Final Project: Elapsed Time for Functions in R for Generating Random Numbers for Simulation

Vanessa Tandiman

4/26/2021

Introduction

Simulation is an important part of a statistical analysis. Often times, before collecting the data, a researcher performs a simulation of what he thinks the data would look like, based on his knowledge or the available prior information. This helps him to know what to expect, to anticipate the statistical analysis involved, and to assess the preferred research design. For example, a researcher can run a simulation for a proposed research design to see if the effect of the variable of interest would have to be analyzed using an approximate F-test rather than an exact F-test.

Part of a simulation is generating random numbers. In R, there are a few functions that perform this task. When the number of desired simulation is large, the elapsed time could affect the efficiency of the process. This experiment will compare the performance of three functions, *replicate*, *loop*, and *rmnorm* in generate a random sample from a standard normal distribution.

This project attempts to address the research questions, which function, if any among the three functions, generates random number the fastest? This will be done by comparing the elapsed time of the three functions to generate a random sample from a standard normal distribution, while varying the numbers of simulation. The number of simulation represents the number of repetition in generating the random sample.The best function would be the function that can perform this task in the shortest amount of time.

Experiment Design There are two factors in this experiment: - The three functions: *replicate*, *loop*, and *rmnorm*. This is considered a fixed effect with three levels. - The number of simulations: 100; 1,000; 10,000; 100,000 and 1,000,000. These numbers are chosen arbitrarily to represent different simulation sizes. Because the arbitrary selection, this is a fixed effect with five levels. This experiment will be done on two devices, a Dell laptop and a Dell PC. This is considered a blocking variable (fixed), with two levels.

The following is the specifications of the two devices:

Table 1: Technical Spec of the Machines Used in the Experiment

Device	Processor	RAM	R Version
Dell Laptop	Intel(R) Core(TM) i5-10210U CPU 1.60GHz 2.11 GHz	8.00 GB (7.79 GB usable)	64-bit
Dell PC	Intel(R) Core(TM) i7-9700 CPU 3.00GHz 3.00 GHz	16.00 GB (15.8 GB usable)	64-bit

We see that the PC seems to be a better machine. For each combination of factors, we will perform 10 replications. The response of interest is the elapsed time while the functions perform the task. It will be measured using the tic() and toc() function in R, and the unit is in seconds (s).

The model associated with this design This is a two-factor fixed effects with blocks design. The statistical

model is $y_{ijkl} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \gamma_k + \epsilon_{ijkl}$, where y_{ijkl} is the elapsed time to execute the function, measured in seconds (s), μ is the baseline elapsed time (s), α_i is the Function effect, $i = 1, 2, 3$, β_j is the Simulation Size effect, $j = 1, 2, 3, 4, 5$, $(\alpha\beta)_{ij}$ is the interaction between the Function and Simulation Size, γ_k is the Block effect of the machine used, $k = 1, 2$, and ϵ_{ijkl} is the error term of the l-th replicate, and we assume that $\epsilon_{ijkl} \sim N(0, \sigma^2)$, and $l = 1, 2, 3, \ldots, 10$. We assume there is no interaction between the factors and the blocking variable.

The hypotheses being tested are:

$$H_0 : \alpha_1 = \alpha_2 = \alpha_3 \text{ vs } H_1 : \alpha_i \neq \alpha_i' \text{ for some } i \neq i'$$

$$H_0 : \beta_1 = \beta_2 = \beta_3 = \beta_4 = \beta_5 \text{ vs } H_1 : \beta_j \neq \beta_j' \text{ for some } j \neq j'$$

$$H_0 : (\alpha\beta)_{11} = (\alpha\beta)_{12} = (\alpha\beta)_{13} = \cdots = (\alpha\beta)_{35} \text{ vs } H_1 : (\alpha\beta)_{ij} \neq (\alpha\beta)_{i'j'} \text{ for some } ij \neq i'j'$$

If any H_0 is rejected, a Multiple Comparison Test (Tukey Test) will be performed to determine which group is different.

The levels for simulation size on the $log10$ scale are equally spaced, so we can test for linear and quadratic trend on the $log10$ scale.

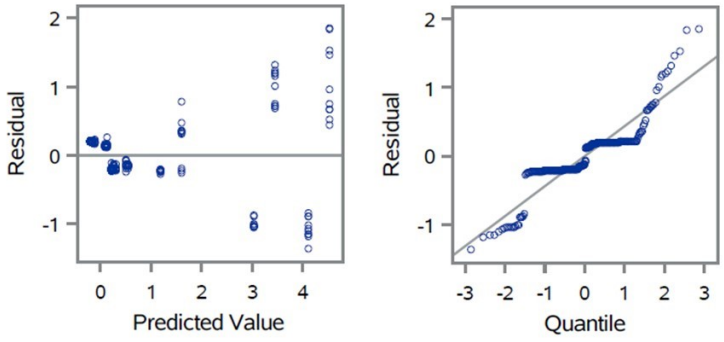
Data Collection Process

1. We set the simulated sample size to 10. The code line that will be used to generate the random sample from the standard normal distribution is *rnorm(10,0,1)*.

2. Randomly determine the order of Function-Simulation Size combination will be used for data collection.
3. For each Function-Simulation Size combination, run the code on R 10 times on one machine (Laptop or PC), and record the elapsed time. Do the same on the other machine.
4. Repeat the process for all combination of Function-Simulation Size.

Summary of Statistical Findings

We start by assessing the Residual_Predicted Value plot and the Residual-Quantile plot When we fit the ANOVA model to the data to see if the assumption for Homogeneity of Variance and the normality assumption is met.



We can see a funneling pattern on the residual plot, which indicates that the Homogeneity of Variance (HoV) assumption is violated. We also see that the dots on the Residual-Quantile plot deviates from the linear trend for most of the part. While ANOVA is quite robust to the violation of the normality assumption, we are more concerned about the violation of the HoV assumption. These violations make the ANOVA result unreliable.

To address this, we perform the empirical method on the restructured version of the data. The data is restructured into a one-way ANOVA, having one categorical variable that represents the machine, function, and the simulation size of the observation. This categorical variable has 30 levels. We perform the empirical method to find the value of λ such that the transformation $y^* = y^\lambda$ meet the HoV assumption better. Below is the ANOVA table to find the empirical selection for α, where $\hat{\alpha} = 0.51$. This ANOVA table shows a decent fit for the simple linear regression, as seen on the R-square value of 75.24%.

ANOVA TO FIND EMPIRICAL SELECTION OF ALPHA

The GLM Procedure

Dependent Variable: logstd

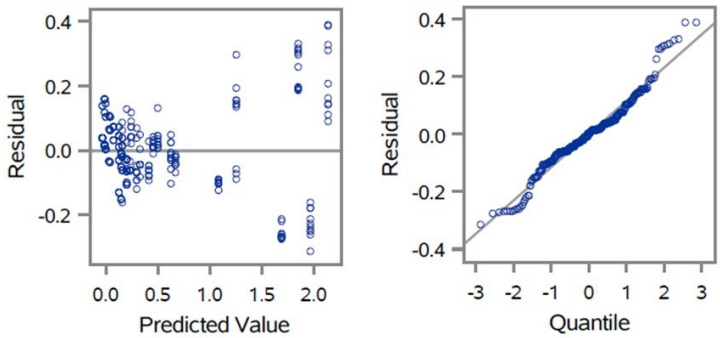
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	38.02407468	38.02407468	85.10	<.0001
Error	28	12.51131566	0.44683270		
Corrected Total	29	50.53539034			

R-Square	Coeff Var	Root MSE	logstd Mean
0.752425	-16.97906	0.668455	-3.936940

Source	DF	Type III SS	Mean Square	F Value	Pr > F
logmean	1	38.02407468	38.02407468	85.10	<.0001

Parameter	Estimate	Standard Error	t Value	Pr > |t|
Intercept	-2.595059242	0.18987984	-13.67	<.0001
logmean	0.510385427	0.05532755	9.22	<.0001

Based on this method, the suggested $\hat{\lambda} = 1 - \hat{\alpha} = 1 - 0.51 = 0.49 \approx 0.5$. Using this suggestion, we will perform a transformation to y by raising it to the power of 0.5, so $y^* = y^{.5} = \sqrt{y}$. Below is the Residual plot and Quantile plot of the transformed data.



We see that the funneling pattern is less obvious, even though there is still some funneling pattern there. The quantile plot also see a better fit of the dot to the line, which means that the transformed data meet the assumptions better. We do acknowledge that there seems to still be some violation, especially against the HoV assumption. We need to keep this in mind as we interpret the result of the ANOVA.

ANOVA Result of the Tranformed Data Below is the ANOVA table of the Transformed data, along with the estimates for the coefficients in the model.

TWO-FACTOR FIXED EFFECTS WITH BLOCK MODEL with SQUARE ROOT TRANSFORMATION

The GLM Procedure

Dependent Variable: sqrty

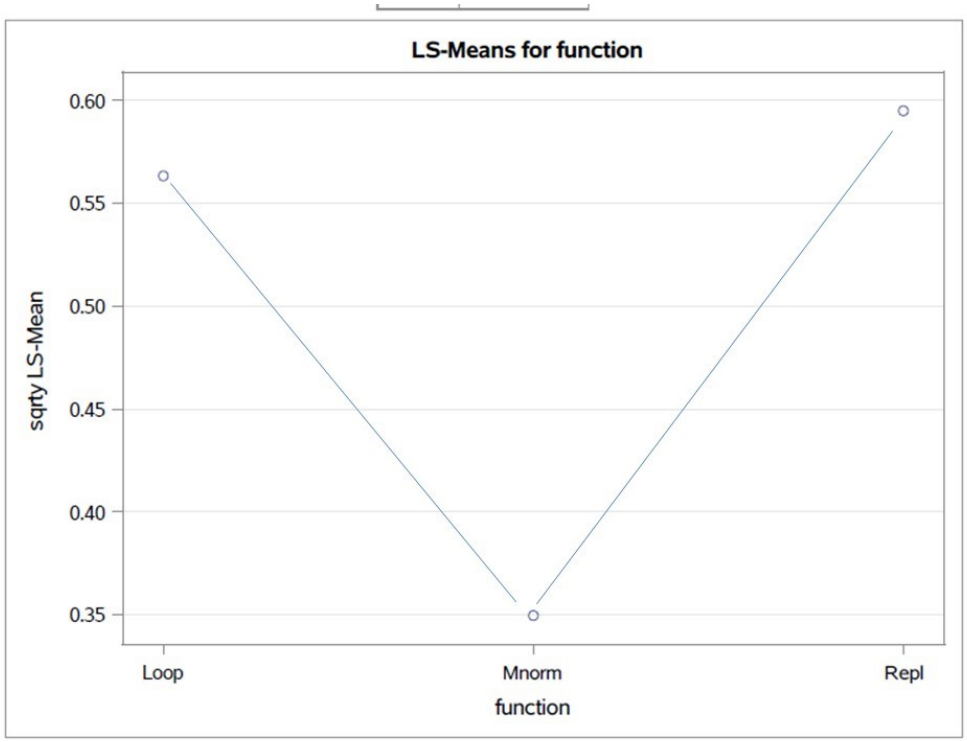
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	15	118.4372423	7.8958162	561.54	<.0001
Error	284	3.9933519	0.0140611		
Corrected Total	299	122.4305942			

R-Square	Coeff Var	Root MSE	sqrty Mean
0.967383	23.59362	0.118580	0.502591

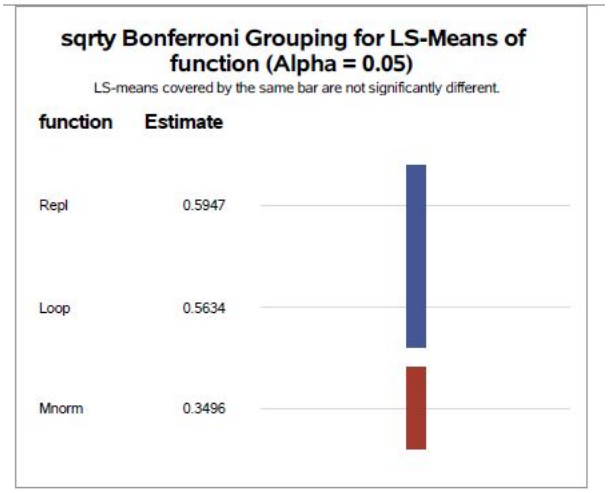
Source	DF	Type III SS	Mean Square	F Value	Pr > F
function	2	3.5576351	1.7788176	126.51	<.0001
simsize	4	107.1268840	26.7817210	1904.67	<.0001
function*simsize	8	5.6736617	0.7092077	50.44	<.0001
machine	1	2.0790615	2.0790615	147.86	<.0001

We see that the R-square shows a good fit of 97% of the model. The unaccounted variance is only about 3%. We also see significant effect of the parameters *function*, *simulation size*, and their interaction effect.

