International Journal of Assessment Tools in Education
2021, Vol. 8, No. 3, 704–713
https://doi.org/10.21449/ijate.804203
Published at https://ijate.net/ https://dergipark.org.tr/en/pub/ijate *Research Article*

MonteCarloSEM: An R Package to Simulate Data for SEM

Fatih Orcan [1,*]

[1]Trabzon University, Faculty of Education, Department of Educational Sciences, Turkey.

ARTICLE HISTORY

Received: Oct. 02, 2020
Revised: May 23, 2021
Accepted: July 11, 2021

Keywords:
Monte Carlo Simulation, Structural Equation Modeling, R package, Data generation

Abstract: Monte Carlo simulation is a useful tool for researchers to estimated accuracy of a statistical model. It is usually used for investigating parameter estimation procedure or violation of assumption for some given conditions. To run a simulation either the paid software or open source but free program such as R is need to be used. For that, researchers must have a good knowledge about the theoretical procedures. This paper introduces the R package called MonteCarloSEM. The package helps to simulate and analyze data sets for some simulation condition such as sample size and normality for a given model. Also, an example is given to show how the functions within the package works.

1. INTRODUCTION

Monte Carlo (MC) simulation studies are used to investigate the accuracy of statistical modeling in educational sciences as well as other social sciences. MC can be used to test such as violations of assumptions (Schumacker & Lomaz, 2010), effect of missing data or sample size on the model-data fit or parameter estimates. In the simplest terms, for example, we can examine how the t-test behaves if the small sample size is small (i.e., de Winter, 2013). Similarly, simulation studies are also used with structural equal modeling techniques. Boomsma (2013) pointed out that 31% of the studies published at the Structural Equation Modeling journal between 1994 and 2012 are MC studies. In order to run a MC study, either paid programs such as Mplus and Lisrel-PRELIS or open source but free program such as R (R Core Team, 2020) is need to be used. In both cases, it is necessary to know how the programs work.

Even though R is a free program it has a somewhat complex structure, especial for beginners. It is because all the coding must be done by the individual or a ready-made package needed to be used. In addition to the coding difficulties, the individual must also have a good knowledge about the theoretical procedures of simulation studies.

The "MonteCarloSEM" package has been developed to facilitate these operations and to enable researchers who are not familiar with such complex operations to perform MC simulation studies. In short, this package allows to test different conditions for a given Structural Equation Models (SEM) such as confirmatory factor analysis.

To run a MC simulation, a SEM model, the true model, must first be determined: number of factors in the model, the correlation between these factors and number of items loaded on each

*CONTACT: Fatih Orçan ✉ fatihorcan@trabzon.edu.tr Trabzon University, Faculty of Education, Department of Educational Sciences, Turkey

e-ISSN: 2148-7456 **/© IJATE** 2021

factor and so on. Then, a model which the data will be tested with is need to be determined. The testing model does not have to be the same with the true model. If it is not the same it is called misspecified model. A simulation study where the true model and misspecified model are used could give the effect of the misspecification on the parameter estimations and the model-data fits.

Different simulation conditions can be used for simulation studies. Number of factors, values of the factor loadings, the sample size and shape of the data (i.e., normal or non-normal) are some of them. This package enables us to use some these conditions. Detailed information about the contents of this package were given below. Each function and its parameters were explained with examples. Then, a simple simulation study is given as an example.

2. MonteCarloSEM PACKAGE

The package can be installed by using `install.packages("MonteCarloSEM")` comment. The documents for the package is available at the Comprehensive R Archive Network (CRAN): https://CRAN.R-project.org/package=MonteCarloSEM. The first version of the package was available as of September 22, 2020. The package requires to install some other packages. For example, the "Matrix" package (Maechler & Bates, 2006) is used for the matrix decomposition. The names of the required packages were given at the package's documents (Orçan, 2020).

There are five functions under this package. In case the arguments required for simulation are given, these functions enable us to a) generate artificial (i.e., simulated) data and b) analyze previously generated data and report the results.

2.1. Generating Artificial Data

The data sets were generated based on the standard normal distribution with mean of zero and standard deviation of one. Based on a given Confirmatory Factor Analysis (CFA) model, first uncorrelated factor scores were generated. Then, Cholesky decomposition method (Higham, 2009) was used to get correlated factor scores. On the other hand, independent error scores for each item were generated. Using correlated factor scores and random error scores, observed item scores were gained by the following formula (Orçan & Yanyun, 2016).

$$X_i = \lambda_i * F + \left(\sqrt{1 - {\lambda_i}^2}\right) * E_i$$

where X_i is an observed score on item i, F is factor score, λ_i and E_i indicates the factor loading and random error for item i, respectively.

If non-normality is desired, Fleishman's power transformation method (Fleishman, 1978) was applied to obtain scores with the predefined skewness and kurtosis values:

$$X_{skewed} = A + B * X_n + C * X_n^2 + D * X_n^3$$

where X_n is a previously generated normally distributed variable. The coefficients A, B, C and D are constants corresponding to a specific skewness and kurtosis values given at Fleishman's (1978) Table 1. For example, for skewness and kurtosis of one the values of B, C and D are 1.0174852, .190995 and -.018577, respectively.

2.2. Analyzing the Data

After the data sets were generated, the data sets can be tested with a given CFA model under "lavaan" package (Rosseel, 2012). To run the simulation under this package, first a lavaan model is needed to be defined. Definition of a lavaan model is not in the scope of this paper. Therefore, please see the lavaan documentation for the modeling strategies. Once the model is

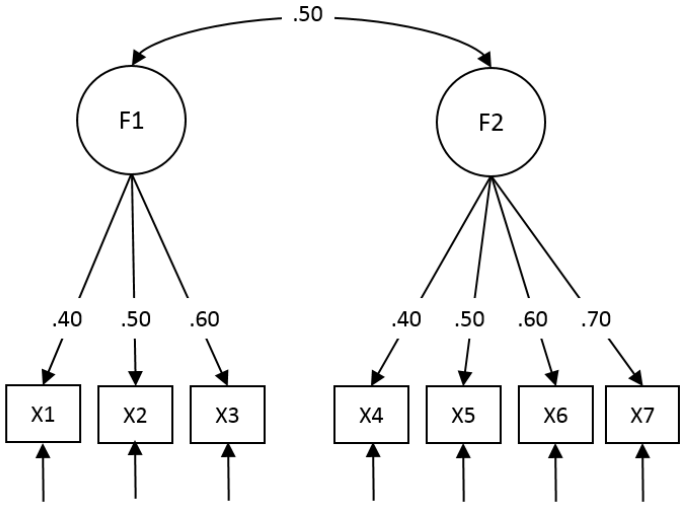
tested with the simulated data sets, the fit indices and the parameters estimated with their standard errors are printed to a Comma Separated Values (CSV) file.

The model given in Figure 1 was used to explain the MonteCarloSEM functions. As seen in Figure 1, there were two factors in the model (F1 and F2). The correlation between these factors was set to .5. Factors were defined with 3 and 4 items, respectively. The factor loadings of these items were indicated in the figure.

2.3. *fcors.value* Function

This function creates a symmetric matrix which specifies the correlation between the factors. If you are familiar with matrix formatting in R there is no need to use the function. Factor correlation matrix can be created by base *matrix* function. *fcors.value* function has two arguments: nf and cors.

Figure 1. *The model which the data simulated based on.*



- `nf`: It represents the number of factor/s in the data generation (true) model.
- `cors`: It represents the values of the correlation between the factors. Values of the correlation should be between -1 and +1. In case there is only one factor in the model, fallowing line should be entered "`cors.value(nf = 1, cors = c(1,1,1))`" to define factor correlation matrix.

For the model shown in Figure 1, the function should be:

```
fcors.value(nf = 2, cors = c(1, .5, .5, 1))
```

2.4. *loading.value* Function

The function specifies the factor loadings as a matrix. For each factor, the factor loadings values should be given by a column. That is, the columns represent the factors and rows represent the items. If there are k factors and m different items in the model the function creates a matrix of $k \times m$. This function also has two arguments: nf and fl.loads.

