scipy.stats.ttest_ind

`scipy.stats.ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate', permutations=None, random_state=None, alternative='two-sided', trim=0, *, keepdims=False)`

Calculate the T-test for the means of *two independent* samples of scores. [source]

This is a 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, default: 0*

If an int, the axis of the input along which to compute the statistic. The statistic of each axis-slice (e.g. row) of the input will appear in a corresponding element of the output. If `None`, the input will be raveled before computing the statistic.

equal_var : *bool, optional*

If True (default), perform a standard independent 2 sample test that assumes equal population variances [1]. If False, perform Welch's t-test, which does not assume equal population variance [2].

> ***New in version 0.11.0.***

nan_policy : *{'propagate', 'omit', 'raise'}*

Defines how to handle input NaNs.

- `propagate`: if a NaN is present in the axis slice (e.g. row) along which the statistic is computed, the corresponding entry of the output will be NaN.
- `omit`: NaNs will be omitted when performing the calculation. If insufficient data remains in the axis slice along which the statistic is computed, the corresponding entry of the output will be NaN.
- `raise`: if a NaN is present, a `ValueError` will be raised.

permutations : *non-negative int, np.inf, or None (default), optional*

If 0 or None (default), use the t-distribution to calculate p-values. Otherwise, *permutations* is the number of random permutations that will be used to estimate p-values using a permutation test. If *permutations* equals or exceeds the number of distinct partitions of the pooled data, an exact test is performed instead (i.e. each distinct partition is used exactly once). See Notes for details.

> ***New in version 1.7.0.***

random_state : *{None, int, `numpy.random.Generator`, `numpy.random.RandomState`}, optional*

If *seed* is None (or *np.random*), the `numpy.random.RandomState` singleton is used. If *seed* is an int, a new `RandomState` instance is used, seeded with *seed*. If *seed* is already a `Generator` or `RandomState` instance then that instance is used.
Pseudorandom number generator state used to generate permutations (used only when *permutations* is not None).

> ***New in version 1.7.0.***

alternative : *{'two-sided', 'less', 'greater'}, optional*

Defines the alternative hypothesis. The following options are available (default is 'two-sided'):

- 'two-sided': the means of the distributions underlying the samples are unequal.
- 'less': the mean of the distribution underlying the first sample is less than the mean of the distribution underlying the second sample.
- 'greater': the mean of the distribution underlying the first sample is greater than the mean of the distribution underlying the second sample.

> ***New in version 1.6.0.***

trim : *float, optional*

If nonzero, performs a trimmed (Yuen's) t-test. Defines the fraction of elements to be trimmed from each end of the input samples. If 0 (default), no elements will be trimmed from either side. The number of trimmed elements from each tail is the floor of the trim times the number of elements. Valid range is [0, .5).

> ***New in version 1.7.***

keepdims : *bool, default: False*

If this is set to True, the axes which are reduced are left in the result as dimensions with size one. With this option, the result will broadcast correctly against the input array.

Returns: **result** : *TtestResult*

An object with the following attributes:

statistic : *float or ndarray*

The t-statistic.

pvalue : *float or ndarray*

The p-value associated with the given alternative.

df : *float or ndarray*

The number of degrees of freedom used in calculation of the t-statistic. This is always NaN for a permutation t-test.

> ***New in version 1.11.0.***

The object also has the following method:

confidence_interval(confidence_level=0.95)

Computes a confidence interval around the difference in population means for the given confidence level. The confidence interval is returned in a `namedtuple` with fields `low` and `high`. When a permutation t-test is performed, the confidence interval is not computed, and fields `low` and `high` contain NaN.

> ***New in version 1.11.0.***

Notes

Suppose we observe two independent samples, e.g. flower petal lengths, and we are considering whether the two samples were drawn from the same population (e.g. the same species of flower or two species with similar petal characteristics) or two different populations.

The t-test quantifies the difference between the arithmetic means of the two samples. The p-value quantifies the probability of observing as or more extreme values assuming the null hypothesis, that the samples are drawn from populations with the same population means, is true. A p-value larger than a chosen threshold (e.g. 5% or 1%) indicates that our observation is not so unlikely to have occurred by chance. Therefore, we do not reject the null hypothesis of equal population means. If the p-value is smaller than our threshold, then we have evidence against the null hypothesis of equal population means.

By default, the p-value is determined by comparing the t-statistic of the observed data against a theoretical t-distribution. When `1 < permutations < binom(n, k)`, where

- `k` is the number of observations in *a*,
- `n` is the total number of observations in *a* and *b*, and
- `binom(n, k)` is the binomial coefficient (`n` choose `k`),

the data are pooled (concatenated), randomly assigned to either group *a* or *b*, and the t-statistic is calculated. This process is performed repeatedly (*permutation* times), generating a distribution of the t-statistic under the null hypothesis, and the t-statistic of the observed data is compared to this distribution to determine the p-value. Specifically, the p-value reported is the "achieved significance level" (ASL) as defined in 4.4 of [3]. Note that there are other ways of estimating p-values using randomized permutation tests; for other options, see the more general **`permutation_test`**.

When `permutations >= binom(n, k)`, an exact test is performed: the data are partitioned between the groups in each distinct way exactly once.

The permutation test can be computationally expensive and not necessarily more accurate than the analytical test, but it does not make strong assumptions about the shape of the underlying distribution.

Use of trimming is commonly referred to as the trimmed t-test. At times called Yuen's t-test, this is an extension of Welch's t-test, with the difference being the use of winsorized means in calculation of the variance and the trimmed sample size in calculation of the statistic. Trimming is recommended if the underlying distribution is long-tailed or contaminated with outliers [4].

The statistic is calculated as `(np.mean(a) - np.mean(b))/se`, where `se` is the standard error. Therefore, the statistic will be positive when the sample mean of *a* is greater than the sample mean of *b* and negative when the sample mean of *a* is less than the sample mean of *b*.

Beginning in SciPy 1.9, `np.matrix` inputs (not recommended for new code) are converted to `np.ndarray` before the calculation is performed. In this case, the output will be a scalar or `np.ndarray` of appropriate shape rather than a 2D `np.matrix`. Similarly, while masked elements of masked arrays are ignored, the output will be a scalar or `np.ndarray` rather than a masked array with `mask=False`.

References

[1] https://en.wikipedia.org/wiki/T-test#Independent_two-sample_t-test

[2] https://en.wikipedia.org/wiki/Welch%27s_t-test

[3] B. Efron and T. Hastie. Computer Age Statistical Inference. (2016).

[4] Yuen, Karen K. "The Two-Sample Trimmed t for Unequal Population Variances." Biometrika, vol. 61, no. 1, 1974, pp. 165-170. JSTOR, www.jstor.org/stable/2334299. Accessed 30 Mar. 2021.

[5] Yuen, Karen K., and W. J. Dixon. "The Approximate Behaviour and Performance of the Two-Sample Trimmed t." Biometrika, vol. 60, no. 2, 1973, pp. 369-374. JSTOR, www.jstor.org/stable/2334550. Accessed 30 Mar. 2021.

Examples

```
>>> import numpy as np
>>> from scipy import stats
>>> rng = np.random.default_rng()
```

Test with sample with identical means:

```
>>> rvs1 = stats.norm.rvs(loc=5, scale=10, size=500, random_state=rng)
>>> rvs2 = stats.norm.rvs(loc=5, scale=10, size=500, random_state=rng)
>>> stats.ttest_ind(rvs1, rvs2)
Ttest_indResult(statistic=-0.4390847099199348, pvalue=0.6606952038870015)
>>> stats.ttest_ind(rvs1, rvs2, equal_var=False)
Ttest_indResult(statistic=-0.4390847099199348, pvalue=0.6606952553131064)
```

`ttest_ind` underestimates p for unequal variances:

```
>>> rvs3 = stats.norm.rvs(loc=5, scale=20, size=500, random_state=rng)
>>> stats.ttest_ind(rvs1, rvs3)
Ttest_indResult(statistic=-1.6370984482905417, pvalue=0.1019251574705033)
>>> stats.ttest_ind(rvs1, rvs3, equal_var=False)
Ttest_indResult(statistic=-1.637098448290542, pvalue=0.10202110497954867)
```

When `n1 != n2`, the equal variance t-statistic is no longer equal to the unequal variance t-statistic:

```
>>> rvs4 = stats.norm.rvs(loc=5, scale=20, size=100, random_state=rng)
>>> stats.ttest_ind(rvs1, rvs4)
Ttest_indResult(statistic=-1.9481646859513422, pvalue=0.05186270935842703)
>>> stats.ttest_ind(rvs1, rvs4, equal_var=False)
Ttest_indResult(statistic=-1.3146566100751664, pvalue=0.1913495266513811)
```

T-test with different means, variance, and n:

```
>>> rvs5 = stats.norm.rvs(loc=8, scale=20, size=100, random_state=rng)
>>> stats.ttest_ind(rvs1, rvs5)
Ttest_indResult(statistic=-2.8415950600298774, pvalue=0.0046418707568707885)
>>> stats.ttest_ind(rvs1, rvs5, equal_var=False)
Ttest_indResult(statistic=-1.8686598649188084, pvalue=0.06434714193919686)
```

When performing a permutation test, more permutations typically yields more accurate results. Use a `np.random.Generator` to ensure reproducibility:

```
>>> stats.ttest_ind(rvs1, rvs5, permutations=10000,
...                 random_state=rng)
Ttest_indResult(statistic=-2.8415950600298774, pvalue=0.0052994700529947)
```

Take these two samples, one of which has an extreme tail.

```
>>> a = (56, 128.6, 12, 123.8, 64.34, 78, 763.3)
>>> b = (1.1, 2.9, 4.2)
```

Use the *trim* keyword to perform a trimmed (Yuen) t-test. For example, using 20% trimming, `trim=.2`, the test will reduce the impact of one (`np.floor(trim*len(a))`) element from each tail of sample *a*. It will have no effect on sample *b* because `np.floor(trim*len(b))` is 0.

```
>>> stats.ttest_ind(a, b, trim=.2)
Ttest_indResult(statistic=3.4463884028073513,
                pvalue=0.01369338726499547)
```

Previous
scipy.stats.poisson_means_test

Next
scipy.stats.mannwhitneyu