

High-Performance Procedures in SAS® 9.4: Comparing Performance of HP and Legacy Procedures

Jessica Montgomery, Sean Joo, Anh Kellermann, Jeffrey D. Kromrey, Diep T. Nguyen, Patricia Rodriguez de Gil, Yan Wang

University of South Florida

ABSTRACT

The growing popularity of big data, coupled with increases in computing capabilities, has led to the development of new SAS® procedures designed to more effectively and efficiently complete such procedures. Although there is a great deal of documentation regarding how to use new high-performance (HP) procedures, relatively little has been disseminated regarding under what specific conditions SAS users can expect performance improvements. This paper serves as a practical guide to getting started with HP procedures in SAS. The paper describes the differences that exist between selected HP procedures (e.g., HPGENSELECT, HPLOGISTIC, HPNLMOD, HPREG, and HPCORR) and their legacy counterparts both in terms of capability and performance, with a particular focus on discrepancies in Real Time required to execute. Simulation will be used to generate data sets that vary on number of observations (10,000, 50,000, 100,000, 500,000, 1,000,000, and 10,000,000) and number of variables (50, 100, 500, and 1,000) to perform these comparisons.

Keywords: HPGENSELECT, HPLOGISTIC, HPNLMOD, HPREG, HPCORR, high-performance analytics procedures

INTRODUCTION

Interest in and utilization of 'big data' has exploded in recent years. As researchers and business interests alike are constantly working with bigger and more complex data sets, it comes as no surprise that new procedures are needed. In response to growing demand, SAS® began incorporating high-performance (HP) procedures into available software releases. These changes to existing legacy procedures can decrease overall real time processing by allowing for multiple threads to run concurrently. HP procedures have been designed to work with the existing capacity of the machines on which they are run. They can run in single-machine mode, using multiple threads to decrease run time and can also work in a distributed, cluster environment where multiple threads exists on multiple nodes (computers). Some updated procedures may also improve performance through increased efficiency, with new programming code that is able to take advantage of computing advances made since the legacy code was developed.

This paper provides an introduction to HP statistics procedures (HPGENSELECT, HPLOGISTIC, HPNLMOD, and HPREG) and HP utility procedures (HPCORR). These procedures were contrasted with related legacy procedures to assess possible differences in performance across simulated conditions. As supporting resources for these procedures are still somewhat sparse, the practical purpose of this paper is to help users of traditional SAS procedures become familiar with available HP counterparts and under what conditions performance differences of various magnitudes can be expected. These assessments were made using simulated data. The following section describes the steps taken as part of the simulation. The paper then turns to constructing comparisons, detailing the primary distinctions between procedural pairs before addressing differences in performance time. The paper closes with a discussion of results and practical recommendations.

HIGH PERFORMANCE PROCEDURES

There are several benefits associated with the set of high performance procedures. Although an exhaustive list is beyond the scope of this paper, the primary updates have been: allowing for multithreading, more efficient data steps, and a greater reliance on appliance memory rather than hard drive capabilities during task execution. For most of these benefits to be accessed in full, the HP procedures should be run in distributed mode, meaning more than one computer is sharing the task. However, as this paper serves as an introduction to these procedures and many new users are unlikely to be working in a distributed, clustered computing environment, we focus on results from single-machine mode, which is the procedural default.

In single-machine mode only one computer is being used, however this single computer can still execute multiple threads simultaneously. Though the benefits associated with revised data steps will not be realized as HP procedures running in single machine mode still access and output data using Base SAS, there should, nonetheless, be differences in run time when contrasting with legacy code due to increases in efficiency and the ability to utilize

multiple threads.

Multithreading allows Real Time to decrease because rather than running an entire job sequentially it allows that job to be subdivided into pieces, based on the number of threads. These pieces can then run concurrently. While this sort of procedure can lead to increases in CPU time, as some amount of time is spent splitting up the task, it tends to decrease Real Time due to overlap. However, it is not the case that Real Time will decrease linearly as additional threads are added. A procedure that runs in 30 seconds on one thread is unlikely to run in 15 seconds on two threads. This is due to the increase in CPU time noted above, but is also related to the parallelizability of the process.

Parallelization refers to the amount of a task that can be subdivided among multiple threads. For a portion of a process to be parallelizable, it must be able to run independently of other concurrent tasks. Therefore, when thinking about improvements in executions times, we must be aware of the amount of parallelization that is possible because this will ultimately control the maximum limit on the increase in speed we will observe. While there are several factors related to the amount of parallelization, for the current purpose two are most primary. First, the size of the problem can impact potential speedup. When the problem is small, much of the time associated with the process will be taken up by bringing in the data. As this must be accomplished before other components of the task can begin, this portion is not parallelizable; thus, when bringing in the data takes up the bulk of the process we are unlikely to see significant performance improvements. Conversely, as the size of the problem grows, we expect the fraction of parallelizable code to do the same, thus increasing the amount of speedup. In a similar vein, the shape of the problem can also impact how processing time is allocated. When there are many observations we are again likely to see more time taken by gathering the data; however, when there are a few observations and many variables, there will be more time devoted to computation and thus, a greater percentage of total processing time available for speedup. Although these components will not be addressed directly in this paper, SAS documentation also includes the following components as factors that can impact scalable speedup:

- Options Specified. Some options available for use with HP procedures are not currently coded to allow for multithreading to take place.
- System Load. Background tasks running on the same machine may divert resources from SAS programs and impact potential speedup.
- Hardware. CPU and memory size, speed, and structure will affect scalability.

To the extent possible this study seeks to control for the remaining factors. As such we did not incorporate any of the available options. Additionally, analyses were done on machines not currently running other programs and the same computer was used for each set of procedural pairs.

METHODS

Data were generated using PROC IML in SAS 9.4. Sample size ($n = 10,000, 50,000, 100,000, 500,000, 1,000,000,$ and $10,000,000$) and number of variables ($k = 50, 100, 500,$ and $1,000$) were manipulated to assess the impact of changes in problem size along with problem shape. These factors were fully crossed for most analyses; however procedures involving Poisson outcomes only considered 10 and 20 variables due to convergence issues with the legacy procedures as the number of variables increased. Data for explanatory variables were drawn from normal distributions, while the shape of the distribution for the outcome variable was altered to allow for testing of the procedures where the outcome is not assumed to be normal (PROC LOGISTIC/HPLOGISTIC and PROC GENMOD/HPGENSELECT).

For each condition, data generation and analysis via the legacy and HP procedures were repeated over 10 iterations. Multiple iterations were selected due to variations noticed in performance time that appeared particular to the unique combination of the generated data set and the procedures being used. To ensure that data presented below were representative of the broader circumstances and were not being driven by a single instance, Real Time and CPU times for each procedure over 10 iterations were averaged. These mean values are reported in the graphs that follow.

As performance time is the focus of this study, results are presented both in terms of CPU Time and Real Time. CPU time is addressed first and then results for differences in Real Time are presented.

RESULTS

Results are presented separately for CPU Time and Real Time. Each graph shows the change in time across various sample sizes ($n = 10,000, 50,000, 100,000, 500,000, 1,000,000, \text{ and } 10,000,000$) for a specified number of explanatory variables.

CPU TIME

Expectations over differences in CPU time varied with the complexity of the data set. Updates embodied in the HP code improved CPU Time relative to legacy code simply due to the increased efficiency brought on by the new code's ability to better utilize improvements in computing capabilities. At the same time, CPU Time was slightly increased in some cases as the process started with each procedure by allocating portions of the overall task to available threads which took some amount of time, even if very small. Therefore, when the problem is small, we expected CPU Time for HP code to be slightly higher due to increases in time associated with allocation, yet as the data become more complex, HP procedures showed improvements relative to legacy procedures due to increases in efficiency created by the new code.

Figure1, Figure 2, and Figure 3 provide support for the expected differences in CPU Time. When sample sizes were relatively small, there was very little divergence between CPU times across procedural pairs. However as sample size increased, we saw greater and greater improvements associated with the HP procedures, suggesting that the efficiency built into this new code provides greater enhancements as the sample size grows.

This pattern, however, did not hold across all procedures examined.

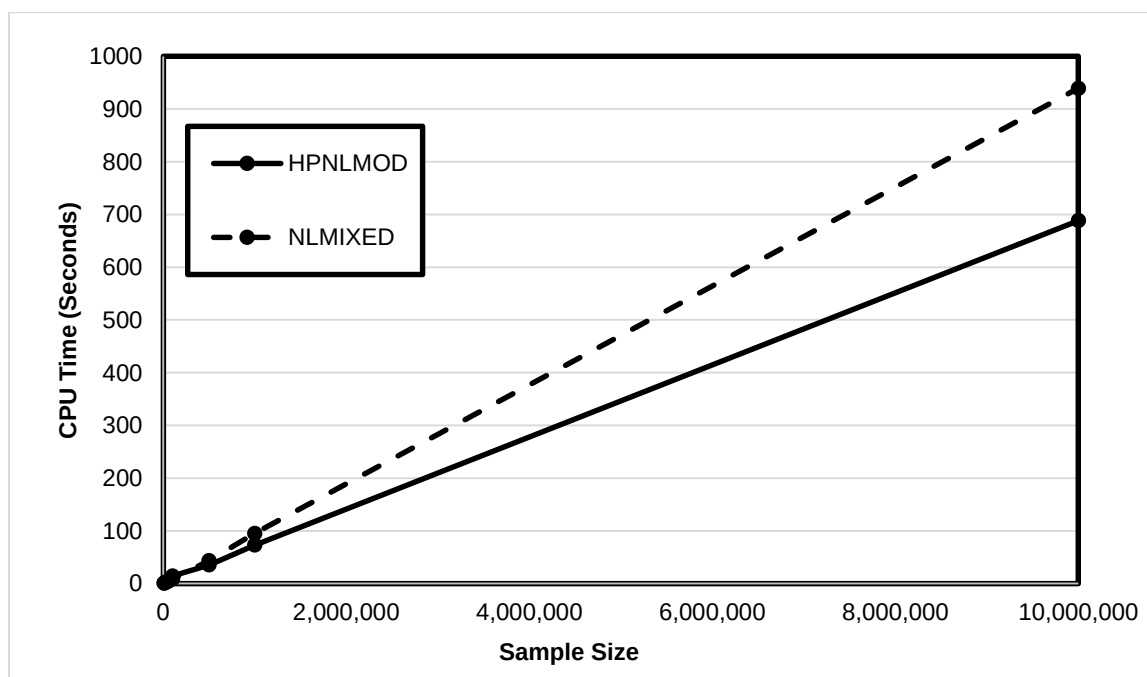


Figure 1: CPU Time for NLMIXED and HPNLMOD ($k=20$)

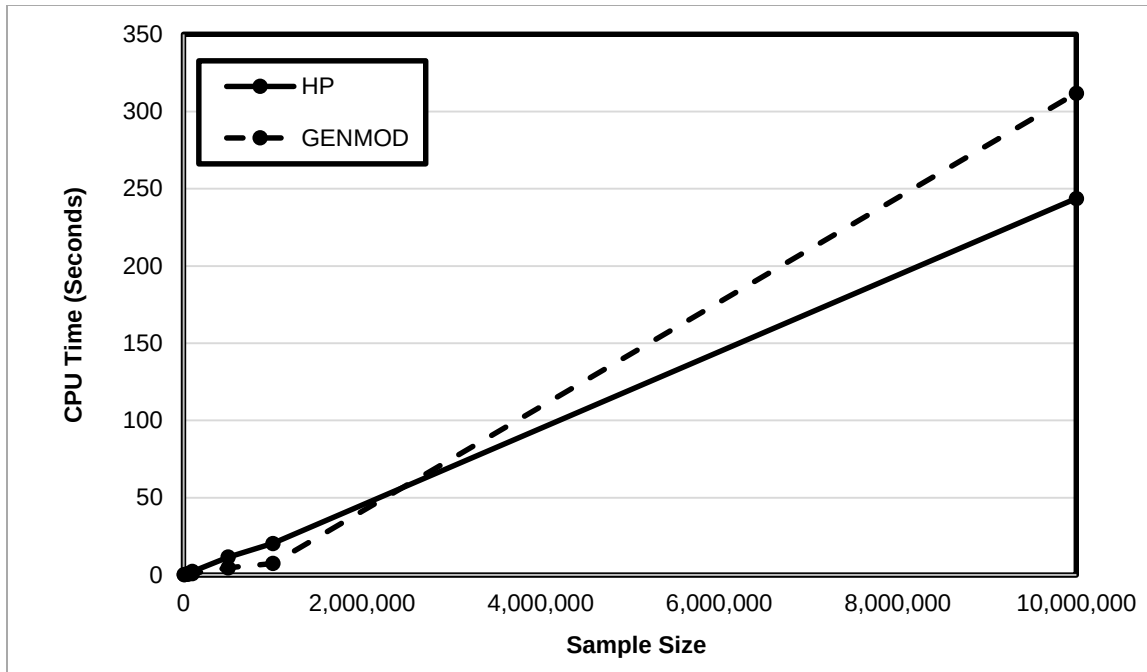


Figure 2: CPU Time for GENMOD and HPGENSELECT (k=50, distribution=Poisson)

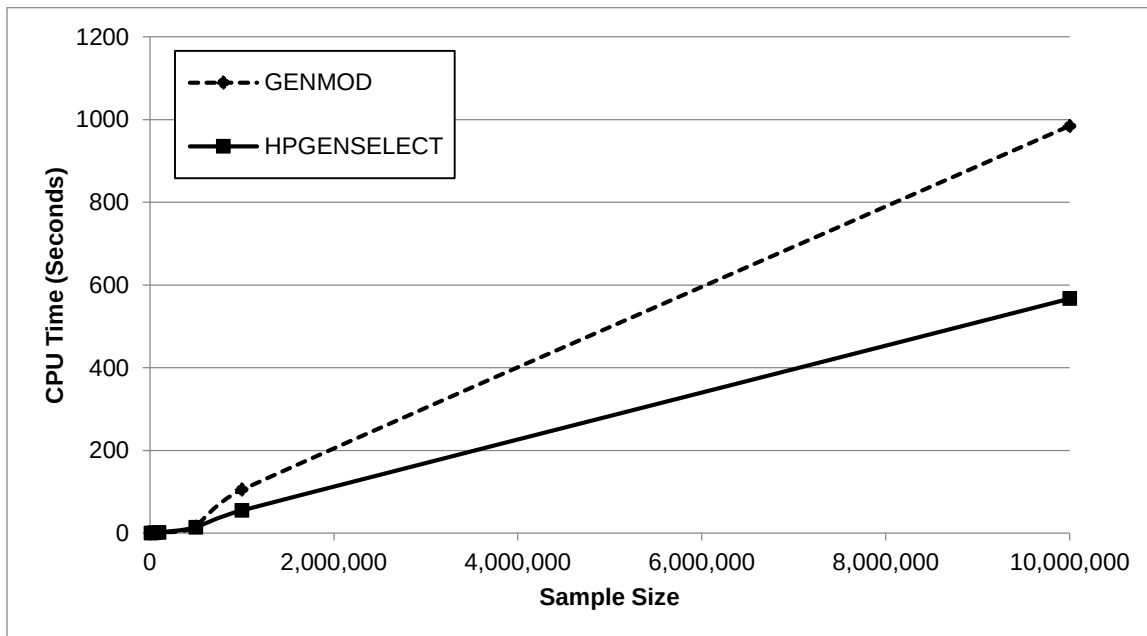


Figure 3: CPU Time for GENMOD and HPGENSELECT (k=50, distribution=Normal)

In the next set of comparison figures (Figure 4, Figure 5, and Figure 6), examining the difference in CPU time for CORR and HPCORR along with REG and HPREG and LOGISTIC and HP LOGISTIC we once again observed very similar values at smaller sample size, but here both procedures increase in CPU Time at similar rates, with the HP procedure consistently showing higher times after roughly 1,000,000 observations. The differences in time also appeared to increase as sample size increased. This results stand in contrast to the results found in the previous set of figures where times were close at small sample sizes, but the HP procedure showed greater efficiency as sample size increased.

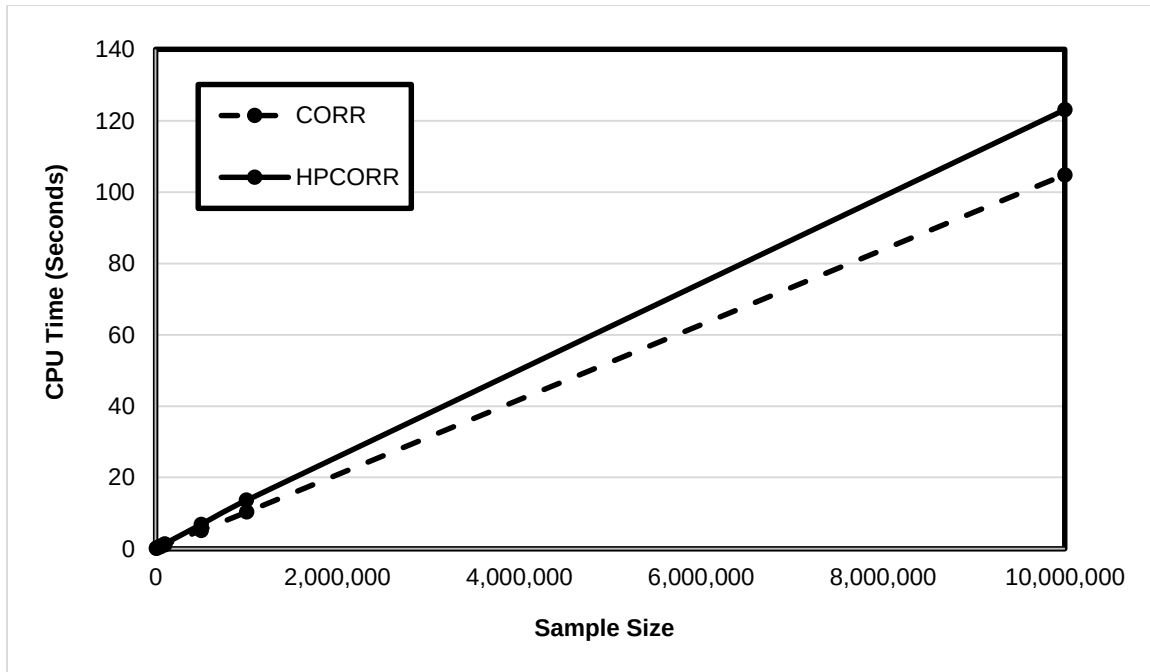


Figure 4: CPU Time for CORR and HPCORR ($k=100$)

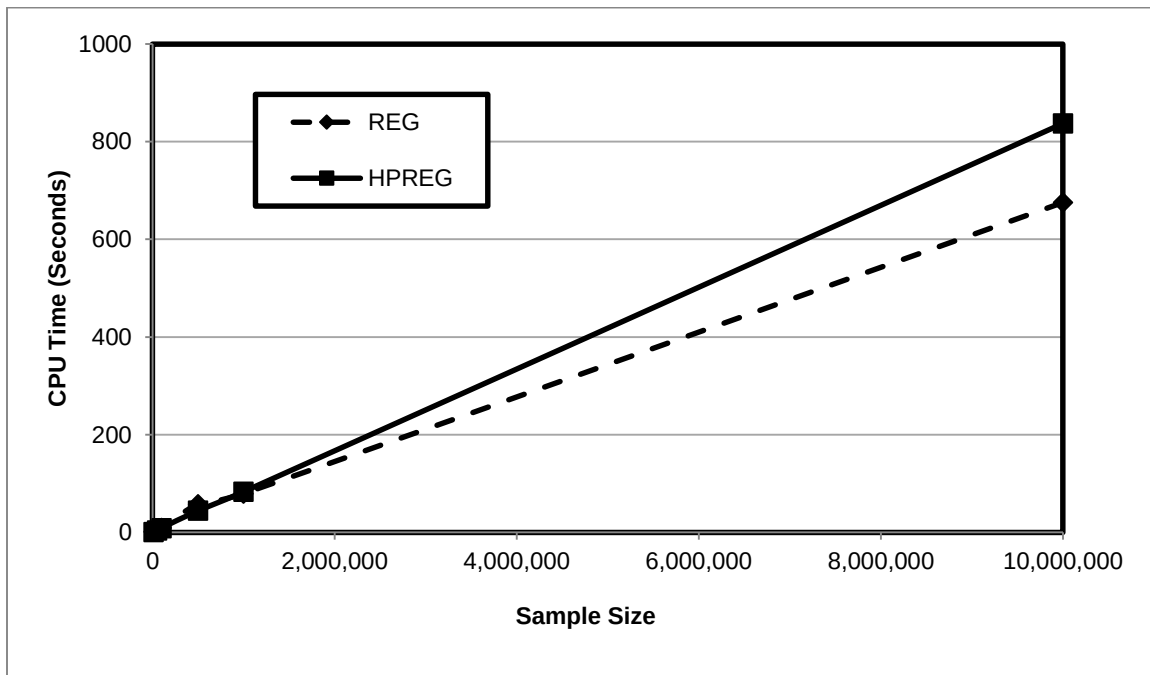


Figure 5: CPU Time for REG and HPREG ($k=50$)

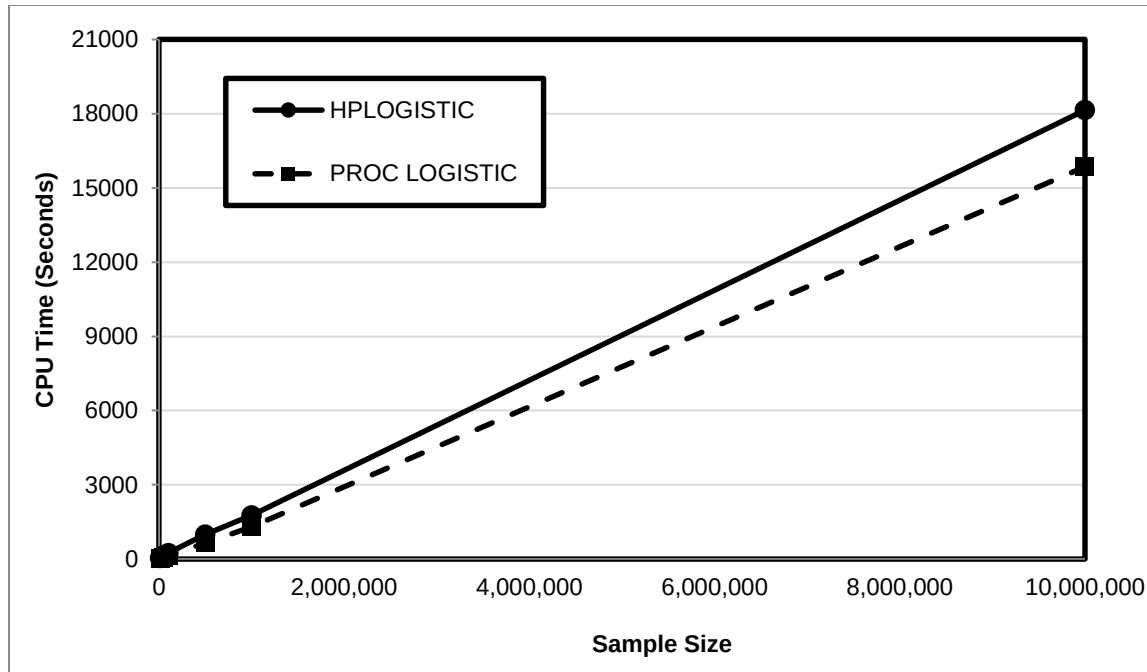


Figure 6: CPU Time for LOGISTIC and HPLOGISTIC (k=1000)

Although the results of this aspect of the analysis are somewhat mixed, with some procedures evidencing an advantage for HP procedures at larger sample sizes while other suggest the opposite results, one pattern remains consistent. In each pair analyzed, the HP procedure and corresponding legacy counterpart behaved quite similarly regardless of the number of explanatory variables included when sample size was below 1,000,000. This suggests that any potential differences are not realized until sample size exceeds a certain bound.

REAL TIME

We would expect the pattern for Real Time to be similar to that for CPU Time in that the size of the discrepancy between procedures should increase as the problem becomes more complex. We would expect also observe improved performance for the HP procedures here, as a focus on Real Time allows for the benefits (and not just the detriments) of multithreading to be taken into account.

In Figures 7 to Figure 10 we observed a similar pattern for these procedures as was observed in the prior section. Regardless of the number of explanatory variables included, when the number of observations is relatively small, Real Time averages are similar for pairs of legacy and HP procedures. However, once the sample size exceeds a certain threshold, the difference in real time required between the procedures increased significantly. All of these pairs, with the exception of LOGISTIC and HP LOGISTIC behaved similarly in terms of real time. This would suggest that some procedures may have gained more in terms of efficiency, while others only showed improvements in Real Time once multithreading is taken into account.

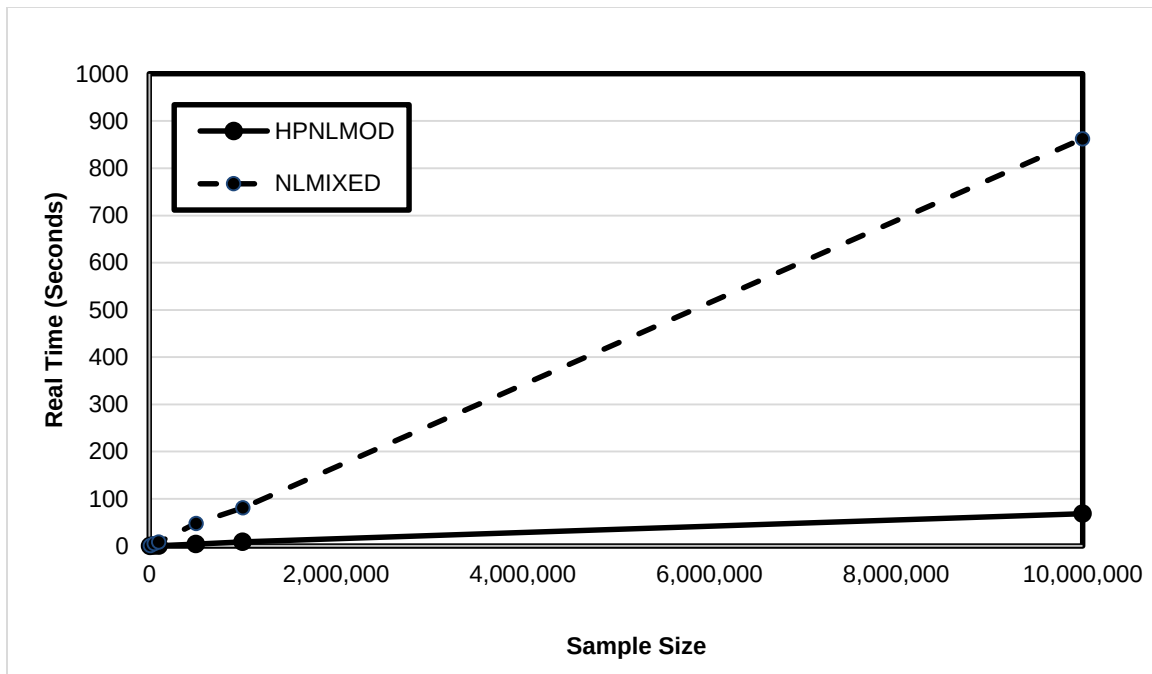


Figure 7: Real Time for NLMIXED and HPNLMOD ($k=10$)

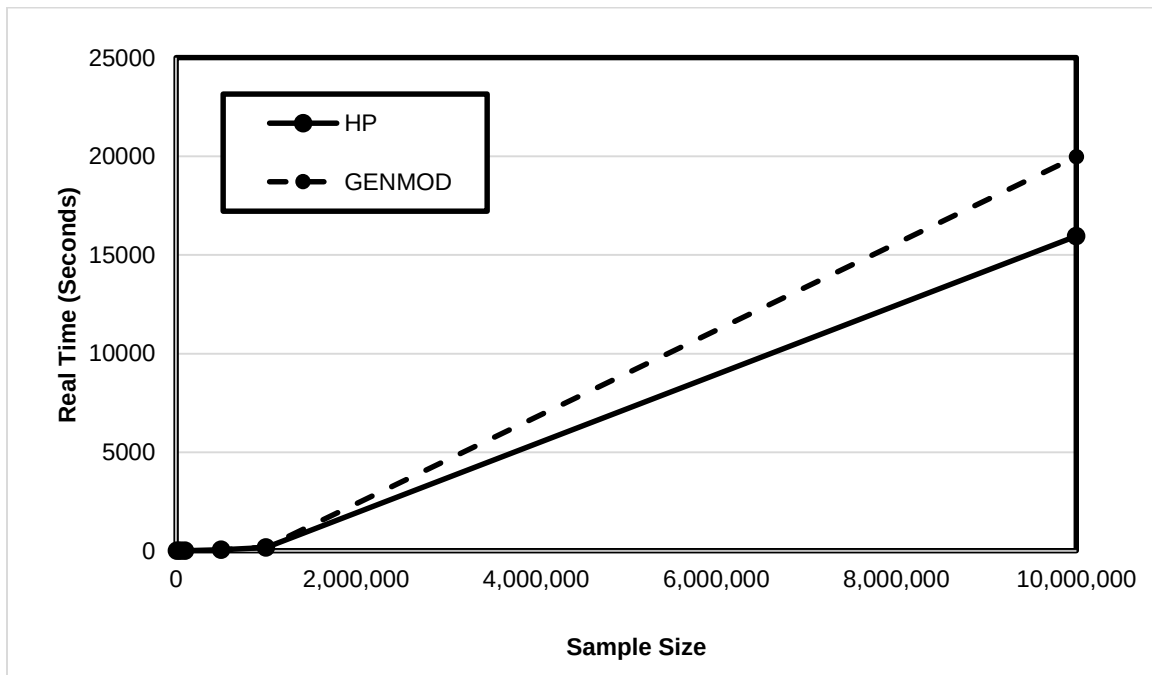


Figure 8: Real Time for GENMOD and HPGENSELECT ($k=100$, distribution=Poisson)

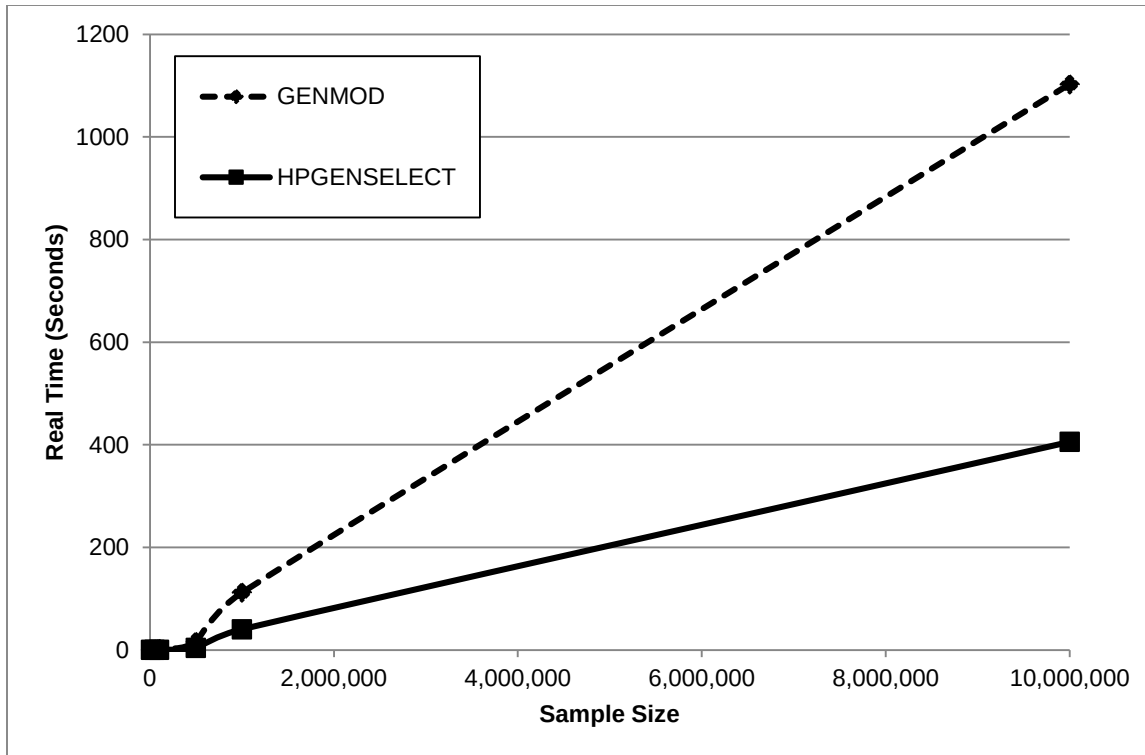


Figure 9: Real Time for GENMOD and HPGENSELECT (k=50, distribution=Normal)