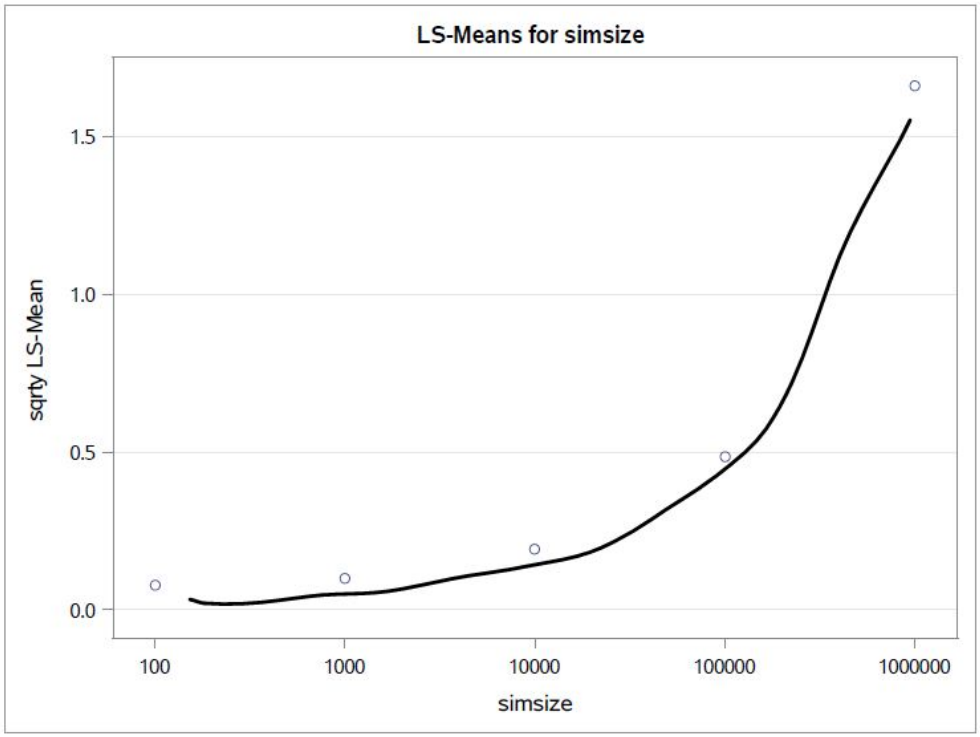
The plot below shows the estimated main effect of *function*.



