**Student’s t-Test from Scratch in Python**

1. Student’s t-Test
2. Student’s t-Test for Independent Samples
3. Student’s t-Test for Dependent Samples

## Student’s t-Test

The Student’s t-Test is a statistical hypothesis test for testing whether two samples are expected to have been drawn from the same population.

The test works by **checking the means** from two samples to see if they are significantly different from each other. It does this by **calculating the standard error** in the difference between means, which can be interpreted to see how likely the difference is, if the two samples have the **same mean** (the null hypothesis).

The **t statistic** calculated by the test can be interpreted by comparing it to **critical values** from the t-distribution. The critical value can be calculated using the **degrees of freedom** and a significance level with the **percent point function (PPF).**

We can interpret the statistic value in a **two-tailed test**, meaning that if we reject the null hypothesis, it could be because the first mean is smaller or greater than the second mean. To do this, we can calculate the **absolute value** of the test statistic and compare it to the **positive (right tailed) critical value**, as follows:

* **If abs(t-statistic) <= critical value**: Accept null hypothesis that the means are equal.
* **If abs(t-statistic) > critical value**: Reject the null hypothesis that the means are equal.

**Another Way To accept or reject the Null Hypothesis**

We can also retrieve the cumulative probability of observing the absolute value of the t-statistic using the **cumulative distribution function (CDF)** of the t-distribution in order to calculate a **p-value**. The p-value can then be compared to a chosen s**ignificance level** (**alpha**) such as **0.05** to determine if the null hypothesis can be rejected:

* **If p > alpha (0.05)**: Accept null hypothesis that the means are equal.
* **If p <= alpha (0.05)**: Reject null hypothesis that the means are equal.

**Assumptions for T-Test**

In working with the means of the samples, the test assumes that both samples were drawn from a **Gaussian distribution**. The test also assumes that the samples have the **same variance**, and the **same size**, although there are corrections to the test if these assumptions do not hold. For example, see Welch’s t-test.

There are two main versions of Student’s t-test:

* **Independent Samples**. The case where the two samples are unrelated.
* **Dependent Samples**. The case where the samples are related, such as repeated measures on the same population. Also called a paired test.

Both the independent and the dependent Student’s t-tests are available in Python via the **ttest\_ind()** and [**ttest\_rel()**](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html) SciPy functions respectively.

## Student’s t-Test for Independent Samples

### Calculation

The calculation of the t-statistic for two independent samples is as follows:

t = observed difference between sample means / standard error of the difference between the means

or

t = (mean(X1) - mean(X2)) / sed

Where *X1* and *X2* are the first and second data samples and *sed* is the standard error of the difference between the means.

The standard error of the difference between the means can be calculated as follows:

sed = sqrt(se1^2 + se2^2)

Where *se1* and *se2* are the standard errors for the first and second datasets.

The standard error of a sample can be calculated as:

se = std / sqrt(n)

Where se is the standard error of the sample, std is the sample standard deviation, and n is the number of observations in the sample.

These calculations make the following assumptions:

* The samples are drawn from a Gaussian distribution.
* The size of each sample is approximately equal.
* The samples have the same variance.

### Implementation

using functions from the Python standard library, NumPy and SciPy

Let’s assume that our two data samples are stored in the variables data1 and data2.

We can start off by calculating the mean for these samples as follows:

|  |  |
| --- | --- |
|  | # calculate means  mean1, mean2 = mean(data1), mean(data2) |

We’re halfway there.

Now we need to calculate the standard error.

We can do this manually, first by calculating the sample standard deviations:

|  |  |
| --- | --- |
|  | # calculate sample standard deviations  std1, std2 = std(data1, ddof=1), std(data2, ddof=1) |

And then the standard errors:

|  |  |
| --- | --- |
|  | # calculate standard errors  n1, n2 = len(data1), len(data2)  se1, se2 = std1/sqrt(n1), std2/sqrt(n2) |

Alternately, we can use the sem() SciPy function to calculate the standard error directly.

|  |  |
| --- | --- |
|  | # calculate standard errors  se1, se2 = sem(data1), sem(data2) |

We can use the standard errors of the samples to calculate the “standard error on the difference between the samples“:

|  |  |
| --- | --- |
|  | # standard error on the difference between the samples  sed = sqrt(se1\*\*2.0 + se2\*\*2.0) |

We can now calculate the t statistic:

|  |  |
| --- | --- |
|  | # calculate the t statistic  t\_stat = (mean1 - mean2) / sed |

We can also calculate some other values to help interpret and present the statistic.

The number of degrees of freedom for the test is calculated as the sum of the observations in both samples, minus two.

|  |  |
| --- | --- |
|  | # degrees of freedom  df = n1 + n2 - 2 |

The critical value can be calculated using the percent point function (PPF) for a given significance level, such as 0.05 (95% confidence).

This function is available for the t distribution in SciPy, as follows:

|  |  |
| --- | --- |
|  | # calculate the critical value  alpha = 0.05  cv = t.ppf(1.0 - alpha, df) |

The p-value can be calculated using the cumulative distribution function on the t-distribution, again in SciPy.

|  |  |
| --- | --- |
|  | # calculate the p-value  p = (1 - t.cdf(abs(t\_stat), df)) \* 2 |

Here, we assume a two-tailed distribution, where the rejection of the null hypothesis could be interpreted as the first mean is either smaller or larger than the second mean.

We can tie all of these pieces together into a simple function for calculating the t-test for two independent samples:

|  |  |
| --- | --- |
|  | # function for calculating the t-test for two independent samples  def independent\_ttest(data1, data2, alpha):  # calculate means  mean1, mean2 = mean(data1), mean(data2)  # calculate standard errors  se1, se2 = sem(data1), sem(data2)  # standard error on the difference between the samples  sed = sqrt(se1\*\*2.0 + se2\*\*2.0)  # calculate the t statistic  t\_stat = (mean1 - mean2) / sed  # degrees of freedom  df = len(data1) + len(data2) - 2  # calculate the critical value  cv = t.ppf(1.0 - alpha, df)  # calculate the p-value  p = (1.0 - t.cdf(abs(t\_stat), df)) \* 2.0  # return everything  return t\_stat, df, cv, p |

### Worked Example

First, let’s generate two samples of 100 Gaussian random numbers with the same variance of 5 and differing means of 50 and 51 respectively. We will expect the test to reject the null hypothesis and find a significant difference between the samples:

hypothesis and find a significant difference between the samples:

|  |  |
| --- | --- |
|  | # seed the random number generator  seed(1)  # generate two independent samples  data1 = 5 \* randn(100) + 50  data2 = 5 \* randn(100) + 51 |

We can calculate the t-test on these samples using the built in SciPy function *ttest\_ind()*. This will give us a t-statistic value and a p-value to compare to, to ensure that we have implemented the test correctly.

The complete example is listed below.

|  |  |
| --- | --- |
|  | # Student's t-test for independent samples  from numpy.random import seed  from numpy.random import randn  from scipy.stats import ttest\_ind  # seed the random number generator  seed(1)  # generate two independent samples  data1 = 5 \* randn(100) + 50  data2 = 5 \* randn(100) + 51  # compare samples  stat, p = ttest\_ind(data1, data2)  print('t=%.3f, p=%.3f' % (stat, p)) |

Running the example, we can see a t-statistic value and p value.

We will use these as our expected values for the test on these data.

|  |  |
| --- | --- |
| 1 | t=-2.262, p=0.025 |

We can now apply our own implementation on the same data, using the function defined in the previous section.

The function will return a t-statistic value and a critical value. We can use the critical value to interpret the t statistic to see if the finding of the test is significant and that indeed the means are different as we expected.

|  |  |
| --- | --- |
|  | # interpret via critical value  if abs(t\_stat) <= cv:  print('Accept null hypothesis that the means are equal.')  else:  print('Reject the null hypothesis that the means are equal.') |

The function also returns a p-value. We can interpret the p-value using an alpha, such as 0.05 to determine if the finding of the test is significant and that indeed the means are different as we expected.

|  |  |
| --- | --- |
|  | # interpret via p-value  if p > alpha:  print('Accept null hypothesis that the means are equal.')  else:  print('Reject the null hypothesis that the means are equal.') |

We expect that both interpretations will always match.

