IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 18, Issue 1 Ser. I (Jan. – Feb. 2022), PP 39-46
www.iosrjournals.org

The effects of Parallel Computing in Statistical Analysis.

Bibi Sherriza Ali
Department of Mathematics, Statistics & Physics
University of Guyana (Berbice Campus)

Abstract
Parallel computing is a type of computing architecture in which several processors execute or process an application or computation simultaneously. Parallel computing helps in performing large computations by dividing the workload between more than one processor, all of which work through the computation at the same time. Most supercomputers employ parallel computing principles to operate. Parallel computing is also known as parallel processing. This paper focuses on parallel programming in R, which is a programming language and software environment for statistical computing and graphics, there are several packages for parallel computing in R, some of which have existed a long time, e.g.Rmpi, nws, snow, sprint, foreach, multicore.R , which can also be compiled against multi-threaded linear algebra libraries, which helps to speed up calculations.

Keywords*: Parallel computing, computation simultaneously, programming language, statistical computing and graphics.*

Date of Submission: 01-01-2022 Date of Acceptance: 12-01-2022

I. Introduction

Traditionally, software has been written for serial computation: A problem is broken into a discrete series of instructions, instructions are executed sequentially one after another, executed on a single processor, only one instruction may execute at any moment in time. Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). The computational problem allows problems to be broken into discrete parts that can be solved concurrently where each part is further broken down to a series of instructions .Instructions from each part execute simultaneously on different processors.

The real world is massively parallel i.e. in the natural world, many complex, interrelated events are happening at the same time, yet within a temporal sequence. Compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real world phenomena. One may justified that the real reasons for using parallel computing are as follows save time and/or money, solve larger/more complex problems, provide concurrency, take advantage of non-local resources, and make better use of underlying parallel hardware. Historically, parallel computing has been considered to be "the high end of computing", and has been used to model difficult problems in many areas of science and engineering such as Atmosphere, Earth, Environment, Physics, Bioscience, Chemistry, and Geology etc.

State of the Art in Parallel Computing with R.

When using the statistical package R, we often need to repeat a computation, or a series of computations, many times, this was then accomplished by using the famous “for loop”. However, the use of “for loop” can be time consuming especially if there are a large number of computations to carry out. To make the task of complex computation less troublesome, one can considered parallel computing, parallel computing deals with hardware and software for computational where many calculations are done simultaneously as it is an ideal function to improve calculations capacity. Regardless of its application, parallel computing has three basic steps: split the problem into pieces, execute in parallel, and collect the results. Many programming languages and li braries have been designed for programming computers .The packages for parallel computing with R are as follows: Rmpi, rpvm, nws, snow, snowFT, snowfall, papply, biopara, taskPR.

Rmpi package has scripts to lunch R instances at the slaves from the mpi.spawn.Rslaves ().These occurrences until closed, with mpi.close.Rslaves ().This package offers many R specificfunctions, in addition to wrapping the MPI API.rpvm package is an interface to PVM.It supports low-level PVM functions from R.nws package offers functions to store data in a “NetworkSpace (NWS)”, then uses the ‘sleigh’mechanism for parallel execution. In the nws package there are essentially two basic functions for performing tasks in parallel these are eachElem () and eachWorker ().The snow package know as Simple Network of workstations, enables simple parallel computing in R. This package has scripts to lunch R instances on the slaves (c1<makeCluster ()).These occurrences run until they are closed explicitly by using the command stopCluster (c1).It also provides the apply () function which is a high level parallel function. The snowFT package is an addition of the snow package assistant fault tolerance, reproducibility and additional management features. The snowfall package also support simple parallel computing in R.It is known as the top level wrapper around the snow package, essentially built to ease development of parallel computing in R.The papply package was developed to augment the parallel apply ()-functionality of the Rmpi package, which is very much similar to apply () function, papply () applies a function to all items within a list and then return the list with the necessary results. Biopara package enables user to allocate execution of parallel problems over several machines, which uses the raw socket connections(SSH).However, in this package there is only one function called biopara().The biopara package provided limited fault tolerance and load balancing, and is therefore no longer being maintained. Finally the taskPR package, this package is an interface that uses the MPI implementation LAM/MPI for parallel computing. If no LAM exists the tasks will be executed serially on the manager. A function that is required to run in parallel has to enclose within the function PE () and then will be executed at the first slave, next PE () call will be executed on the next slave and so forth. In this package there is no support of high level- parallel function such as apply().

Currently there are two R packages and one Java-based software group exist for grid computing in R.These are GridR MultiR AND Biocep-R.There are five dissimilar R packages for multi-core systems to execute computations in R,as follows:pnmath and pnmath(),fork,R/parallel, romp and multicore.

The performance of these parallel computing packages are highly influence by a number of reasons, such as design and efficiency of the application, as well as technology and hardware are probable leading factors. We can say that RMPI and snow are the two well-developed R packages that stands out for the use in high performance multicomputer environments. They both have acceptable usability, support a spectrum of functionality for parallel computing in R, and produce a good performance packages for multicomputer environments, are considered to be well developed and is essentially useful. However, there is room for some minor improvements in documentation and performance optimization. When performing parallel computing with R, attention should be taken when dealing with unnecessary data transfer and to maximize the amount of computation performed in each remote evaluation.

i. How does the input space dimensionality *p* affect the computational time as a function of the number of CPU cores?

Exploration to find out how does the dimensionality of p (variables) affect computational time as a function of the number of CPU cores. For this investigation the sample size and bootstrap samples remain constant while number of variables changes to make a conclusion of the investigation.
Now exploring when p=10, 30, 40, 80,100, where sample size=200 and bootstrap sample =840

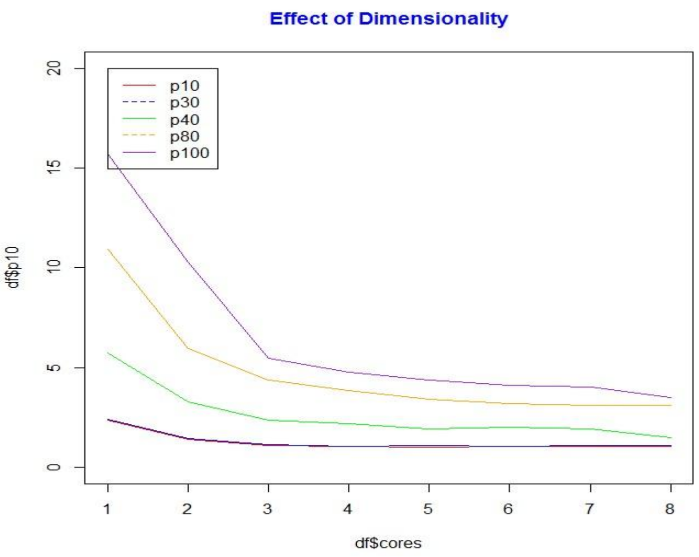

Figure 1: Plot showing the time/core

As the dimensionality that is variable size (p) increased, the computational time increase for the ordinary least square regression. This can be seen in figure 1, which indicates the computational time for dimensionality size equal to 10, 30, 40, 80, and 100 as functions of the number of CPU cores used during the computing process in this case the number of CPU is 8.It is important to note that the computational time decreases as the number of CPU increses.This is true for all dimensionality space.

ii. How does the sample size *n* affect the computational time as a function of the number of CPU cores?

For this investigation the number of variables and bootstrap samples remain constant while sample size changes to make a conclusion of the investigation.
Exploring when sample size=10, 20,40,60,80,100, p =50 and bootstrap sample =840