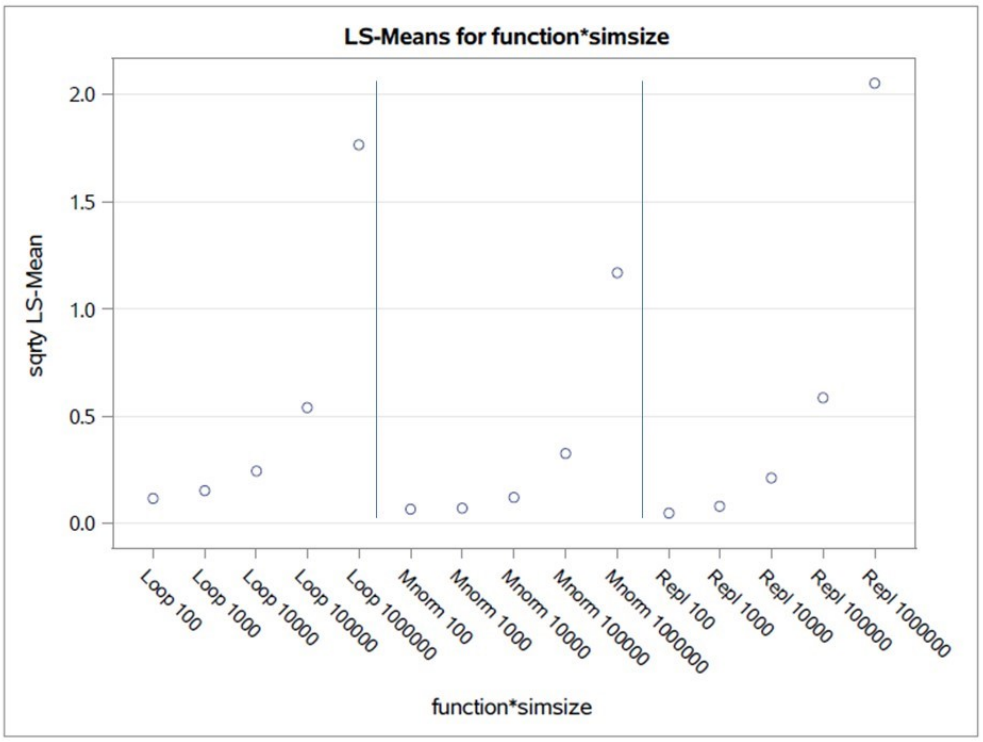
We see that *Mnorm* has the smallest elapsed time which puts it as the most efficient function. We also look at the Bonferroni multiple comparison test (output below), and we can conclude that *Mnorm* is siginificantly faster than the other two functions. The other two functions are not significantly different.



The effect of the simulation size, as expected, is that the elapsed time increases as the simulation size increases. However, we see that the increase is larger as the simulation size gets larger.



For the interaction effect, we see quite similar pattern, with the exception for *Mnorm 1000000*, which takes considerably less time compared to the other two functions for the same simulation size. We think that this is what causes the interaction effect to be significant.



The following table displays the estimated effect sizes of the parameters in the model.

Dependent Variable: sqrty

Parameter	Estimate	Standard Error	t Value	Pr > |t|
Function=Replicate	0.06082205	0.00968198	6.28	<.0001
Function=Loop	-0.15294137	0.00968198	-15.80	<.0001
Function=Rmnorm	0.09211932	0.00968198	9.51	<.0001
Sim Size = 100	-0.42602805	0.01369238	-31.11	<.0001
Sim Size = 1000	-0.40349378	0.01369238	-29.47	<.0001
Sim Size = 10000	-0.31105905	0.01369238	-22.72	<.0001
Sim Size = 100000	-0.01874302	0.01369238	-1.37	0.1721
Sim Size = 1000000	1.15932390	0.01369238	84.67	<.0001
Replicate 100	-0.02047729	0.01936395	-1.06	0.2912
Replicate 1000	-0.00955088	0.01936395	-0.49	0.6222
Replicate 10000	-0.00909144	0.01936395	-0.47	0.6391
Replicate 100000	-0.00469339	0.01936395	-0.24	0.8087
Replicate 1000000	0.04381299	0.01936395	2.26	0.0244
Loop 100	0.14294665	0.01936395	7.38	<.0001
Loop 1000	0.12127025	0.01936395	6.26	<.0001
Loop 10000	0.08322103	0.01936395	4.30	<.0001
Loop 100000	-0.00536154	0.01936395	-0.28	0.7821
Loop 1000000	-0.34207640	0.01936395	-17.67	<.0001
Rmnorm 100	-0.12246937	0.01936395	-6.32	<.0001
Rmnorm 1000	-0.11171937	0.01936395	-5.77	<.0001
Rmnorm 10000	-0.07412959	0.01936395	-3.83	0.0002
Rmnorm 100000	0.01005492	0.01936395	0.52	0.6040
Rmnorm 1000000	0.29826340	0.01936395	15.40	<.0001

The rows with the yellow highlight are the non-significant estimates.

Conclusion

This experiment shows that *Rmnorm* is a faster function to generate a random sample from a standard normal distribution. When the simulation size increases, this function is able to handle the increase with considerably less extended lapsed time. However, we need to note that there is a slight violation of the HoV assumption, so the result might not be as credible as we'd like.