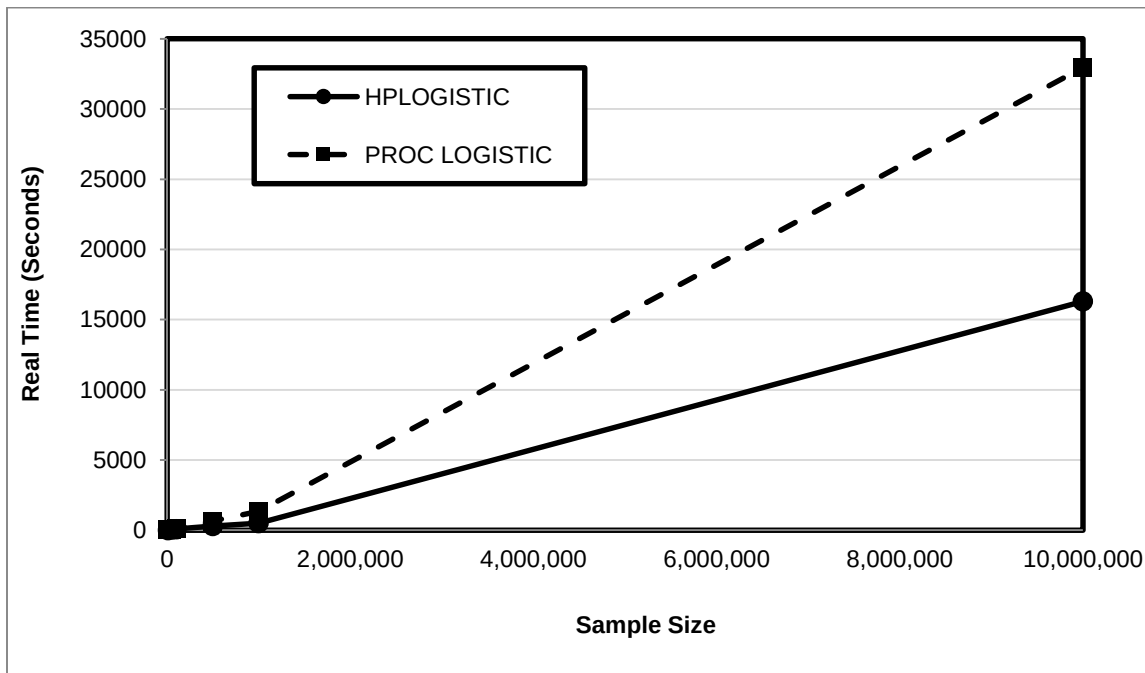


Figure 10: Real Time for LOGISTIC and HPLOGISTIC (k=1000)

These results, however, did not hold across all pairs of procedures. Despite multithreading capabilities several comparisons showed greater Real Time for HP procedures. Results for comparison between CORR and HPCORR and REG and HPREG (Figure 11 and Figure 12) were similar to what was found when examining discrepancies in CPU Time. Despite the capacity for multithreading available with the HP procedures, we nonetheless observed better

performance from the legacy procedures. The results for these procedures once again stand in contrast to the results from the prior section except at small sample sizes where both procedures exhibited similar run times.

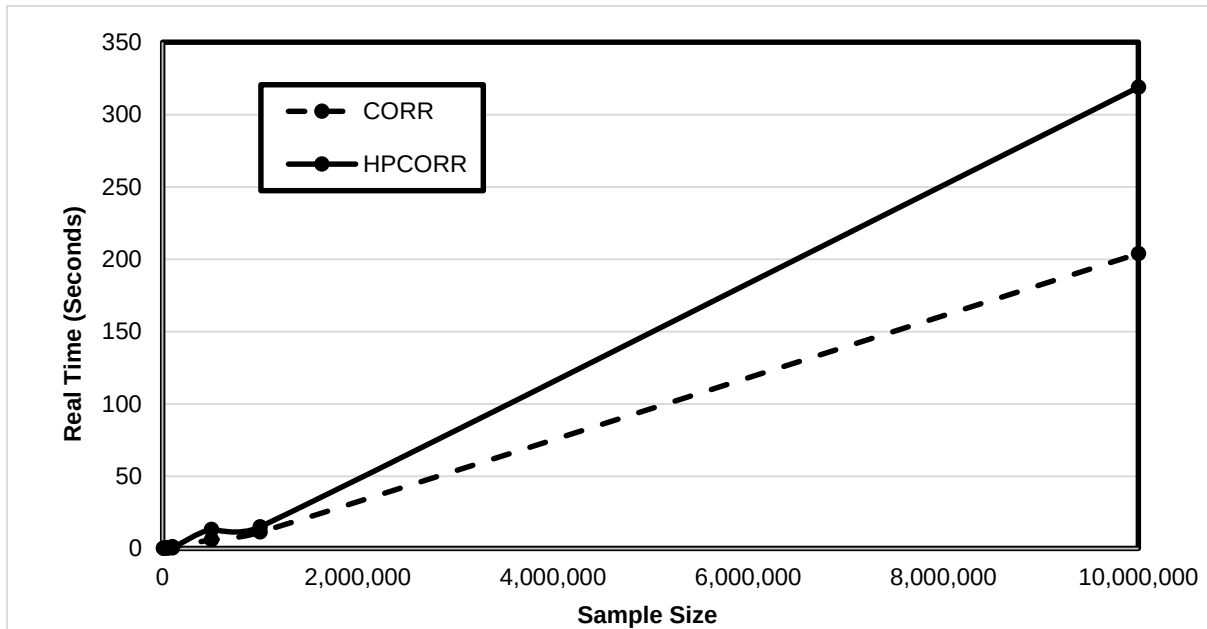


Figure 11: Real Time for CORR and HPCORR (k=100)

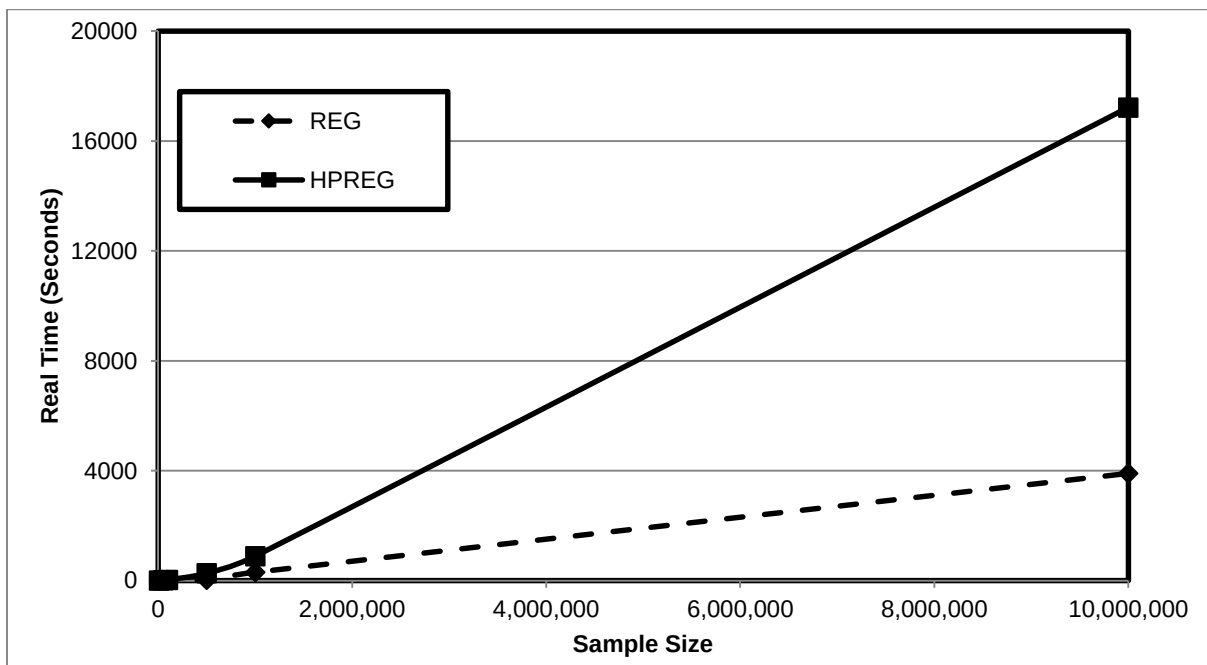


Figure 12: Real Time for REG and HPREG (k=1000)

CONCLUSION

The high performance capabilities available in more recent version of SAS allow users to take advantage of increased computing capabilities in order to analyze larger and more complex data sets. HP procedures improve on legacy code in a number of ways; however this paper has focused primarily on improvements in CPU Time generated

by increased efficiency as well as improvements in Real Time resulting from multithreading.

The results gathered through this simulation study indicate that decreases in run time, both CPU Time and Real Time, are greater as sample size increases. At sample sizes below 1,000,000 there is very little to no difference in processing time across procedural pairs regardless of the specific procedures being used or the number of explanatory variables involved. This pattern was consistently found in all examined conditions.

As the number of observations continued to grow greater variation was observed. For HPNLMOD/NLMIXED, HPGENSELECT/GENMOD with a Poisson outcome, and HPGENSELECT/GENMOD with a Normal outcome the HP procedure executed in consistently less CPU time, with the size of the gap growing as the number of observations increased. The opposite pattern was found with HPREG/REG, HPCORR/CORR, HPLOGISTIC/LOGISTIC and while the differences in CPU Time found with these pairs also increased with sample size, the gap did not grow as quickly as with the prior set of procedures.

These results suggest that improvements in efficiency caused by updates to the code may not operate in a similar manner across all procedural pairs. It is also possible that these results are being driven to some extent by the specific machines utilized by the research team when performing the analyses. To generate a clearer picture of precisely when one can expect decreases in CPU Time it may be necessary to run each of the conditions contained herein on multiple machines to determine which components have a greater relationship with the amount of speed gained.

The number of observation was also related to improvements in Real Time required by HP procedures and their legacy counterparts. The same pairs exhibiting lower CPU time for HP procedures (HPNLMOD/NLMIXED, HPGENSELECT/GENMOD with a Poisson outcome, and HPGENSELECT/GENMOD with a Normal outcome) also evidenced lower Real Times for those procedures. Just as before these differences were minimal when sample size was below 1,000,000 and increased in magnitude as sample size grew beyond that point. An additional HP procedure, HPLOGISTIC as compared to PROC LOGISTIC, evidenced improved Real Time despite the fact that the HP procedure showed higher CPU Time.

As was found in the assessment of CPU Time, some procedures showed results that are more unexpected. HPREG/REG and HPCORR/CORR required greater Real Time for the HP procedures. The difference in Real Time required for the HP procedure and the legacy counterpart was minimal at small sample sizes but the discrepancy began to widen as the number of observations increased beyond 1,000,000. Once again, further studies may be needed to determine precisely what is driving these differences and determine what portion of observed speedup or lack thereof is related to updates to the code, the specifications of the machine being used, and possibly the interaction between these components.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Jessica Montgomery
Enterprise: University of South Florida
Address: 4202 E. Fowler Avenue, EDU 105
City, State ZIP: Tampa, FL 33620,
Work Phone: (813) 974-3220
Fax: (813) 974-4495
E-mail: jnmontgomery@usf.edu
Web: <http://www.coedu.usf.edu/main/departments/me/me.html>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.