The complete example is listed below.

|  |  |
| --- | --- |
|  | # t-test for independent samples  from math import sqrt  from numpy.random import seed  from numpy.random import randn  from numpy import mean  from scipy.stats import sem  from scipy.stats import t    # function for calculating the t-test for two independent samples  def independent\_ttest(data1, data2, alpha):  # calculate means  mean1, mean2 = mean(data1), mean(data2)  # calculate standard errors  se1, se2 = sem(data1), sem(data2)  # standard error on the difference between the samples  sed = sqrt(se1\*\*2.0 + se2\*\*2.0)  # calculate the t statistic  t\_stat = (mean1 - mean2) / sed  # degrees of freedom  df = len(data1) + len(data2) - 2  # calculate the critical value  cv = t.ppf(1.0 - alpha, df)  # calculate the p-value  p = (1.0 - t.cdf(abs(t\_stat), df)) \* 2.0  # return everything  return t\_stat, df, cv, p    # seed the random number generator  seed(1)  # generate two independent samples  data1 = 5 \* randn(100) + 50  data2 = 5 \* randn(100) + 51  # calculate the t test  alpha = 0.05  t\_stat, df, cv, p = independent\_ttest(data1, data2, alpha)  print('t=%.3f, df=%d, cv=%.3f, p=%.3f' % (t\_stat, df, cv, p))  # interpret via critical value  if abs(t\_stat) <= cv:  print('Accept null hypothesis that the means are equal.')  else:  print('Reject the null hypothesis that the means are equal.')  # interpret via p-value  if p > alpha:  print('Accept null hypothesis that the means are equal.')  else:  print('Reject the null hypothesis that the means are equal.') |

Running the example first calculates the test.

|  |  |
| --- | --- |
|  | t=-2.262, df=198, cv=1.653, p=0.025  Reject the null hypothesis that the means are equal.  Reject the null hypothesis that the means are equal. |

## Student’s t-Test for Dependent Samples

look at the case of calculating the Student’s t-test for dependent samples.

This is the case where we collect some observations on a sample from the population, then apply some treatment, and then collect observations from the same sample.

The result is two samples of the same size where the observations in each sample are related or paired.

The t-test for dependent samples is referred to as the paired Student’s t-test.

### Calculation

The calculation of the paired Student’s t-test is similar to the case with independent samples.

The main difference is in the calculation of the denominator.

|  |  |
| --- | --- |
|  | t = (mean(X1) - mean(X2)) / sed |

Where X1 and X2 are the first and second data samples and sed is the standard error of the difference between the means.

Here, sed is calculated as:

|  |  |
| --- | --- |
|  | sed = sd / sqrt(n) |

Where sd is the standard deviation of the difference between the dependent sample means and n is the total number of paired observations (e.g. the size of each sample).

The calculation of sd first requires the calculation of the sum of the squared differences between the samples:

|  |  |
| --- | --- |
|  | d1 = sum (X1[i] - X2[i])^2 for i in n |

It also requires the sum of the (non squared) differences between the samples:

|  |  |
| --- | --- |
|  | d2 = sum (X1[i] - X2[i]) for i in n |

We can then calculate sd as:

|  |  |
| --- | --- |
|  | sd = sqrt((d1 - (d2\*\*2 / n)) / (n - 1)) |

That’s it.

### Implementation

We can implement the calculation of the paired Student’s t-test directly in Python.

The first step is to calculate the means of each sample.

|  |  |
| --- | --- |
|  | # calculate means  mean1, mean2 = mean(data1), mean(data2) |

Next, we will require the number of pairs (n). We will use this in a few different calculations.

|  |  |
| --- | --- |
|  | # number of paired samples  n = len(data1) |

Next, we must calculate the sum of the squared differences between the samples, as well as the sum differences.

|  |  |
| --- | --- |
|  | # sum squared difference between observations  d1 = sum([(data1[i]-data2[i])\*\*2 for i in range(n)])  # sum difference between observations  d2 = sum([data1[i]-data2[i] for i in range(n)]) |

We can now calculate the standard deviation of the difference between means.

|  |  |
| --- | --- |
|  | # standard deviation of the difference between means  sd = sqrt((d1 - (d2\*\*2 / n)) / (n - 1)) |

This is then used to calculate the standard error of the difference between the means.

|  |  |
| --- | --- |
|  | # standard error of the difference between the means  sed = sd / sqrt(n) |

Finally, we have everything we need to calculate the t statistic.

|  |  |
| --- | --- |
|  | # calculate the t statistic  t\_stat = (mean1 - mean2) / sed |

The only other key difference between this implementation and the implementation for independent samples is the calculation of the number of degrees of freedom.

|  |  |
| --- | --- |
|  | # degrees of freedom  df = n - 1 |

As before, we can tie all of this together into a reusable function. The function will take two paired samples and a significance level (alpha) and calculate the t-statistic, number of degrees of freedom, critical value, and p-value.

The complete function is listed below.

|  |  |
| --- | --- |
|  | # function for calculating the t-test for two dependent samples  def dependent\_ttest(data1, data2, alpha):  # calculate means  mean1, mean2 = mean(data1), mean(data2)  # number of paired samples  n = len(data1)  # sum squared difference between observations  d1 = sum([(data1[i]-data2[i])\*\*2 for i in range(n)])  # sum difference between observations  d2 = sum([data1[i]-data2[i] for i in range(n)])  # standard deviation of the difference between means  sd = sqrt((d1 - (d2\*\*2 / n)) / (n - 1))  # standard error of the difference between the means  sed = sd / sqrt(n)  # calculate the t statistic  t\_stat = (mean1 - mean2) / sed  # degrees of freedom  df = n - 1  # calculate the critical value  cv = t.ppf(1.0 - alpha, df)  # calculate the p-value  p = (1.0 - t.cdf(abs(t\_stat), df)) \* 2.0  # return everything  return t\_stat, df, cv, p |

### Worked Example

In this section, we will use the same dataset in the worked example as we did for the independent Student’s t-test.

The data samples are not paired, but we will pretend they are. We expect the test to reject the null hypothesis and find a significant difference between the samples.

|  |  |
| --- | --- |
|  | # seed the random number generator  seed(1)  # generate two independent samples  data1 = 5 \* randn(100) + 50  data2 = 5 \* randn(100) + 51 |

As before, we can evaluate the test problem with the SciPy function for calculating a paired t-test. In this case, the ttest\_rel() function.

The complete example is listed below.

|  |  |
| --- | --- |
|  | # Paired Student's t-test  from numpy.random import seed  from numpy.random import randn  from scipy.stats import ttest\_rel  # seed the random number generator  seed(1)  # generate two independent samples  data1 = 5 \* randn(100) + 50  data2 = 5 \* randn(100) + 51  # compare samples  stat, p = ttest\_rel(data1, data2)  print('Statistics=%.3f, p=%.3f' % (stat, p)) |

Running the example calculates and prints the t-statistic and the p-value.

We will use these values to validate the calculation of our own paired t-test function.

|  |  |
| --- | --- |
| 1 | Statistics=-2.372, p=0.020 |

We can now test our own implementation of the paired Student’s t-test.

The complete example, including the developed function and interpretation of the results of the function, is listed below.

|  |  |
| --- | --- |
|  | # t-test for dependent samples  from math import sqrt  from numpy.random import seed  from numpy.random import randn  from numpy import mean  from scipy.stats import t    # function for calculating the t-test for two dependent samples  def dependent\_ttest(data1, data2, alpha):  # calculate means  mean1, mean2 = mean(data1), mean(data2)  # number of paired samples  n = len(data1)  # sum squared difference between observations  d1 = sum([(data1[i]-data2[i])\*\*2 for i in range(n)])  # sum difference between observations  d2 = sum([data1[i]-data2[i] for i in range(n)])  # standard deviation of the difference between means  sd = sqrt((d1 - (d2\*\*2 / n)) / (n - 1))  # standard error of the difference between the means  sed = sd / sqrt(n)  # calculate the t statistic  t\_stat = (mean1 - mean2) / sed  # degrees of freedom  df = n - 1  # calculate the critical value  cv = t.ppf(1.0 - alpha, df)  # calculate the p-value  p = (1.0 - t.cdf(abs(t\_stat), df)) \* 2.0  # return everything  return t\_stat, df, cv, p    # seed the random number generator  seed(1)  # generate two independent samples (pretend they are dependent)  data1 = 5 \* randn(100) + 50  data2 = 5 \* randn(100) + 51  # calculate the t test  alpha = 0.05  t\_stat, df, cv, p = dependent\_ttest(data1, data2, alpha)  print('t=%.3f, df=%d, cv=%.3f, p=%.3f' % (t\_stat, df, cv, p))  # interpret via critical value  if abs(t\_stat) <= cv:  print('Accept null hypothesis that the means are equal.')  else:  print('Reject the null hypothesis that the means are equal.')  # interpret via p-value  if p > alpha:  print('Accept null hypothesis that the means are equal.')  else:  print('Reject the null hypothesis that the means are equal.') |

Running the example calculates the paired t-test on the sample problem.

The calculated t-statistic and p-value match what we expect from the SciPy library implementation. This suggests that the implementation is correct.

The interpretation of the t-test statistic with the critical value, and the p-value with the significance level both find a significant result, rejecting the null hypothesis that the means are equal.

|  |  |
| --- | --- |
|  | t=-2.372, df=99, cv=1.660, p=0.020  Reject the null hypothesis that the means are equal.  Reject the null hypothesis that the means are equal. |