How to Easily Perform Hypothesis Testing in Python with SciPy

Authored by

stats writer

November 28, 2025

Hypothesis testing is a fundamental statistical procedure used to determine whether there is enough evidence in a sample data set to infer that a certain condition or assumption about a population is true. In the world of data science and analysis, performing these tests efficiently requires powerful tools. Fortunately, Python, combined with its robust scientific ecosystem, provides excellent capabilities for conducting rigorous statistical analysis.

Specifically, the SciPy library, particularly the `scipy.stats` module, offers built-in functions designed for calculating critical values and performing tests such such as z-tests, chi-squared tests, and the various forms of the t-test. These functions abstract away complex mathematical derivations, allowing data scientists to focus on interpreting the results, drawing sound conclusions, and making data-driven decisions. Understanding how to apply these functions correctly is essential for any statistical modeling task.

Before diving into the code, it is important to clarify what a statistical test aims to achieve. A hypothesis testing procedure involves setting up two opposing statements about a population parameter: the null hypothesis (H0) and the alternative hypothesis (HA). The test evaluates the likelihood of observing the sample data if the null hypothesis were true.

The outcome of a statistical test typically yields a test statistic and a corresponding p-value. The p-value measures the probability of observing data as extreme as, or more extreme than, the actual data, assuming H0 is true. By comparing the p-value to a predetermined significance level (alpha, typically 0.05), we can decide whether to reject H0 or fail to reject H0. A p-value less than alpha suggests strong evidence against the null hypothesis.

This tutorial will walk through three common types of t-tests, demonstrating their implementation using Python's SciPy library:

One-Sample t-test: Comparing a sample mean to a known population value.
Two-Sample t-test: Comparing the means of two independent groups.
Paired Samples t-test: Comparing the means of two related samples.

Example 1: Performing a One-Sample t-test in Python

A one-sample t-test is employed when the goal is to determine if the mean of a single population differs significantly from a hypothesized value or standard. This test is crucial in quality control, calibration checks, and fundamental research where a baseline or benchmark is established. The assumptions for this test generally include that the data is approximately normally distributed and that the observations are independent.

Consider a scenario where researchers are studying a specific species of turtle and hypothesize that the average weight of the population is 310 pounds. To test this claim, a simple random

sample of turtles is collected, and their weights are recorded. We must determine if the sample evidence contradicts the hypothesized population mean (310 pounds).

The collected sample weights are as follows:
Weights: 300, 315, 320, 311, 314, 309, 300, 308, 305, 303, 305, 301, 303.

We utilize the `ttest_1samp()` function from the `scipy.stats` module to execute the one-sample t-test. This function requires the sample data (`a`) and the hypothesized population mean (`popmean`) as inputs.

import scipy.stats as stats

#define data
data =

#perform one sample t-test
stats.ttest_1samp(a=data, popmean=310)

Ttest_1sampResult(statistic=-1.5848116313861254, pvalue=0.1389944275158753)

Interpreting the Results of the One-Sample t-test

The code execution yields a t-test statistic of **-1.5848** and a corresponding two-sided p-value of **0.1389**. These two values are central to drawing a statistical conclusion regarding the turtle weights. The specific hypotheses being tested are formally defined as follows:

H0: μ = 310 (The mean weight for this species of turtle is equal to 310 pounds)
HA: μ ? 310 (The mean weight is significantly different from 310 pounds)

We typically use a standard significance level (alpha) of 0.05. Since the calculated p-value (0.1389) is greater than our chosen alpha (0.05), we must fail to reject the null hypothesis. This finding suggests that, based on our sample, we do not have sufficient statistical evidence to conclude that the true mean weight for this particular species of turtle is different from 310 pounds.

Example 2: Executing a Two-Sample t-test in Python

The two-sample t-test, also known as the independent samples t-test, is used to determine if the means of two independent populations are equal. This test is highly utilized in comparative studies, such as assessing the performance difference between two distinct groups, products, or treatments. It assumes that the data in both samples are independently drawn and approximately normally distributed.

Let's extend our previous turtle example. Suppose we now want to compare the mean weight of the first species (Sample 1) against a second, different species of turtle (Sample 2). The research question is whether there is a statistically significant difference in mean weights between these two distinct populations.

We collect simple random samples from both populations, yielding the following weights:

Sample 1: 300, 315, 320, 311, 314, 309, 300, 308, 305, 303, 305, 301, 303

Sample 2: 335, 329, 322, 321, 324, 319, 304, 308, 305, 311, 307, 300, 305

To perform this comparison in Python, we use the `ttest_ind()` function from `scipy.stats`. The `_ind` suffix stands for independent samples.

import scipy.stats as stats

#define array of turtle weights for each sample
sample1 =
sample2 =

#perform two sample t-test
stats.ttest_ind(a=sample1, b=sample2)

Ttest_indResult(statistic=-2.1009029257555696, pvalue=0.04633501389516516)

Analyzing the Two-Sample t-test Outcome

The resulting output provides a t-test statistic of **-2.1009** and a two-sided p-value of **0.0463**. These results allow us to evaluate the following hypotheses regarding the population means (μ1 and μ2):

H0: μ1 = μ2 (The mean weights between the two species are equal.)
HA: μ1 ? μ2 (The mean weights between the two species are not equal.)

Comparing the calculated p-value (0.0463) to the standard significance level of alpha = 0.05, we observe that 0.0463 is less than 0.05. Therefore, we reject the null hypothesis. This rejection indicates that we have statistically sufficient evidence to conclude that there is a significant difference in the mean weight between the two species of turtles.

Example 3: Conducting a Paired Samples t-test in Python

A paired samples t-test is structurally distinct from the independent two-sample test because it deals with dependent samples. This test is used when each observation in one sample is directly

matched or "paired" with an observation in the other sample. Common applications include "before-and-after" studies, where the same subjects are measured under two different conditions, thus eliminating inter-subject variability.

For example, suppose we want to know whether or not a certain training program is able to increase the max vertical jump (in inches) of basketball players. To test this, we may recruit a simple random sample of 12 college basketball players and measure each of their max vertical jumps. Then, we may have each player use the training program for one month and then measure their max vertical jump again at the end of the month.

The following data shows the max jump height (in inches) before and after using the training program for each player:

Before: 22, 24, 20, 19, 19, 20, 22, 25, 24, 23, 22, 21

After: 23, 25, 20, 24, 18, 22, 23, 28, 24, 25, 24, 20

To analyze this dependent data, we use the `ttest_rel()` function from the `scipy.stats` library. The `_rel` suffix signifies a related or paired comparison.

import scipy.stats as stats

#define before and after max jump heights
before =
after =

#perform paired samples t-test
stats.ttest_rel(a=before, b=after)

Ttest_relResult(statistic=-2.5289026942943655, pvalue=0.02802807458682508)

Interpreting the Paired Samples t-test Results

The paired samples t-test results in a t-test statistic of **-2.5289** and a two-sided p-value of **0.0280**. We are testing the following hypotheses:

H0: μ1 = μ2 (The mean jump height before and after using the program is equal, meaning the program had no effect.)
HA: μ1 ? μ2 (The mean jump height before and after using the program is not equal, meaning the program caused a change.)

Since the calculated p-value (0.0280) is less than the typical significance level of 0.05, we possess

sufficient statistical evidence to reject the null hypothesis. Consequently, we conclude that the mean vertical jump height before and after using the training program is statistically unequal, implying that the training program had a significant effect on the players' jump performance.

Conclusion and Further Resources

Python's SciPy library provides a streamlined and powerful interface for executing crucial statistical analyses like hypothesis testing. By correctly identifying whether you require a one-sample, two-sample independent, or paired-sample t-test, you can apply the appropriate function (`ttest_1samp()`, `ttest_ind()`, or `ttest_rel()`) and quickly obtain the test statistic and p-value necessary for drawing rigorous statistical conclusions.

Mastering these techniques is a foundational step in advanced data analysis, enabling you to move beyond simple descriptive statistics and rigorously test claims about population parameters using sample data.

You can use the following online calculators to automatically perform various t-tests: