T_test

Continuous Data:

The t-test is appropriate for data that is continuous, meaning the data can take on any value within a range. Examples include height, weight, temperature, and test scores.

Normally Distributed Data:

Ideally, the data should be approximately normally distributed. This is particularly important for smaller sample sizes. For larger sample sizes (typically n > 30), the t-test is more robust to violations of normality due to the Central Limit Theorem.

Independent Samples:

For an independent samples t-test (also known as an unpaired t-test), the data should come from two independent groups. Examples include comparing the test scores of two different classes or the weights of two different species.

Paired Samples:

For a paired samples t-test (also known as a dependent t-test), the data should come from the same group at two different times or under two different conditions. Examples include before-and-after measurements on the same subjects or comparing the left and right hand strength of the same individuals.

In [ ]:
```
1:- One-Sample t-Test:
Used to compare the mean of a single sample to a known value or a theoretical expectation.

2:- Independent Samples t-Test:
Used to compare the means of two independent groups to see if there is evidence that the associated population means

3:- Paired Samples t-Test:
Used to compare the means of two related groups, such as measurements taken on the same subjects at two different tir
```

One_sample T test

In [1]:
```
import scipy.stats as stats

#sample Data
x= [1, 2, 3, 4, 5]

#known population mean
mu=3

#preform one sample t test

t_statistics, p= stats.ttest_1samp(x, mu)

#Print results
print("T_statistics : ", t_statistics)
print("P_value :", p)
```

```
T_statistics :  0.0
P_value : 1.0
```

In [2]:
```
#print the result using if condition
if p>0.05:
    print(f'p_value: {p}, sample mean is equal to the population mean (fail to reject H0)')
else:
    print(f'p_value: {p}, sample mean is not equal to the population mean (reject H0)')
```

```
p_value: 1.0, sample mean is equal to the population mean (fail to reject H0)
```

Two_sample t_test

In [5]:
```
# two sample t test

import scipy.stats as stats

# sample data

G1= [2.3, 3.4, 4.5, 2.3, 3.4]
G2= [1.2, 2.2, 3.2, 2.2, 2.3]

# preform independent two sample t_test

```

```
t_statistics, p= stats.ttest_ind(G1, G2)

# print the results
print("t_stats :", t_statistics)
print("p_value :", p)
```

```
t_stats : 1.8482055087756457
p_value : 0.10175647371829193
```

In [6]:
```
#print the result using if condition
if p>0.05:
    print(f'p_value: {p},G1 mean is equal to the G2 mean (fail to reject H0)')
else:
    print(f'p_value: {p}, G1 mean is not equal to the G2 mean (reject H0)')
```

```
p_value: 0.10175647371829193,G1 mean is equal to the G2 mean (fail to reject H0)
```

Paired sample t_test

In [9]:
```
import scipy.stats as stats

# sample data
before= [2, 3, 4, 5, 6]
after= [3, 4, 5, 6, 7]

# preform paired sample t_test

t_statistics, p= stats.ttest_rel(before, after)

# print the results
print("t_stats :", t_statistics)
print("p_value :", p)

#print the result using if condition
if p>0.05:
    print(f'p_value: {p},before mean is equal to the after mean (fail to reject H0)')
else:
    print(f'p_value: {p}, before mean is not equal to the after mean (reject H0)')
```

```
t_stats : -inf
p_value : 0.0
p_value: 0.0, before mean is not equal to the after mean (reject H0)
```

In T_Test the sample size always less then 30

S_Size>30

In [ ]: