



# Statistics

## Statistical tests



<code>ttest_1samp(a, popmean[, axis, nan_policy, ...])</code>	Calculate the T-test for the mean of ONE group of scores.
<code>ttest_ind(a, b[, axis, equal_var, ...])</code>	Calculate the T-test for the means of <i>two independent</i> samples of scores.
<code>ttest_ind_from_stats(mean1, std1, nobs1, ...)</code>	T-test for means of two independent samples from descriptive statistics.
<code>ttest_rel(a, b[, axis, nan_policy, alternative])</code>	Calculate the t-test on TWO RELATED samples of scores, a and b.
<code>chisquare(f_obs[, f_exp, ddof, axis])</code>	Calculate a one-way chi-square test.
<code>cramervonmises(rvs, cdf[, args])</code>	Perform the one-sample Cramér-von Mises test for goodness of fit.
<code>cramervonmises_2samp(x, y[, method])</code>	Perform the two-sample Cramér-von Mises test for goodness of fit.
<code>power_divergence(f_obs[, f_exp, ddof, axis, ...])</code>	Cressie-Read power divergence statistic and goodness of fit test.
<code>kstest(rvs, cdf[, args, N, alternative, mode])</code>	Performs the (one-sample or two-sample) Kolmogorov-Smirnov test for goodness of fit.
<code>ks_1samp(x, cdf[, args, alternative, mode])</code>	Performs the one-sample Kolmogorov-Smirnov test for goodness of fit.

... and many more!!!

# Theoretical background

- Normal distribution plot



Compare observed sample against the expected distribution

$H_0$  = sample was taken from distribution

$H_0$  can only be rejected or kept due to insufficient evidence against it

$H_0$  can NEVER be confirmed!

# Shapiro-Wilk test

## scipy.stats.shapiro

`scipy.stats.shapiro` (*x*, *a*=None, *reta*=False)

[\[source\]](#)

Perform the Shapiro-Wilk test for normality.

The Shapiro-Wilk test tests the null hypothesis that the data was drawn from a normal distribution.

**Parameters:** *x* : *array\_like*

Array of sample data.

*a* : *array\_like, optional*

Array of internal parameters used in the calculation. If these are not given, they will be computed internally. If *x* has length *n*, then *a* must have length *n*/2.

*reta* : *bool, optional*

Whether or not to return the internally computed *a* values. The default is False.

**Returns:**

*W* : *float*

The test statistic.

*p-value* : *float*

The p-value for the hypothesis test.

*a* : *array\_like, optional*

If *reta* is True, then these are the internally computed “*a*” values that may be passed into this function on future calls.

**See also:**

`anderson` The Anderson-Darling test for normality

`kstest` The Kolmogorov-Smirnov test for goodness of fit.

## Notes

The algorithm used is described in [\[R634\]](#) but censoring parameters as described are not implemented. For  $N > 5000$  the *W* test statistic is accurate but the p-value may not be.

The chance of rejecting the null hypothesis when it is true is close to 5% regardless of sample size.

<https://docs.scipy.org>

# Correlation

## scipy.stats.pearsonr

### scipy.stats.pearsonr(*x*, *y*)

[\[source\]](#)

Calculates a Pearson correlation coefficient and the p-value for testing non-correlation.

The Pearson correlation coefficient measures the linear relationship between two datasets. Strictly speaking, Pearson's correlation requires that each dataset be normally distributed. Like other correlation coefficients, this one varies between -1 and +1 with 0 implying no correlation. Correlations of -1 or +1 imply an exact linear relationship. Positive correlations imply that as *x* increases, so does *y*. Negative correlations imply that as *x* increases, *y* decreases.

The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets. The p-values are not entirely reliable but are probably reasonable for datasets larger than 500 or so.

**Parameters:** *x* : (N,) array\_like

Input

*y* : (N,) array\_like

Input

**Returns:** (Pearson's correlation coefficient,  
2-tailed p-value)

<https://docs.scipy.org>

# t-test

## scipy.stats.ttest\_ind

```
scipy.stats.ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate')
```

[\[source\]](#)

Calculates the T-test for the means of *two independent* samples of scores.

This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that the populations have identical variances by default.

**Parameters:** **a, b** : *array\_like*

The arrays must have the same shape, except in the dimension corresponding to *axis* (the first, by default).

**axis** : *int or None, optional*

Axis along which to compute test. If None, compute over the whole arrays, *a*, and *b*.

**equal\_var** : *bool, optional*

If True (default), perform a standard independent 2 sample test that assumes equal population variances [\[R643\]](#). If False, perform Welch's t-test, which does not assume equal population variance [\[R644\]](#).

*New in version 0.11.0.*

**nan\_policy** : {'propagate', 'raise', 'omit'}, *optional*

Defines how to handle when input contains nan. 'propagate' returns nan, 'raise' throws an error, 'omit' performs the calculations ignoring nan values. Default is 'propagate'.

**Returns:**

**statistic** : *float or array*

The calculated t-statistic.

**pvalue** : *float or array*

The two-tailed p-value.

### Notes

We can use this test, if we observe two independent samples from the same or different population, e.g. exam scores of boys and girls or of two ethnic groups. The test measures whether the average (expected) value differs significantly across samples. If we observe a large p-value, for example larger than 0.05 or 0.1, then we cannot reject the null hypothesis of identical average scores. If the p-value is smaller than the threshold, e.g. 1%, 5% or 10%, then we reject the null hypothesis of equal averages.

<https://docs.scipy.org>



# Wilcoxon test

## scipy.stats.wilcoxon

**scipy.stats.wilcoxon**(*x*, *y=None*, *zero\_method='wilcox'*, *correction=False*)

[\[source\]](#)

Calculate the Wilcoxon signed-rank test.

The Wilcoxon signed-rank test tests the null hypothesis that two related paired samples come from the same distribution. In particular, it tests whether the distribution of the differences  $x - y$  is symmetric about zero. It is a non-parametric version of the paired T-test.

**Parameters:** *x* : *array\_like*

The first set of measurements.

*y* : *array\_like, optional*

The second set of measurements. If *y* is not given, then the *x* array is considered to be the differences between the two sets of measurements.

**zero\_method** : *string*, {"pratt", "wilcox", "zsplit"}, *optional*

**"pratt":**

Pratt treatment: includes zero-differences in the ranking process (more conservative)

**"wilcox":**

Wilcox treatment: discards all zero-differences

**"zsplit":**

Zero rank split: just like Pratt, but splitting the zero rank between positive and negative ones

**correction** : *bool, optional*

If True, apply continuity correction by adjusting the Wilcoxon rank statistic by 0.5 towards the mean value when computing the z-statistic.  
Default is False.

**Returns:**

**T** : *float*

The sum of the ranks of the differences above or below zero, whichever is smaller.

**p-value** : *float*

The two-sided p-value for the test.

### Notes

Because the normal approximation is used for the calculations, the samples used should be large. A typical rule is to require that  $n > 20$ .

# Mann-Whitney rank test

## scipy.stats.mannwhitneyu

`scipy.stats.mannwhitneyu`(*x*, *y*, *use\_continuity*=True, *alternative*=None)

[\[source\]](#)

Computes the Mann-Whitney rank test on samples *x* and *y*.

**Parameters:** *x*, *y* : *array\_like*

Array of samples, should be one-dimensional.

**use\_continuity** : *bool, optional*

Whether a continuity correction (1/2.) should be taken into account. Default is True.

**alternative** : *None (deprecated), 'less', 'two-sided', or 'greater'*

Whether to get the p-value for the one-sided hypothesis ('less' or 'greater') or for the two-sided hypothesis ('two-sided'). Defaults to None, which results in a p-value half the size of the 'two-sided' p-value and a different U statistic. The default behavior is not the same as using 'less' or 'greater': it only exists for backward compatibility and is deprecated.

**Returns:**

**statistic** : *float*

The Mann-Whitney U statistic, equal to min(U for *x*, U for *y*) if *alternative* is equal to None (deprecated; exists for backward compatibility), and U for *y* otherwise.

**pvalue** : *float*

p-value assuming an asymptotic normal distribution. One-sided or two-sided, depending on the choice of *alternative*.

### Notes

Use only when the number of observation in each sample is > 20 and you have 2 independent samples of ranks. Mann-Whitney U is significant if the u-obtained is LESS THAN or equal to the critical value of U.

This test corrects for ties and by default uses a continuity correction.

<https://docs.scipy.org>



# Chi square test

## scipy.stats.chisquare

`scipy.stats.chisquare` (*f\_obs*, *f\_exp=None*, *ddof=0*, *axis=0*)

[\[source\]](#)

Calculates a one-way chi square test.

The chi square test tests the null hypothesis that the categorical data has the given frequencies.

**Parameters:** *f\_obs* : array\_like

Observed frequencies in each category.

*f\_exp* : array\_like, optional

Expected frequencies in each category. By default the categories are assumed to be equally likely.

*ddof* : int, optional

“Delta degrees of freedom”: adjustment to the degrees of freedom for the p-value. The p-value is computed using a chi-squared distribution with  $k - 1 - \text{ddof}$  degrees of freedom, where  $k$  is the number of observed frequencies. The default value of *ddof* is 0.

*axis* : int or None, optional

The axis of the broadcast result of *f\_obs* and *f\_exp* along which to apply the test. If *axis* is None, all values in *f\_obs* are treated as a single data set. Default is 0.

**Returns:** *chisq* : float or ndarray

The chi-squared test statistic. The value is a float if *axis* is None or *f\_obs* and *f\_exp* are 1-D.

*p* : float or ndarray

The p-value of the test. The value is a float if *ddof* and the return value *chisq* are scalars.

### See also:

`power_divergence`, `mstats.chisquare`

### Notes

This test is invalid when the observed or expected frequencies in each category are too small. A typical rule is that all of the observed and expected frequencies should be at least 5.

The default degrees of freedom,  $k-1$ , are for the case when no parameters of the distribution are estimated. If  $p$  parameters are estimated by efficient maximum likelihood then the correct degrees of freedom are  $k-1-p$ . If the parameters are estimated in a different way, then the dof can be between  $k-1-p$  and  $k-1$ .

However, it is also possible that the asymptotic distribution is not a chisquare, in which case this test is not appropriate.

<https://docs.scipy.org>

# Human dataset - Exercise 3

- Input: `expression_human_chr1_short.txt`
- Calculate the mean expression for each gene! (e.g. use numpy)
- Which gene has the highest expression value?
- Visualize the expression data!

# Human dataset - Exercise 4

- Input: genes\_human\_chr1.txt
- Which gene is the longest human gene on chr1?
- Calculate the mean gene length of all genes on chr1.
- Visualize the gene length distribution.

# Human dataset - Exercise 5

- Are longer genes stronger expressed than shorter genes?
- Statistical analysis