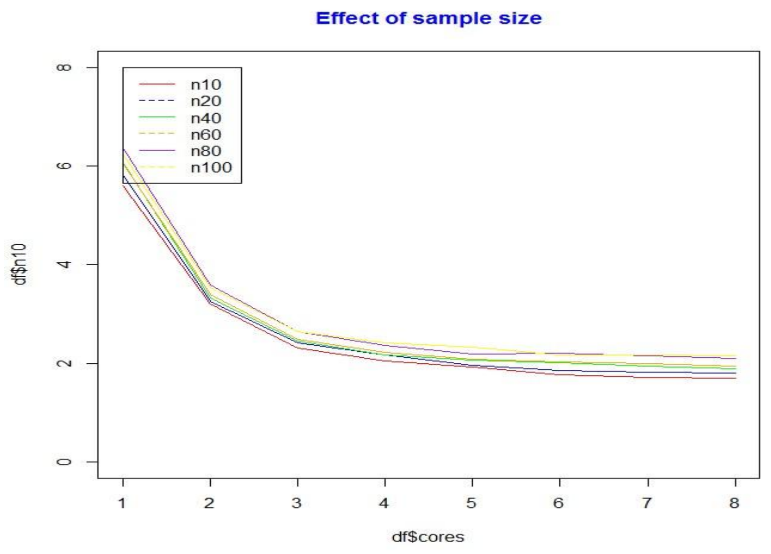

Figure 2: Plot showing the time/core

Figure 2, shows the computational time for sample size equal to 10, 20,40,60,80,100 as the functions of the number of CPU cores used during the computational process for the ordinary least square regression. Unlike dimensionality space (p), the size of the sample (n) does not have an effect on the computational time as a function of the number of CPU cores used.

iii. How does the number of bootstrap samples *L* affect the computational time as a function of the number of CPU cores?

For this investigation the number of variables and sample size remain constant while the number of bootstrap samples changes to make a conclusion of the investigation.

Exploring when bootstrap sample =50, 100, 5000,500,700, sample size=1000 p =50

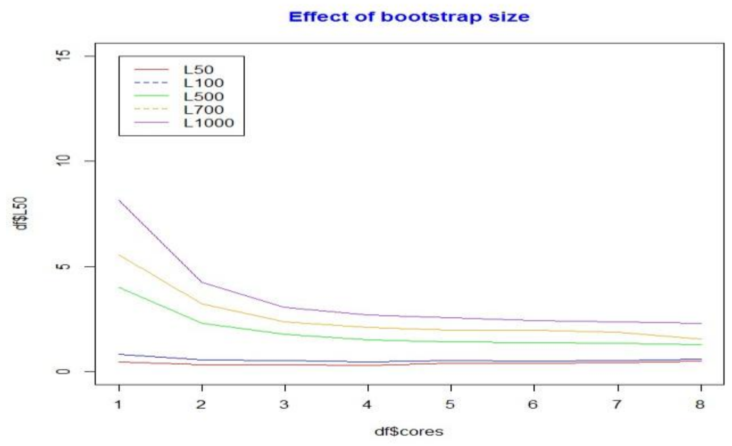

Figure 3: Plot showing the time/core

As the bootstrap sample (L) increases the computational time for the ordinary least square regression will also increase. This conclusion is made from figure 3, where bootstrap samples equal to 50,100,500,700 and 1000 were used, as functions of number of CPU cores used during the computational process. Again we can see that the computational time decreases as the number of CPU cores increases.

Investigating the importance of variables for the linear model using 100 variables, sample size =1000 and bootstrap sample =1000.
To test the most significance variables an alpha value of 10% was chosen.

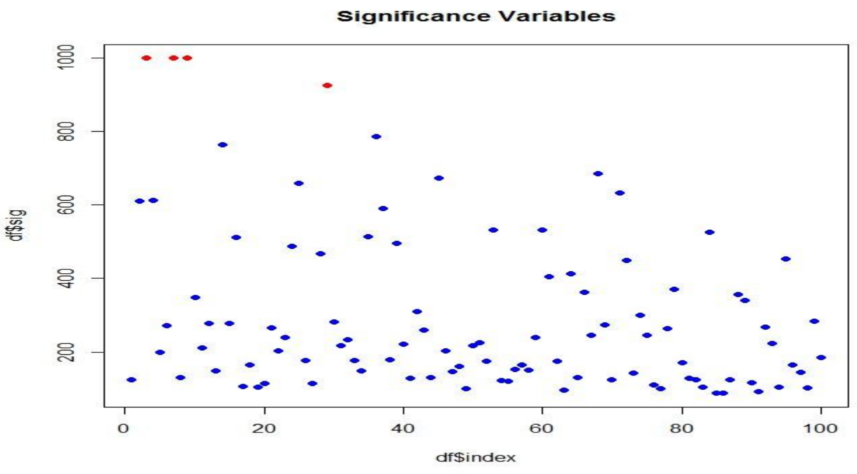

Figure4: Significance variables

It is important to note that variable x3, x7 and x9 would always be the three most significant variables in this case, because the simulated data was generated by a function called
"generate.lm.xy", which entails the variable named "eta"which influence the response variable "y".In this function the variable "eta"is highly dependent on the variables x3, x7, and x9.

Variable	Frequency
X3	1000
X7	1000
X9	1000
X29	925

Table1: Frequency of variable

Below is histogram showing the most important significant coefficients, x3, x7, x9 and x29.

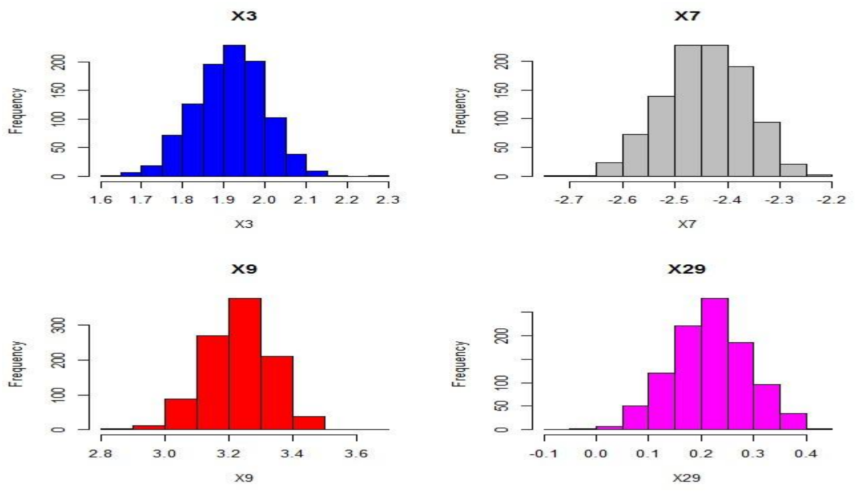

Figure 5: Histograms of significant coefficients.

Investigating how the dimensionality affects computational time as a function of the number of CPU cores when using generalized linear model.
Exploring when p=10, 20, 50, 80,100,200, where sample size=1000 and bootstrap sample =840

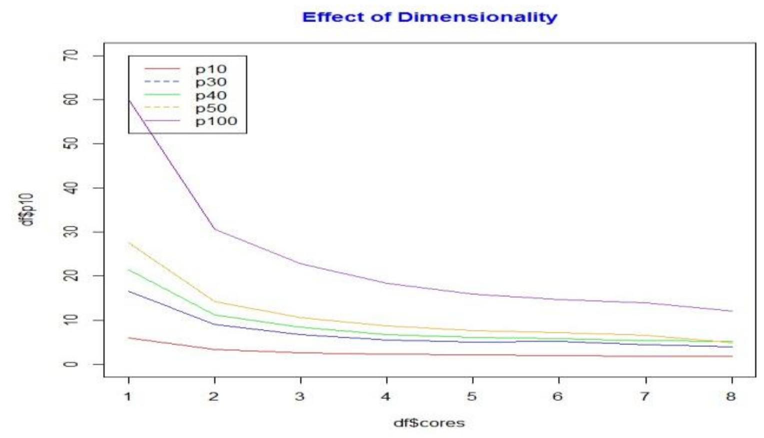

Figure 6: Plot showing the time/core

As the dimensionality (p) increased, the computational time increases when the multiple linear regressions were performed. The dimensionality effect for the multiple linear regression is as the same as the ordinary least square regression i.e. as the dimensionality space increases the computational time increases as well, also as the number of CPU cores increases the computational time decreases.

Exploring to find out how does the sample size affect computational time as a function of the number of CPU cores with generalized linear model. For this investigation the number of variables and bootstrap samples remain constant while sample size changes to make a conclusion of the investigation.
Exploring when sample size=10,20,30,40,50,60,70,80,90,100, p =50 and bootstrap sample =840

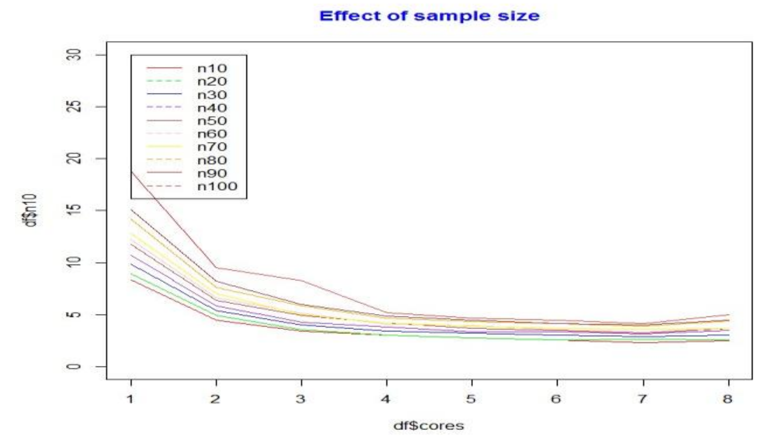

Figure7: Plot showing the time/core

For the multiple linear regressions, ten evenly space sample sizes were chosen to investigate the effect of sample as function of the number of CPU cores use during the computational process. Again has seen with the linear regression there is a lack of significance difference in the computational time performances, which indicates that sample sizes does not have much effect as the dimensionality space.

Exploring to find out how does the number of bootstrap samples affect computational time as a function of the number of CPU cores for generalized linear model. For this investigation the number of variables and sample size remain constant while the number of bootstrap samples changes to make a conclusion of the investigation.

Exploring when bootstrap sample =50,100,500,700,1000, sample size=100, p =50

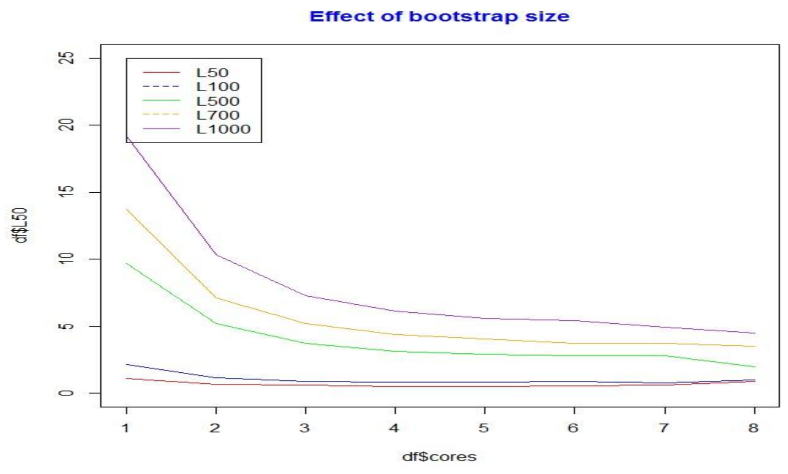

Figure8: Plot showing the time/core

As seen with the original least square regression, an increase in the bootstrap sample will result in an increase in the computational time for the multiple linear regressions. It is also important to note that as the bootstrap sample increases the computational time decreases as the number of CPU cores increases.

Investigating the importance of variables for the generalized linear model using 20 variables, sample size =200 and bootstrap sample =100.
To test the most significance variables an alpha value of 10% was chosen.

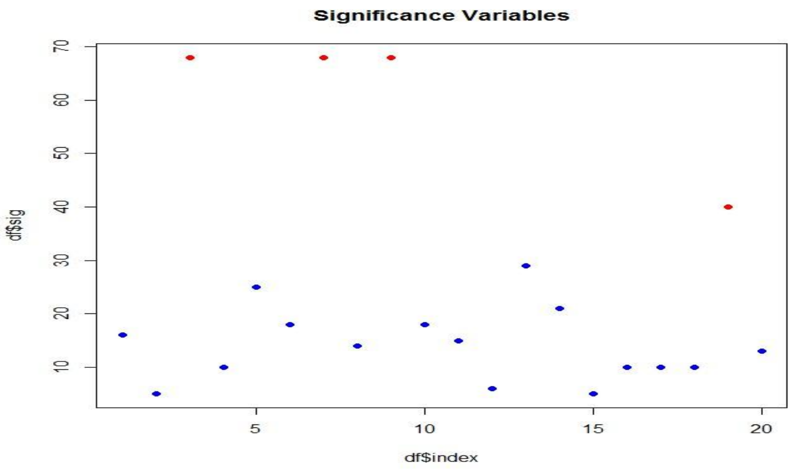

Figure 9: Significance variables

Again as seen in the ordinary least square regression, the variables x3, x7, and x9 were identified as the three most significant varaibles.Here the simulated data were generated by a function called "generate glm.xy"which contained the variable "eta"that is highly dependent on the variable x3, x7, and x9, which has a significant effect on the response variable"y".As such variable x3, x7 and x9 will always be significant in this case.

Variable	Frequency
X3	68
X7	68
X9	68
X12	40

Table2: Frequency of variable

Below are the histograms showing the most important significant coefficients, x3, x7, x9 and x19.

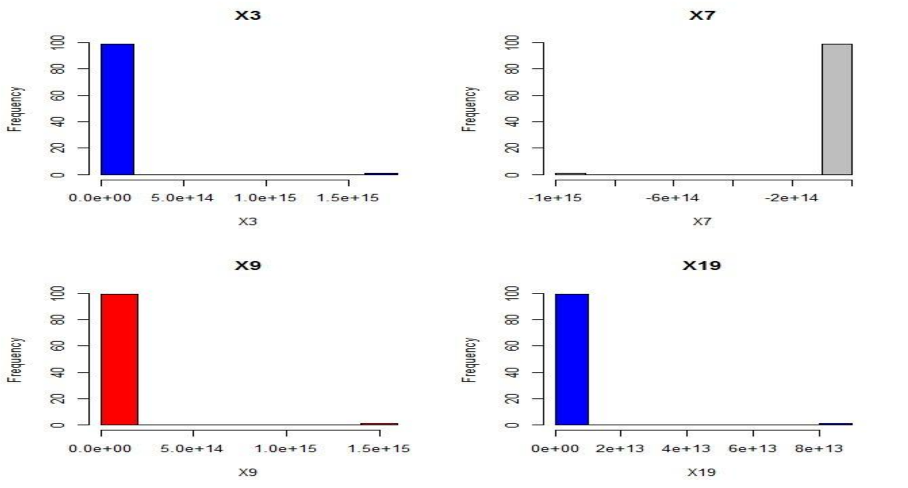

Figure 10: Histograms of significant coefficients.

Finally an investigation was conducted to find out the effect of the basic task using the regression generator and the classification generator.

To compare the effect of parallel analysis using linear model and generalized linear model, the complexity of the task is considered the same. A sample size of 1000, space dimensionality of 50 and bootstrap sample of 1000 was chosen. The figure below explains the importance of parallel analysis, for linear model the task is considered rather simple when compare to generalized linear model, therefore it can be concluded that there is a lower computational time per core whereas for the generalized linear model this task is considered complex, as such the computational time is much higher.

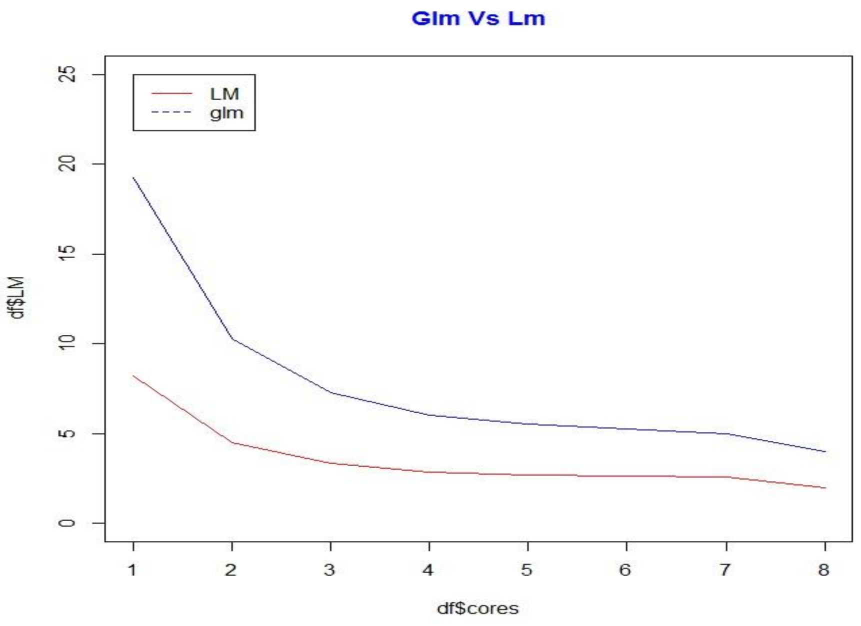

Figure 11: Parallel Analysis-linear model vs Generalized linear model.

When consider the linear model and generalized linear model machinery, the number of variable, and bootstrap sampling do affect the computational time as a function of the number of CPU cores

Single core is an ideal core for simple task rather than 2 or more cores since single core will be able to get the task done faster than the larger core, as there is no need for parallel computing i.e. tasks are divided between more than one processor, one would say that using more than a single core for a simple tasks is rather useless and time consuming as this can be done quickly with just one core, however, for complex tasks a single core will struggle to execute the necessary computation therefore it is necessary to use more than one core, this is where parallel computing is ideal or essential.

II. Findings and Conclusions

In this study, the researcher was curious to investigate the effect of sample size, variable size and bootstrap sample size for the linear model and generalized linear model and to make a conclusion on the effect of parallel analysis. In statistical analysis, sometimes dealing with large data may be classified as a complex task in the analysis stage as such many may encountered difficulties in getting this task accomplished in a timely manner, or it might be a long drawn out process, especially if we are to use “for loop” in R when there is multiple computation to be formed. With parallel analysis complex tasks can be carried out rather easily as this process problems are broken down into a discrete series of instructions, which enables calculations to be carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel").However, there are times in statistically analysis where parallel computing is not ever needed and may considered to be a waste of time.

In this study the effect of dimensionality space (p), sample size (n) and bootstrap samples (L) on the computational time as functions of the number of CPU were highly investigated. In this investigation it was concluded that both dimensionality space, and bootstrap samples increases the computational time has p and n increases, whereas sample size does not seems to have any significant effect on the computational time as it increases. However, for simple tasks smaller number of CPU cores performs better rather than larger core, but for complex tasks small CPU cores struggle to perform the necessary task and is therefore recommended to use larger CPU cores when the task at hand is considered complex and smaller cores when simple tasks are required to be computed.

Reference

[1]. Brian Ripley [aut, cre, cph], Bill Venables [ctb], Douglas M. Bates [ctb], Kurt Hornik [trl] (partial port ca 1998), Albrecht Gebhardt [trl] (partial port ca 1998), David Firth [ctb], MASS: Support Functions and Datasets for Venables and Ripley's MASS,August 10 ,2021,< https://cran.r-project.org/web/packages/MASS/index.html>

[2]. **Eddelbuettel, D, 2019, Parallel Computing With R: A Brief Review, August 15, 2021** <https://arxiv.org/abs/1912.11144>

[3]. Esam Mahdi , 2014, A Survey of R Software for Parallel Computing, American Journal of Applied Mathematics and Statistics, July 10,2021, <https://www.researchgate.net/publication/264518579_A_Survey_of_R_Software_for_Parallel_Computing>

[4]. Hadley Wickham [aut, cre], plyr: Tools for Splitting, Applying and Combining Data,August 15 ,2021, < https://cran.r-project.org/web/packages/plyr/index.html>

[5]. Hao Yu, Luke Tierney, Ulrich Mansmann, 2009, State-of-the-art in Parallel Computing with R, July10, 2021, <https://www.researchgate.net/publication/38105173_State_of_the_Art_in_Parallel_Computing_with_R>

[6]. Jochen Knaus, 2015, Easier cluster computing (based on snow), August 14, 2021, < https://cran.r-project.org/web/packages/snowfall/snowfall.pdf>

[7]. Kjetil Halvorsen, The Elements of Statistical Learning, Data Mining, Inference, and Prediction, August16, 2021, <https://ftp.unibayreuth.de/math/statlib/R/CRAN/src/contrib/Descriptons/ElemStatLearn.html>

[8]. Lawrence Livermore National Laboratory, Introduction to Parallel Computing Tutorial, August 01, 2021, < https://computing.llnl.gov/tutorials/parallel_comp/ **>**

[9]. Michelle Wallig [cre], Microsoft [aut, cph], Steve Weston, foreach: Provides Foreach Looping Construct, September 15,2021, < https://github.com/RevolutionAnalytics/foreach>

[10]. Michelle Wallig [cre], Microsoft Corporation [aut, cph], Steve Weston [aut], Dan Tenenbaum [ctb] , doParallel: Foreach Parallel Adaptor for the 'parallel', Package,August 31, 2021, < https://cran.r-project.org/web/packages/doParallel/index.html>

Bibi Sherriza Ali. "The effects of Parallel Computing in Statistical Analysis." *IOSR Journal of Mathematics (IOSR-JM)*, 18(1), (2022): pp. 39-46.