Analysis of Two-Sample t-Test Timings in Python, R, and SAS

Justin Klein
Miami University
STA402: Statistical Programming
Thomas Fisher

May 14, 2025

The two-sample t-test is a statistical test that is used to determine if there is a significant difference between the means of two independent groups. This test is implemented in Python, R, and SAS, albeit slightly differently in each language. Each programming language follows its own paradigm and may be either compiled or interpreted, affecting resource usage and runtime. Python, R, and SAS all have ways to measure the time taken for an operation. This paper will report the results of an experiment in which the time taken to generate various amounts of pairs of datasets and the time taken to run a t-test on each pair in all three languages were measured.

1 Methods

The programs created in all three languages followed the same format: First, a starting time for the data generation was set to the current time (Neo, 2024) (*Measuring Function Execution Time in R*, 2011) (*Solved: Output the Timing of Each Data Step Into a Table*, 2018). Then, two groups (one pair of two numbers) were generated, and the code to generate a pair would be run n times. Then, this would be repeated m times. The values tested with m were 100, 500, and 1,000, and the values tested with n were 10, 100, and 500. The generated data followed a normal distribution, with one group having a mean of 50 and a standard deviation of 10, and the other group having a mean of 25 and a standard deviation of five. Each dataset was generated in all three languages in the same way: there would be two nested loops, with the first iterating through the values from one to m and the second iterating from one to n. For each value in the nested loop, two numbers were randomly generated following the distribution specified previously - one for each group. This (along with the group number and the index for the first and second loops) was then all appended into one dataset. (Kosourova, 2025) After all of the pairs were generated, the end time was recorded. Then, the starting time for the t-tests was set to the current time. For each pair of elements in the dataset, a two-sided t-test was run. Once all of the t-tests were run for the entire dataset, the ending time was recorded. The time taken (in seconds) to generate all of the data and

to run all of the t-tests, and the t-test results themselves (just for the last pair), were then printed. The results are shown in the next section.

2 Analysis

Programming Language	m and n combination (m, n)	Time to generate m*n pairs of data (s)	Time to run m t-Tests (s)
Python	100, 10	0.004123	0.029847
R	100, 10	0.022675	0.020530
SAS	100, 10	0.016001	0.578999
Python	500, 100	0.148827	1.199296
R	500, 100	0.118374	0.802468
SAS	500, 100	0.031001	3.273999
Python	1,000, 500	1.616826	48.457583
R	1,000, 500	0.902811	18.609748
SAS	1,000, 500	0.078000	8.624000

Table 1: The table of results for the time taken to generate n pairs of data and to run n t-Tests for each language.

Python	R	SAS
last t-test result: T = 10.175268, p-value = 0.000000	last t-test result: T = 7.057114, p-value = 0.000016	Method | Variances | tValue | DF | Probt Pooled | Equal | 6.05 | 18 | <.0001 Satterthwaite | Unequal | 6.05 | 14.658 | <.0001

Table 2: The t-Test results for the last pair in Python, R, and SAS for $m = 100$ and $n = 10$.

This table was created after running each program with each n once; these are not averages. In R, the t-test function returns a list with ten variables: the t-statistic value, the degrees of freedom, the p-value, the confidence interval, the estimated difference in means, the hypothesized null value, the standard error, the alternative hypothesis, the type of t-test performed, and the name of the data. In Python, only the t-statistic, the p-value, the degrees of freedom, and the confidence interval are returned (SciPy, n.d.). For simplicity, only the t-statistic and the p-value of the last t-test were printed. In SAS, the results of the t-test were saved in a new data set, and PROC SQL was used to only print the last result (Clinical Programming Team, 2024). In SAS, the PROC TTEST returns the method of the t-test used, whether the variances are equal or not, the t-statistic, degrees of freedom, and the p-value. In all three languages, the p-value was reported as being less than 0.0001, which suggests that the null hypothesis should be rejected. This means that there is evidence to suggest that the two means are significantly different. This makes sense, as the two generated datasets were purposely set with vastly different means and standard deviations. The T-statistic was also being reported as around five units within each other in all 3 languages as well. All three languages had similar runtimes

when m and n were only 100 and 10, respectively. However, a pattern emerges when m and n increase substantially enough. When m and n were 1,000 and 500, Python was the slowest at both generating the data and performing the t-tests, R was faster than Python, and SAS was the fastest. Interestingly, when m and n were smaller than 1,000 and 500, Python outperformed SAS in calculating the t-tests but was slower than R. In all cases (except when m and n were really small, as the minute differences could be based off of things out of the user's control, like startup time and memory allocation costs), SAS generated the data faster than R and Python, and R performed the t-tests faster than Python.

3 Conclusion

There are multiple ways to construct a dataset and append data to it. There was an issue in SAS where, originally, every iteration of m was a new data set being created, but this made the program run thousands of times slower than just putting it all into one dataframe. However, generating the combined dataframe in R presented challenges. The overhead of this operation varies significantly depending on the implementation; multiple methods were tried, such as using rbind() with a dataframe, adding vectors to a preexisting dataframe, and making four vectors (for m, n, the group, and the randomly generated value), populating them, and then creating a dataframe with the vectors. The first two methods took almost an hour to run; however, the last method ran the fastest and was recorded in Figure 1. However, the "real time" to generate the data and run the t-tests was the only statistic measured in this experiment. In this case, real time is the amount of time between executing the program and the output that is displayed. Memory usage, user time, and system time are all other statistics that could be measured; however, the "real time" is the type of time that is easiest to understand. When testing with m = 10,000 and n = 1,000, SAS encountered an ODS memory limit, so n had to be lowered to 500 and m to 1,000. It is possible that there are more efficient ways of writing the code so that it still runs on one thread and on one process but still runs faster than the current implementation. In addition, it is possible to utilize multiprocessing to achieve even faster results. Future work could investigate the memory usage of each program to identify potential optimizations. It is important to know how memory is allocated and the time complexity of operations performed in these languages so as to make algorithms that can generate the data and perform the t-tests as fast as possible, so as not to unnecessarily waste time.

References

[1] Clinical Programming Team. (2024, August 8). "Using PROC SQL: Creating Macro Variables with the INTO Statement & Other Applications". Quanticate. Retrieved May 13, 2025, from `https://www.quanticate.com/blog/bid/58366/the-into-statement-in-proc-sql-to-create-macro-variables`

[2] Kosourova, E. (2025, April 23). "How to Append Rows to a Data Frame in R (with 7 Code Examples)". Dataquest. Retrieved May 13, 2025, from `https://www.dataquest.io/blog/append-to-dataframe-in-r/`

[3] "Measuring function execution time in R". (2011, June 7). Stack Overflow. Retrieved May 6, 2025, from https://stackoverflow.com/questions/6262203/measuring-function-execution-time-in-r

[4] Neo, B. (2024, June 5). "Timing Functions in Python: A Guide". Built In. Retrieved May 6, 2025, from https://builtin.com/articles/timing-functions-python

[5] SciPy. (n.d.). "test_ind — SciPy v1.15.2 Manual". Numpy and Scipy Documentation. Retrieved May 6, 2025, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html

[6] "Solved: How to stop printing results". (2022, June 30). SAS Support Communities. Retrieved May 6, 2025, from https://communities.sas.com/t5/SAS-Programming/How-to-stop-printing-results/td-p/821154

[7] "Solved: output the timing of each data step into a table". (2018, August 8). SAS Support Communities. Retrieved May 6, 2025, from https://communities.sas.com/t5/SAS-Programming/output-the-timing-of-each-data-step-into-a-table/td-p/485302