

Monte Carlo Simulation Studies in Item Response Theory with the R Programming Language

R Programlama Dili ile Madde Tepki Kuramında Monte Carlo Simülasyon Çalışmaları

Okan BULUT*

Önder SÜN BÜL**

Abstract

Monte Carlo simulation studies play an important role in operational and academic research in educational measurement and psychometrics. Item response theory (IRT) is a psychometric area in which researchers and practitioners often use Monte Carlo simulations to address various research questions. Over the past decade, R has been one of the most widely used programming languages in Monte Carlo studies. R is a free, open-source programming language for statistical computing and data visualization. Many user-created packages in R allow researchers to conduct various IRT analyses (e.g., item parameter estimation, ability estimation, and differential item functioning) and expand these analyses to comprehensive simulation scenarios where the researchers can investigate their specific research questions. This study aims to introduce R and demonstrate the design and implementation of Monte Carlo simulation studies using the R programming language. Three IRT-related Monte Carlo simulation studies are presented. Each simulation study involved a Monte Carlo simulation function based on the R programming language. The design and execution of the R commands is explained in the context of each simulation study.

Key Words: Psychometrics, measurement, IRT, simulation, R.

Öz

Eğitimde ölçme ve psikometri alanlarında yapılan akademik ve uygulamaya dönük araştırmalarda Monte Carlo simülasyon çalışmaları önemli bir rol oynamaktadır. Psikometrik çalışmalarda araştırmacıların Monte Carlo simülasyonlarına sıklıkla başvurduğu temel konulardan birisi Madde Tepki Kuramı'dır (MTK). Geçtiğimiz son on yılda MTK ile ilgili yapılan simülasyon çalışmalarında R'ın sıklıkla kullanıldığı görülmektedir. R istatistiksel hesaplama ve görsel üretme için kullanılan ücretsiz ve açık kaynak bir programlama dilidir. R kullanıcıları tarafından üretilen birçok paket program ile madde parametrelerini kestirme, madde yanlılık analizleri gibi birçok MTK temelli analiz yapılabilmektedir. Bu çalışma, R programına dair giriş niteliğinde bilgiler vermek ve R programlama dili MTK temelli Monte Carlo simülasyon çalışmalarının nasıl yapılabileceğini göstermeyi amaçlamaktadır. R programlama dilini örneklerle açıklamak için üç farklı Monte Carlo simülasyon çalışması gösterilmektedir. Her bir çalışmada, simülasyon içerisindeki R komutları ve fonksiyonları MTK kapsamında açıklanmaktadır.

Anahtar Kelimeler: Psikometri, ölçme, MTK, simülasyon, R.

INTRODUCTION

Monte Carlo simulation studies are the key elements of operational and academic research in educational measurement and psychometrics. Both academic researchers and psychometricians often choose to simulate data instead of collecting empirical data because (a) it is impractical and costly to collect the empirical data while manipulating several conditions (e.g., sample size, test length, and test characteristics); (b) it is not possible to investigate the real impact of the study conditions without knowing the true characteristics of the items and examinees (e.g., item parameters, examinee ability

* Assistant Professor, University of Alberta, Faculty of Education, Edmonton, Alberta – Canada, e-mail: bulut@ualberta.ca, ORCID ID: orcid.org/0000-0001-5853-1267

** Assistant Professor, Mersin University, Faculty of Education, Mersin – Turkey, e-mail: ondersunbul@mersin.edu.tr, ORCID ID: orcid.org/0000-0002-1775-1404

distributions); and (c) the empirical data are often incomplete, which may affect the outcomes of the study, especially when the amount of missing data is large and the pattern of missingness is not random (Brown, 2006; Feinberg & Rubright, 2016; Robitzsch & Rupp, 2009; Sinharay, Stern, & Russell, 2001). Furthermore, when conducting psychometric studies, it is impossible to eliminate the effects of potential confounding variables related to examinees (e.g., gender, attitudes, and motivation) and test items (e.g., content, linguistic complexity, and cognitive complexity). The growing number of research articles, books, and technical reports as well as unpublished resources (e.g., conference presentations) involving simulations also depict the importance of Monte Carlo simulations in the field of educational measurement.

Item response theory (IRT) is one of the most popular research areas in educational measurement and psychometrics. Both researchers and practitioners often use Monte Carlo simulation studies to investigate a wide range of research questions in the context of IRT (Feinberg & Rubright, 2016). Monte Carlo simulation studies are often used for evaluating how validly IRT-based methods can be applied to empirical data sets with different kinds of measurement problems (Harwell, Stone, Hsu, & Kirisci, 1996). To be able to conduct Monte Carlo simulation studies in IRT, there is a large number of psychometric software packages (e.g., IRTPRO [Cai, Thissen, & du Toit, 2011], flexMIRT [Cai, 2013], BMIRT [Yao, 2003], and Mplus [Muthén & Muthén, 1998-2015]) and programming languages (e.g., C++, Java, Python, and Fortran) available to the researchers. For some researchers, the psychometric software packages can be more suitable when conducting simulation studies in IRT because most of these packages often provide built-in functions to simulate and analyze the data. However, such psychometric software packages are often not free, only capable of particular types of IRT analyses, and generally slow when running large, computation-intensive simulations. For other researchers, the programming languages (e.g., C++ and Java) can be more tempting due the speed and flexibility, although these programming languages require intermediate to advanced programming skills to design and implement Monte Carlo simulation studies. Therefore, researchers often prefer general statistical packages – such as SAS (SAS Institute Inc., 2014), Stata (StataCorp, 2015), and R (R Core Team, 2017), which are not only more flexible and faster than the psychometric software packages but also require relatively less knowledge of programming. Among these statistical packages, R has been particularly popular because it is free, flexible, and capable of various statistical analyses and data visualizations.

Despite the increasing use of the R programming language for conducting various statistical and psychometric analyses, many researchers are still unfamiliar with the capabilities of R for conducting Monte Carlo simulation studies. As Hallgren (2013) pointed out, the use of simulation studies should be available to researchers with a broad range of research expertise and technical skills. Researchers should be familiar with how to address research questions that simulations can answer best (Feinberg & Rubright, 2016). Given the growing demand for IRT-related research in the field of educational measurement, this study aims to demonstrate how to use R for the design and implementation of Monte Carlo simulation studies in IRT, specifically for individuals with minimal experience in running simulation studies in R. The purposes of this study are threefold. First, we introduce readers the packages and functions in R for simulating response data and analyzing the simulated data using various IRT models. Second, we summarize the principles of Monte Carlo simulations and recommend some guidelines for conducting Monte Carlo simulation studies. Third, we illustrate the logic and procedures involved in conducting IRT-related Monte Carlo simulation studies in R with three examples – including the R codes for simulating item response data, analyzing the simulated data, and summarizing the analysis results. The examples will target three different uses of Monte Carlo simulation studies in IRT, including item parameter recovery, evaluating the accuracy of a method for detecting differential item functioning (DIF), and investigating the unidimensionality assumption. Each simulation study uses different criteria to evaluate the simulation results (e.g., accuracy, power, and Type I error rate). For the sake of simplicity and conciseness, the readers of this study are assumed knowledgeable about (1) the basics of the R programming language and (2) the fundamentals of IRT. The readers who are not familiar with the Monte Carlo simulation studies in IRT are referred to Harwell et al. (1996) and Feinberg and Rubright (2016) for a comprehensive review. In addition, the

readers are referred to the R user manuals (<https://cran.r-project.org/doc/manuals/>) for a detailed introduction to the R programming language.

Some Functions to Simulate Data in R

One of the greatest advantages of R is the ability to generate variables and data sets using various probability distributions (e.g., standard normal (Gaussian) distribution, uniform distribution, and the Bernoulli distribution). This section will provide a brief summary of the probability distributions in R that are commonly used when simulating data for the IRT simulation studies. The names of the functions for generating data in R typically begin with “d” (density function), “p” (cumulative probability function), “q” (quantile function), or “r” (random sample function). The latter part of the function represents the type of the distribution. For example, the `rnorm` function generates random data with a normal distribution, while the `runif` function generates random data with a uniform distribution. The common distributions for continuous and categorical data include `exp` for the exponential distribution, `norm` for the normal distribution, `unif` for the uniform distribution, `binom` for the binomial distribution, `beta` for the beta distribution, `lnorm` for the log-normal distribution, `logis` for the logistic distribution, and `geom` for the geometric distribution.

When generating random samples using the R functions mentioned above, the randomization of the generated values is systematically controlled based on the random number generator (RNG). The RNG algorithm assigns a particular integer to each random sample. In the R programming language, the integer associated with random samples is called “seed”. The user can select a particular seed when generating a random sample and then use the same seed again whenever the same random sample needs to be obtained. The `set.seed` function can be used for selecting a particular seed in R (e.g., `set.seed(1111)`, where 1111 is the seed). The `set.seed` function plays an important role in simulations studies because it allows the researcher to create a reproducible simulation.

R Packages for Estimating IRT Models

R has many user-created psychometric packages that allow researchers and practitioners to conduct statistical analysis using psychometric models and methods. The CRAN website has a directory of the R packages categorized by topic, which is called “Task View”. One of these task views, the Psychometric Task View, is specifically dedicated to psychometric methods (see <https://cran.r-project.org/web/views/Psychometrics.html>), such as IRT, classical test theory, factor analysis, and structural equation modeling. A lot of the packages in the Psychometrics Task View focus on the estimation of IRT models, such as unidimensional and multidimensional IRT models, nonparametric IRT modeling, differential item functioning, and computerized adaptive testing (see Ünlü and Yanagida [2011] for a review of the CRAN Psychometrics Task View). Rusch, Mair, and Hatzinger (2013) also provided a detailed summary of the R packages for conducting IRT analysis. The primary R packages for estimating item parameters and person abilities in IRT include `mirt` (Chalmers, 2012), `eRm` (Mair, Hatzinger, & Maier, 2016), `irtoys` (Partchev, 2016), and `ltm` (Rizopoulos, 2006). There are also specific packages for a particular IRT analysis – such as `lordif` (Choi, Gibbons, & Crane, 2016) and `difR` (Magis, Beland, Tuerlinckx, & De Boeck, 2010) for differential item functioning; `catR` (Magis & Raiche, 2012) and `mirtCAT` (Chalmers, 2016) for computerize adaptive testing; and `equate` (Albano, 2016) for test equating. In addition to these packages, we encourage the readers to browse through the Psychometric Task View for other types of IRT methods available in R.

Guidelines for Conducting Monte Carlo Simulation Studies

Monte Carlo simulation studies can be used for investigating a wide range of research questions, such as evaluating the accuracy of existing statistical models under unfavorable conditions (e.g., small sample and non-normality), answering a novel statistical question, or understanding the empirical

distribution of a particular statistic through bootstrapping (Feinberg & Rubright, 2016; Hallgren, 2013; Harwell et al., 1996). A Monte Carlo simulation study typically consists of the following steps:

1. The researcher determines a set of simulation factors expected to influence the operation of a particular statistical procedure. The simulation factors can be either fully crossed or partially crossed. If the simulation factors are fully crossed, then a data set needs to be generated for every possible combination of the simulation factors. If, however, they are partially crossed, only some simulation factors are assumed to be interacting with each other.
2. A series of assumptions are made about the nature of data to be generated (e.g., types of variables and probability distributions underlying the selected variables). These assumptions are crucial to the authenticity of the simulation study because the quality of the simulation outcomes depends on the extent to which the selected assumptions are realistic.
3. Multiple data sets are generated based on the simulation factors and the assumptions about the nature of the data. The process of generating multiple data sets is often called replication. Monte Carlo simulation studies often involve multiple replications (a) to acquire the sampling distribution of parameter estimates, (b) to reduce the chance of obtaining implausible results from a single data set, and (c) to have the option to resample the true parameters based on the assumption made in Step 2.
4. Statistical analyses are performed on the simulated data sets and the parameter estimates of interest from these analyses are recorded. The parameter estimates can be p-values, coefficients, or a particular element of the statistical model.
5. Finally, the estimated parameters are evaluated based on a criterion or a set of criteria – such as Type I error, power (or hit rate), correlation, bias, and root mean squared error (RMSE). The researcher can report the findings of the simulation study in different ways (e.g., a narrative format, tables, or graphics). The researcher should determine how the simulation results will be communicated to the target audience based on the size of the simulation study (i.e., the number of simulation factors), the complexity of the simulation design (e.g., several fully-crossed simulation factors or a simpler design with one or two factors), and the type of the reporting outlet (e.g., technical reports, journal articles, or presentations).

It should be noted that the five steps summarized above could be slightly different for each simulation study, depending on the research questions that need to be addressed.

Principles of Monte Carlo Simulation Studies

Apart from the guidelines summarized above, there are also three principles that the researchers need to consider when designing and conducting a Monte Carlo simulation study. These principles are authenticity, feasibility, and reproducibility.

The *authenticity* of a Monte Carlo simulation study refers to the degree to which the simulation study reflects the real conditions. For example, assume that a researcher wants to investigate the impact of test length on ability estimates obtained from a particular IRT model. The researcher selects 30, 60, 90, and 300 items as the hypothesized values for the test length factor. Because a 300-item test is quite unlikely to occur in real life, the researcher should probably consider eliminating this option from the simulation study. The authenticity of a Monte Carlo simulation study is also related to the necessity of the simulation factors. Continuing with the same example, the researcher might consider sample size as a potential factor for the simulation study on the recovery of ability estimates; but sample size is known to have no effect on the estimation of ability (or latent trait) when item parameters are already known (e.g., Bulut, 2013; Bulut, Davison, & Rodriguez, 2017). Therefore, the researcher would not need to include sample size as a simulation factor in the study.

The *feasibility* of a Monte Carlo study refers to the balance between the goals of the simulation study and the scope of the simulation study. The combination of many simulation factors and a high number of replications may often lead to a highly complex simulation study that is hard to complete and

summarize within a reasonable period. Therefore, the researcher should determine which simulation factors are essential and how many replications can be accomplished based on the scope of the simulation study. For example, assume that a researcher plans to use test length (10, 20, 40, or 60 items), sample size (100, 250, 500, or 1000 examinees), ability distribution (normal, positively skewed, or negatively skewed), and inter-dimensional correlation ($r = .2$, $r = .5$, or $r = .7$) as the simulation factors. If the simulation factors were fully crossed, then there would be $4 \times 4 \times 3 \times 3 = 144$ cells in the simulation design. If the researcher conducted 10,000 replications for each cell, the entire simulation process would result in 1,440,000 unique data sets that need to be analyzed and summarized. Depending on the complexity of the statistical analysis, this simulation study might take several weeks (or possibly months) to complete, even with the parallel computing feature available in R and other statistical software programs.

The *reproducibility* of a Monte Carlo simulation study refers to the likelihood that the researcher who conducted the simulation study can replicate the same findings at a later time, or that other researchers who have access to the simulation parameters can replicate (or at least approximate) the findings. To ensure reproducibility, the researcher should specify the seed before generating data and store a record of the selected seeds in the simulation study. The researcher can either use the seeds to replicate the findings later on or give them to other researchers interested in replicating their findings (Feinberg & Rubright, 2016). However, it should be noted that even using the same seeds might not guarantee that identical simulation results will be obtained because the mechanism of the random number generator can differ from one computer to another, or across the different versions of the same software program.

METHOD

The following sections of this study will demonstrate three Monte Carlo simulation studies about item parameter recovery, differential item functioning, dimensionality in IRT-based assessment forms. Each study focuses on a set of research questions in the context of IRT and aims to address the research questions through a Monte Carlo simulation study. Each study consists of three steps: data generation, statistical analysis, and summarizing the simulation results. The implementation of these steps will be demonstrated using R. The readers are strongly encouraged to run the examples in their own computers by copying and pasting the provided R codes into the R console. It should be noted that most R packages are regularly updated by their creators and/or maintainers, and thus the R functions presented in this study are subject to change in the future. Therefore, we recommend the readers to check out the R packages used in this study before using the Monte Carlo simulation functions. For this study, we use the latest version of Microsoft R Open (version 3.4.0). The readers are strongly encouraged to use this particular version of Microsoft R Open to ensure the reproducibility of the Monte Carlo studies presented in the following sections.

Simulation Study 1: Item Parameter Recovery in IRT

In this study, we aim to investigate to what extent the accuracy of estimated item parameters in the unidimensional three-parameter logistic (3PL) IRT model depends on the number of examinees who respond to the items (sample size) and the number of items (test length). In addition, we want to find out which item parameter (item difficulty, item discrimination, and guessing) is the most robust against changes in sample size and test length. To address these research questions, we design a small-scale Monte Carlo simulation study in which sample size and test length are the two simulation factors. The simulation study will be based on a fully crossed design with three sample sizes (500, 1000, or 2000 examinees) and three test lengths (10, 20, or 40 items), resulting in $3 \times 3 = 9$ cells in total. For each cell, 100 replications will be conducted with unique item parameters and person abilities in each replication. For the evaluation of the recovery of true item parameters, we use bias and RMSE:

$$Bias = \frac{\sum_{i=1}^K (\hat{X}_i - X_i)}{K}, \text{ and} \quad (1)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^K (\hat{X}_i - X_i)^2}{K}}, \quad (2)$$

where K is the total test length, \hat{X}_i is the estimated item parameter for item i ($i = 1, 2, \dots, K$), and X_i is the true item parameter for item i . The average bias and RMSE values over 100 replications will be reported for each of the nine simulation cells.

Data generation

The item response data will be simulated using the 3PL model. The mathematical formulation of the 3PL model can be shown as follows:

$$P_i(\theta) = c_i + (1 - c_i) \frac{e^{Da_i(\theta - b_i)}}{1 + e^{Da_i(\theta - b_i)}}, \quad (3)$$

where $P_i(\theta)$ is the probability of an examinee with the ability of θ responding to item i correctly, b_i is the item difficulty parameter for item i , a_i is the item discrimination parameter for item i , c_i is the lower asymptote (also known as the pseudo-guessing parameter), e is the base of the natural logarithm approximated at 2.718, and D is a constant of 1.7 to transform the logistic IRT scale into the normal ogive scale (Camilli, 1994; Crocker & Algina, 1986).

Following the suggestions from previous studies regarding data simulation for the 3PL model (e.g., Harwell & Baker, 1991; Feinberg & Rubright, 2016; Mislevy & Stocking, 1989; Mooney, 1997), the item difficulty parameters are drawn from a normal distribution, $b \sim N(0, 1)$; the item discrimination parameters are drawn from a log-normal distribution, $a \sim \ln N(0.3, 0.2)$; and the lower asymptote parameters are drawn from a beta distribution, $c \sim \text{Beta}(20, 90)$. Furthermore, the ability parameters are drawn from a standard normal distribution, $\theta \sim N(0, 1)$. For simulating dichotomous item responses and estimating the item parameters based on the 3PL model, we will use the `mirt` package (Chalmers, 2012) in R. To install and activate the `mirt` package, the following commands should be first run:

```
install.packages("mirt")
library("mirt")
```

Then, we define a simulation function called `itemrecovery`, which generates item parameters, simulates dichotomous item response data using the generated item parameters, estimates the item parameters of the 3PL model for the simulated data, and finally computes bias and RMSE values for each set of estimated item parameters:

```
itemrecovery <- function(nitem, sample.size, seed) {

  #Set the seed and generate the parameters
  set.seed(seed)
  a <- as.matrix(round(rlnorm(nitem, meanlog = 0.3, sdlog = 0.2),3), ncol=1)
  b <- as.matrix(round(rnorm(nitem, mean = 0, sd = 1),3), ncol=1)
  c <- as.matrix(round(rbeta(nitem, shape1 = 20, shape2 = 90),3), ncol=1)
  ability <- as.matrix(round(rnorm(sample.size, mean = 0, sd = 1),3), ncol=1)

  #Simulate response data and estimate item parameters
  dat <- simdata(a = a, d = b, N = sample.size, itemtype = 'dich',
                guess = c, Theta = ability)
  model3PL <- mirt(data=dat, 1, itemtype='3PL', SE=TRUE, verbose=FALSE)

  #Extract estimated item parameters and compute bias and RMSE
  parameters <- as.data.frame(coef(model3PL, simplify=TRUE)$items)
```

```

bias.a <- round(mean(parameters[,1]-a), 3)
bias.b <- round(mean(parameters[,2]-b), 3)
bias.c <- round(mean(parameters[,3]-c), 3)
rmse.a <- round(sqrt(mean((parameters[,1]-a)^2)), 3)
rmse.b <- round(sqrt(mean((parameters[,2]-b)^2)), 3)
rmse.c <- round(sqrt(mean((parameters[,3]-c)^2)), 3)

#Combine the results in a single data set
result <- data.frame(sample.size=sample.size, nitem=nitem,
                    bias.a=bias.a, bias.b=bias.b, bias.c=bias.c,
                    rmse.a=rmse.a, rmse.b=rmse.b, rmse.c=rmse.c)

return(result)
}

```

In the `itemrecovery` function, there are three input values that need to be specified by the researcher: `nitem` as the number of items (i.e., test length), `sample.size` as the number of examinees (i.e., sample size), and `seed` as the seed for the random number generator. The `itemrecovery` function begins with setting the seed for the random values that are going to be generated, which will ensure reproducibility of the simulated data. Next, the item parameters are randomly generated based on the distribution characteristics explained earlier using the `rlnorm`, `rnorm`, and `rbeta` functions. Each set of the generated parameters (called `a`, `b`, and `c`) is saved as a matrix with a single column. The `simdata` function from the `mirt` package simulates dichotomous item responses according to the 3PL model using the generated item parameters. More details about the `simdata` function can be obtained by running the `?simdata` command in the R console. Then, we estimate item parameters using the `mirt` function, extract the estimated item parameters from the model using the `coef` function, and save the parameters in a data frame called `parameters`. More details about the estimation process in the `mirt` function can be obtained by running the `?mirt` command in the R console. At the end of the function, we compute the bias and RMSE values for each item parameter, save the values into a data set called `result`, and return the `result` data set as the outcome of the simulation. To enable the `itemrecovery` function, we can either copy and paste the entire function into the R console and hit the “enter” button, or select all the entire function in the R script file, right-click on the selected lines, and choose “Run line or selection” to execute the commands in the R console.