- `nf`: It represents the number of factor/s in the data generation (true) model. This value should be identical to nf argument at fcors.value function.
- `fl.loads`: This is a vector where all values of the loadings are given. The values entered should be larger than 0 and smaller then 1.

The values are assigned by columns. Therefore, firstly the values of loading for factor 1, starting from the first item to last, have to be given. For the model shown in Figure 1 the function should be:

```
loading.value(nf = 2, fl.loads = c(.4, .5, .6, 0, 0, 0 ,0,
                                   0, 0, 0, .4, .5, .6, .7))
```

2.5. *sim.normal* Function

This function generates normally distributed data sets by a given (i.e., true) CFA model. In each data file, the first column shows sample numbers. Starting from the second all other columns show actual simulated data for each item. For the model shown in Figure 1, for example, the column names colud be ID, F1X1, F1X2, F1X3, F2X4, F2X5, F2X6, F2X7. Besides the data set files, the function produces two more files. The first of them is "Model_Info.dat". This file shows the factor correlation and the factor loading matrices which were used for the data generation process. The second file is "Data_List.dat". The file includes names of the data files which were generated. *sim.normal* function has five arguments: nd, ss, fcors, loadings, f.loc.

- `nd`: It is an integer. It shows the number of data sets which will be generated. The default values for the arguments is 10.
- `ss`: It is an integer larger than 10. It indicates sample size for the data generation process. The default values for the arguments is 100.
- `fcors`: A symmetric factor correlation matrix. In order to define this argument fcors.value function would be used.
- `loadings`: It is the factor loading matrix. Columns represent factors and non-zero rows represent number of items under each factor. In order to define this argument loading.value function would be used.
- `f.loc`: It indicates where the simulated data sets will be saved. In order to run the function successfully, a file path has to be defined.

Let's now assume that we would like to generated 100 data sets by the model given in Figure 1. The sample size is 500 and the data sets will be saved *Documents* folder under C driver. Following codes do this simulation.

```
FCorr <- fcors.value(nf = 2, cors = c(1, .5, .5, 1))
FLoad <- loading.value(nf = 2, fl.loads = c(.4, .5, .6, 0, 0, 0 ,0,
                                            0, 0, 0, .4, .5, .6, .7))
FileLoc <- "C:/Users/user/Documents"
sim.normal(nd = 100, ss = 500, fcors = FCorr, loading = FLoad, f.loc = FileLoc)
```

2.6. *sim.skewed* Function

The function is similar to *sim.normal* function except that sim.skewed generates non-normally (skewed) distributed data sets by a given CFA model. The function generates data files, model information and data list files as *sim.normal*. However, the model information file has two more information under this function. One of them indicates if the items were non-normal or not. The second one shows Fleishman's B, C and D values which were used for data generation procedure. *sim.skewed* function has seven arguments: nd, ss, fcors, loadings, nonnormal, Fleishman and f.loc. The details about new arguments are given here. The others are the same as in *sim.normal* function.

- `nonnormal`: It is a vector of 0 or 1s. Each value in the vector represents an item in the model. If there are seven items in the model, for example, the length of the vector has to be seven. 0 indicates if the item is distributed normally while 1 indicates non-normal distribution for the corresponding item.

- `Fleishman`: It shows B, C and D values from Fleishman's power method. Since $A = -C$, the value of A is not needed here.

Again, let's assume that we would like to generated 100 data sets by the model given in Figure 1. Also, assume that only the item X6 and item X7 were non-normally distributed with skewness and kurtosis of 1. The sample size is 500 and the data sets will be saved *Documents* folder under C driver. Following codes do this simulation.

2.7. *fit.simulation* Function

sim.normal and *sim.skewed* functions generate data sets and save them into the specified folder. However, *fit.simulation* function uses these data sets and runs a CFA model, which does not have to be the same with the true model. Therefore, in order to run the function, the data sets have to be generated previously.

```
FCorr <- fcors.value(nf = 2, cors = c(1, .5, .5, 1))
FLoad <- loading.value(nf = 2, fl.loads = c(.4, .5, .6, 0, 0, 0 ,0,
                                            0, 0, 0, .4, .5, .6, .7))
IfNon <- c(0, 0, 0, 0, 0, 1, 1)
FleisV <- c(1.0174852, .190995, -.018577)
FileLoc <- "C:/Users/user/Documents"
sim.skewed(nd = 100, ss = 500, fcors = FCorr, loading = FLoad, nonnormal = IfNon,
   Fleishman = FleisV, f.loc = FileLoc)
```

The "lavaan" package (Rosseel, 2012) is used to fit the model to the data sets. Please see the lavaan documentation for more detailed information about how to build a CFA model. Once the model run is done, fit indices and parameters estimated with their standard errors are printed to a Comma Separated Values (CSV) file named as "All_Results.csv". Each line in the output file represents results of a simulated data set. The columns such as "chisq", "df", "cfi" or "rmsea" are self-explanatory but the second column named as "Notes". This column shows the model-data convergence. If the model is converged without any problem the value will be "CONVERGE". If it is not converged the value will be "NONCONVERGE" and all the other values in the row will be "NA". Lastly, if there were some kind of warning such as negative variance during the model run the value will be "WARNING" and based on the types of warnings, some of the values in the row might be "NA".

To run the simulation, the generated data sets and the list of the data sets' names (Data_List.dat) have to be located at the same folder. *fit.simulation* function has five arguments: model, PEmethod, datalist and f.loc.

- `model`: It indicates the lavaan model. The model will be used to test the generated data sets. Therefore, it does not have to be the same with data generation (true) model.
- `PEmethod`: It indicates the parameter estimation method. The default estimation method is Maximum Likelihood (ML). Other estimation methods such as Robust Maximum Likelihood (MLR) or Weighted Least Squares (WLS) are available under lavaan package. Please see the lavaan documentation for more information.
- `dataList`: It shows the name of the file which has list of the data sets' names. If *sim.normal* and *sim.skewed* functions were used for the data generation, as it was pointed earlier the name of the file is "Data_List.dat". However, data sets generated with other statistical programs or r-packages can also be used with this function. In that case, name of the file might be different. Therefore, it should be specified here.

- `f.loc`: It indicates where the simulated data sets are located. The dataList file and the simulated data sets have to be in this location.

Let's run a simulation based on previously generated data sets from *sim.skewed* function. For this, first a model needed to be defined by lavaan. The model given below (LavaanM) defines the model given in Figure 1.

```
LavaanM <- '
# CFA Model
f1 =~ NA*x1 + x2 + x3
f2 =~ NA*x4 + x5 + x6 + x7
# Factor Correlations
f1 ~~ f2
# Factor variance
f1 ~~ 1*f1
f2 ~~ 1*f2
'
DList <- "Data_List.dat"
FileLoc <- "C:/Users/user/Documents"
fit.simulation(model = LavaanM, PEmethod = "MLR", dataList = DList, f.loc =
   FileLoc)
```

Once the simulation process is completed the result of the simulation is saved as a CSV file to the location. A part of the CSV file was displayed in Figure 2. As it is seen form the figure, the first column showed replication number. For example, the results which were obtained from the first data set was given at the replication number 1. Based on the results, the first data set was converged. The chi-square value for the model was 6.351 with 13 degrees of freedom. The CSV file also shows fit indices such as the comparative fit indices (CFI) and the standardized root mean square residual (SRMR). Their values for the first data set were 1 and .018, respectively. The column W in the file shows the values of the factor loadings from item X1 to factor F1. The value for the first data was .403. Fallowing two columns (X and Y) shows standard errors of the estimates and the p-values. The standard error of the factor loading for the first data set was .061, for example. On the other hand, column Z (std.est) shows values of the standardized parameter estimates (.411).

Figure 2. *Sample view of "All_Results.csv" file.*

	A	B	C	D	E	M	O	W	X	Y	Z	AA	AB	AC
1	rep#	Notes	chisq	df	pvalue	cfi	srmr	f1=~x1	se	pvalue	std.est	std.se	pvalue	f1=~x2
2	1	CONVERGE	6.351	13	0.932	1	0.018	0.403	0.061	0	0.411	0.058	0	0.569
3	2	CONVERGE	11.928	13	0.534	1	0.025	0.449	0.057	0	0.459	0.055	0	0.534
4	3	CONVERGE	17.553	13	0.175	0.985	0.03	0.471	0.057	0	0.458	0.052	0	0.441
5	4	CONVERGE	11.047	13	0.607	1	0.025	0.343	0.063	0	0.34	0.059	0	0.493
6	5	CONVERGE	9.534	13	0.731	1	0.02	0.357	0.063	0	0.345	0.059	0	0.566
7	6	CONVERGE	24.218	13	0.029	0.969	0.034	0.413	0.063	0	0.421	0.059	0	0.446
8	7	CONVERGE	12.578	13	0.481	1	0.026	0.364	0.054	0	0.371	0.054	0	0.641
9	8	CONVERGE	13.15	13	0.436	1	0.024	0.417	0.057	0	0.44	0.055	0	0.424
10	9	CONVERGE	8.488	13	0.81	1	0.02	0.424	0.058	0	0.412	0.053	0	0.528
11	10	CONVERGE	13.048	13	0.444	1	0.023	0.421	0.061	0	0.422	0.058	0	0.529
12	11	CONVERGE	17.445	13	0.18	0.988	0.029	0.339	0.066	0	0.332	0.062	0	0.407
13	12	CONVERGE	10.255	13	0.673	1	0.023	0.438	0.063	0	0.438	0.059	0	0.519
14	13	CONVERGE	11.874	13	0.538	1	0.022	0.356	0.053	0	0.375	0.054	0	0.643

3. EXAMPLE SIMULATION STUDY

An example simulation is given here to demonstrated functions in the package. Let's assume that a researcher wanted to see the effects of nonnormality on the parameter estimation under a CFA model. For this purpose, a CFA model was defined which has three factor and each factor was loaded by three items. Factor loadings for the model were all equal to .7. Besides, the correlation among the factors were set to .5.

In order to create a manageable example only three conditions were simulated with sample size of 500 for 1000 times: All items were a) normally distributed (skewness = 0, kurtosis = 0), b) skewed (skewness = 1, kurtosis = 1) and c) skewed (skewness = 1.5, kurtosis = 2.5). The true model then used to get the parameter estimates under each conditions. Following codes simulate normal data sets at the step 1 and analyze the simulated data sets at the step 2.

```
# First, the package has to be installed.
install.packages("MonteCarloSEM")
library("MonteCarloSEM")

## Step 1: Simulate normally distributed data sets
# Define correlation among the factors

Ex.FCorr <- fcors.value(nf = 3, cors = c(1, .5, .5,
                                         .5, 1, .5,
                                         .5, .5, 1))
# Define factor loadings
Ex.FLoad <- loading.value(nf = 3, fl.loads = c(.7, .7, .7, 0, 0, 0, 0, 0, 0,
                                               0, 0, 0, .7, .7, .7, 0, 0, 0,
                                               0, 0, 0, 0, 0, 0, .7, .7, .7))
# Define where the simulated data sets will be saved
Ex.FileLoc <- "C:/Users/user/Documents"
sim.normal(nd = 1000, ss = 500, fcors = Ex.FCorr, loading = Ex.FLoad,  f.loc =
   Ex.FileLoc)

# Step 2: Analyze simulated data sets with the true model.

# Define CFA model under lavaan package
Ex.CorrectM <- '
# CFA Model
f1     =~ NA*x1 + x2 + x3
f2     =~ NA*x4 + x5 + x6
f3     =~ NA*x7 + x8 + x9
# Factor Correlations
f1     ~~ f2
f1     ~~ f3
f2     ~~ f3
# Factor variance
f1     ~~ 1*f1
f2     ~~ 1*f2
f3    ~~ 1*f3
'
```

```
# Define the folder which contains names of the simulated data sets
Ex.DList <- "Data_List.dat"

# Define where the data sets are located
Ex.FileLoc <- "C:/Users/user/Documents"

fit.simulation(model = Ex.CorrectM, PEmethod = "ML", dataList = Ex.DList, f.loc =
   Ex.FileLoc)
```

Similarly, following codes simulates skewed data (skewness =1 and kurtosis = 1) at step 1 and runs simulated data set at step 2. Since the same model was used for the simulations, step 2 is the same as normal data study above. Therefore, the same codes were used for the skewed data at step 2 and codes were not given again. Also, to run the second skewed data study the Ex.FleisV values should be replaced by .9920986, .3452694 and -.0418153 respectively. Later, the step 1 and the step 2 should be run again for the second skewed data simulation.

```
## Step 1: Simulate first skewed (non-normal) data sets.

# Define correlation among the factors
Ex.FCorr <- fcors.value(nf = 3, cors = c(1, .5, .5,
                                         .5, 1, .5,
                                         .5, .5, 1))

# Define factor loadings
Ex.FLoad <- loading.value(nf = 3, fl.loads = c(.7, .7, .7, 0, 0, 0, 0, 0, 0,
                                               0, 0, 0, .7, .7, .7, 0, 0, 0,
                                               0, 0, 0, 0, 0, 0, .7, .7, .7))

# Define which items are non-normal.
Ex.IfNon <- c(1, 1, 1, 1, 1, 1, 1, 1, 1)

# Define Fleishman's values: B, C and D
Ex.FleisV <- c(1.0174852, .190995, -.018577) # Values for Sk = 1, K = 1

# Note: Replace these values with following values for Sk = 1.5, K =
2.5: .9920986, .3452694 and -.0418153

# Define where the simulated data sets will be saved
Ex.FileLoc <- "C:/Users/user/Documents"

sim.skewed(nd = 1000, ss = 500, fcors = Ex.FCorr, loading = Ex.FLoad, nonnormal =
   Ex.IfNon, Fleishman = Ex.FleisV, f.loc = Ex.FileLoc)

# Step 2: Run simulated data sets with the correct model.
## This was the same as above. Therefore, use the same codes at step 2 given
   under normal data study.
```

4. RESULTS OF THE EXAMPLE

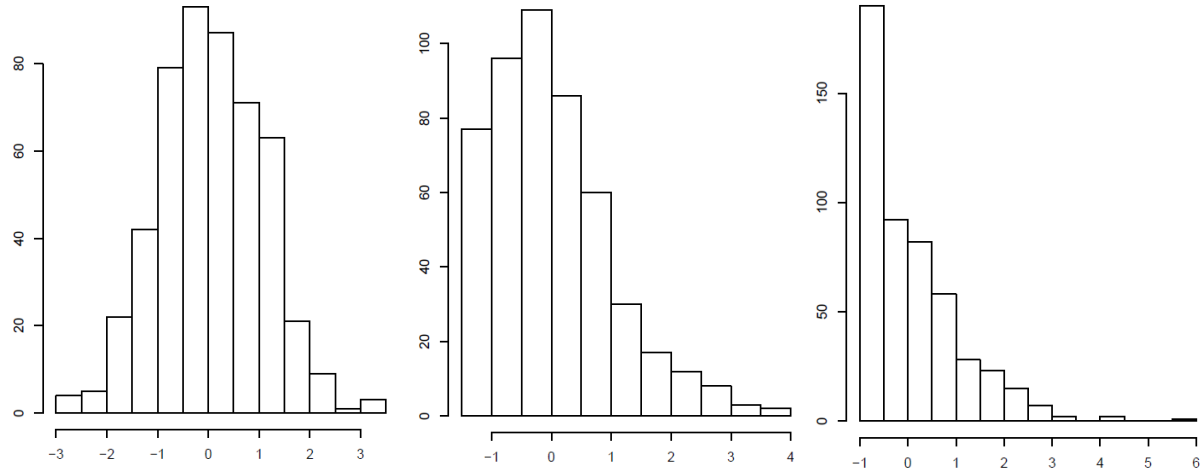
First, sample histograms for item scores generated under each simulation condition were pictured in Figure 3. The left panel shows an item scores under normal data generation while the right panel shows an item scores under skewness = 1.5 conditions. The actual skewness values of the specific items were .07, .99 and 1.46 respectively.

Table 1 summarizes the results of the simulation studies. The first row shows the average Chi-squares values. The second row shows that as the skewness increased the number of chi-square test which had p-values smaller than .05 were also increased. When the data were normal only 4.5% of the models showed no-model-data fit based on the chi-square tests. However, when the skewness increased to 1.5, the ratio increased to 17.4%. Since the factor loading had similar values only four of them were reported in table 1. Based on the results, as it was expected, the values of the parameter estimates were equal to the true values under the normally distributed data. However, as skewness increased the values of the parameter estimates got worse. Similar conclusions can also be made for the values of the correlations among factors. Therefore, as it was expected, as skewness increased the parameter estimated got worsen.

5. CONCLUSION

MonteCarloSEM package was introduced in this study. The package is useful for simulation data sets for a given model and analyzing the simulated data sets. There are five function in the package. This study described the functions and gave examples for the usage of the functions. Also, R codes for an example simulation study were provide in the study.

Figure 3. *The model which the data simulated based on.*



With the help of this package, researchers do not have to pay for expansive software to simulated data sets. Besides, the researchers can do simulation studies with little knowledge about R programing, considering simulating data for a given model is relatively complex and requires some knowledge about the formulas.

Table 1. *The Results of Example Simulation.*

		Normal	Skewness = 1	Skewness = 1.5
Chi-Square		23.84	26.85	29.01
Number of p <.05		45	113	174
Factor Loadings	Item 1	.70	.68	.65
	Item 2	.70	.68	.65
	Item 8	.70	.68	.65
	Item 9	.70	.68	.65
Factor Correlations	Factor 1 and 2	.50	.49	.47
	Factor 1 and 3	.50	.49	.46
	Factor 2 and 3	.50	.49	.46

The first version of the package is available now at CRAN (https://github.com/cran/MonteCarloSEM). The package has some limitations for now. However, some new functions are under construction now. For example, it is planned to add a function which creates missing data based on missing completely random (MCAR), missing at random (MAR) and missing not at random (MNAR) to the package for the later versions. Besides this, functions which creates item parcels and categorical (i.e., Likert) data sets were also under construction.

Declaration of Conflicting Interests and Ethics

The author declares no conflict of interest. This research study complies with research publishing ethics. The scientific and legal responsibility for manuscripts published in IJATE belongs to the author.

ORCID

Fatih Orçan https://orcid.org/0000-0003-1727-0456

6. REFERENCES

Boomsma, A. (2013) Reporting Monte Carlo Studies in Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, *20*(3), 518-540. https://doi.org/10.1080/10705511.2013.797839

de Winter, J.C.F. (2013). Using the Student's t-test with extremely small sample sizes. *Practical Assessment, Research, and Evaluation, 18*, 10. https://doi.org/10.7275/e4r6-dj05

Fleishman, A. I. (1978). A method for simulating non-normal distributions. *Psychometrika, 43,* 521-532. https://doi.org/10.1007/BF02293811

Higham, N.J. (2009). Cholesky factorization. *WIREs Computational Statistics, 1,* 251-254.

Maechler, M., & Bates, D. (2006). *2nd Introduction to the Matrix package.* URL: https://cran.r-project.org/web/packages/Matrix/vignettes/Intro2Matrix.pdf

Orçan, F. & Yanyun, Y. (2016). A Note on the Use of Item Parceling in Structural Equation Modeling with Missing Data. *Journal of Measurement and Evaluation in Education and Psychology, 7* (1), 59-72. https://doi.org/10.21031/epod.88204

Orçan, F. (2020). MonteCarloSEM 0.0.1. https://CRAN.R-project.org/package=MonteCarloSEM

R Core Team (2020). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. Retrieved September 10, 2020, from http://www.R-project.org/

Rosseel, Y. (2012). lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software, 48* (2), 1-36. URL: http://www.jstatsoft.org/v48/i02/

Schumacker, R. E., & Lomax, R. G. (2010). A beginner's guide to structural equation modeling (3rd ed.). Routledge.