

```
df1.describe()
```

Out[13]:

	Laboratory 1	Laboratory 2	Laboratory 3	Laboratory 4
count	120.000000	120.000000	120.000000	120.00000
mean	178.361583	178.902917	199.913250	163.68275
std	13.173594	14.957114	16.539033	15.08508
min	138.300000	140.550000	159.690000	124.06000
25%	170.335000	168.025000	188.232500	154.05000
50%	178.530000	178.870000	199.805000	164.42500
75%	186.535000	189.112500	211.332500	172.88250
max	216.390000	217.860000	238.700000	205.18000

In [15]:

```
df1.isna().sum()
```

Out[15]:

```
Laboratory 1    0
Laboratory 2    0
Laboratory 3    0
Laboratory 4    0
dtype: int64
```

In [14]:

```
stat, p, dof, expected = chi2_contingency(df1)
print('stat=%.3f, p=%.3f' % (stat,p))
if p>0.05:
    print('Null hypo,indepndant categories')
else:
    print('Non null,dependant categories')
```

```
stat=450.673, p=0.001
Non null,dependant categories
```

In [17]:

```
df1.columns
```

Out[17]:

```
Index(['Laboratory 1', 'Laboratory 2', 'Laboratory 3', 'Laboratory 4'], dtyp
e='object')
```

In [18]:

```
DF=df1[['Laboratory 1', 'Laboratory 2']]
DF
```

Out[18]:

	Laboratory 1	Laboratory 2
0	185.35	165.53
1	170.49	185.91
2	192.77	194.92
3	177.33	183.00
4	193.41	169.57
...	...	...
115	178.49	170.66
116	176.08	183.98
117	202.48	174.54
118	182.40	197.18
119	182.09	215.17

120 rows × 2 columns

In [ ]: