Normal Distribution

Scipy.stats.norm

Cdf- cumulative density function **cdf(x, loc=0, scale=1)**	When you need to find out the probability that a randomly picked value would fall below a value **x** Cumulative Area under the curve (normal distribution) at left of x
Percent Point Function **ppf(q, loc=0, scale=1)**	Reverse of cdf. Calculates values corresponding to given q value. Gives percentile values.
pdf(x, loc=0, scale=1)	Probability density function. Calculates probability of a given value of x
Confidence Interval interval(alpha,loc=, scale=) alpha is confidence coefficient	Alpha is probability that the randomly picked value will lie in the given range

Student's t distribution

Scipy.stats.t

Df--- degrees of freedom

Cdf- cumulative density function **cdf(x,df,loc=0, scale=1)**	When you need to find out the probability that a randomly picked value would fall below a value **x** Cumulative Area under the curve (normal distribution) at left of x
Percent Point Function **ppf(q, df,loc=0, scale=1)**	Reverse of cdf. Calculates values corresponding to given q value. Gives percentile values.
pdf(x, df, loc=0, scale=1)	Probability density function. Calculates probability of a given value of x
Confidence Interval interval(alpha,df,loc=, scale=) alpha is confidence coefficient	Alpha is probability that the randomly picked value will lie in the given range

t test

1 sample t test

Compare means of one sample with a given mu0.

scipy.stats.ttest_1samp(*a*, *popmean*, *axis=0*, *nan_policy='propagate'*)

a -> sample values
popmean -> population mean (mu0)
axis -> 0 for rows 1 for columns along which axis the data has to be read.
nan_policy -> if there are any missing values in that case ho should they be treated for the test.

- 'propagate' returns nan,
- 'raise' throws an error,
- 'omit' performs the calculations ignoring nan values.

Default is 'propagate'.

Output is: t statistic and p value (for two tail)

2 sample t test

Compare means for the same variable of two samples or two subgroups.

Independent t test:

scipy.stats.ttest_ind(*a*, *b*, *axis=0*)

a → sample 1
b → sample 2

Paired t test

Compare two samples where respondents are same

scipy.stats.ttest_rel(a, b, axis=0)

a → sample 1
b → sample 2