The next step is to conduct the simulation study using the `itemrecovery` function. First, we randomly generate 100 integers to be used as the random seeds in the study. We sample random integers ranging from 0 to 1,000,000 using the `sample.int` function and store the generated values in a data set called `myseed`. We export the seeds into a text file called “simulation seeds.txt”. This file will be saved in the current working directory designated by the user. To save the document in a specific folder, a complete folder path should be provided, such as “C:/Users/username/Desktop/simulation seeds.txt”. Note that when defining a folder path, forward slash (/) should be used instead of a backslash (\). Next, we define an empty data set (i.e., `result`) that will store the simulation results out of 100 replications. The final step of the simulation study is to run the simulation within a loop and save the results into the `result` data set. `for (i in 1:length(myseed)) { }` creates a loop to run a procedure 100 times (i.e., the same length of `myseed`). In this study, we want to run the `itemrecovery` function 100 times using a different seed from `myseed` for each replication. Once all the iterations are complete, the `result` data set will consist of one hundred rows (one row per iteration). At the end, we use the `colMeans` function to find the average bias and RMSE values across 100 replications. Because we use the `round(colMeans(result), 3)`, all of the values will be rounded off to three decimal digits.

```

#Generate 100 random integers
myseed <- sample.int(n = 1000000, size = 100)
write.csv(myseed, "simulation seeds.txt", row.names = FALSE)

```

```
#Define an empty data frame to store the simulation results
result <- data.frame(sample.size=0, nitem=0, bias.a=0, bias.b=0, bias.c=0,
                    rmse.a=0, rmse.b=0, rmse.c=0)

#Run the loop and return the results across 100 iterations
for (i in 1:length(myseed)) {
  result[i,] <- itemrecovery(nitem = 20, sample.size = 1000, seed = myseed[i])
}
round(colMeans(result), 3)
```

For each of the nine cells planned for this simulation study, we set the `nitem` and `sample.size` values accordingly and re-run the R script presented above. When the simulations for all of the cells are complete (i.e., the R script has been run nine times in total), we report the results shown in Table 1. The simulation results show that the lower asymptote (i.e., guessing) parameter had the smallest bias and RMSE, whereas the item discrimination parameter indicated the largest bias and RMSE. As sample size and test length increased, bias and RMSE decreased for all of the item parameters.

Table 1. Simulation Results from the Item Parameter Recovery Study

Sample Size	Test Length	Bias a	Bias b	Bias c	RMSE a	RMSE b	RMSE c
500	10	0.375	-0.250	-0.004	0.988	0.931	0.160
	20	0.189	-0.139	-0.003	0.603	0.658	0.144
	40	0.140	-0.109	-0.001	0.480	0.555	0.132
1000	10	0.140	-0.091	-0.012	0.515	0.589	0.134
	20	0.076	-0.046	-0.004	0.357	0.428	0.122
	40	0.070	-0.063	0.001	0.299	0.384	0.109
2000	10	0.068	-0.033	-0.010	0.310	0.376	0.112
	20	0.043	-0.031	-0.006	0.241	0.302	0.092
	40	0.031	-0.020	-0.004	0.197	0.264	0.085

In addition to Table 1, we can also present the simulation results graphically using the `lattice` package (Sarkar, 2008) in R. First, we manually enter the simulation results in Table 1 into an empty Excel spreadsheet using a long format and save the spreadsheet with a `.csv` extension using the “Save As” option under the “File” menu. The saved file is called “simulation results.csv”. Figure 1 shows a screenshot of the “simulation.csv” file.

	A	B	C	D	E
1	SampleSize	TestLength	Parameter	Bias	RMSE
2	500	10	a	0.375	0.988
3	500	20	a	0.189	0.603
4	500	40	a	0.14	0.48
5	1000	10	a	0.14	0.515
6	1000	20	a	0.076	0.357
7	1000	40	a	0.07	0.299
8	2000	10	a	0.068	0.31
9	2000	20	a	0.043	0.241
10	2000	40	a	0.031	0.197

Figure 1. A Screenshot of the First Nine Rows of the “simulation results.csv” File

After we read the data set "simulation results.csv" in R, we define the two variables (SampleSize and TestLength) as categorical variables by using the `as.factor` function. Next, we install the `lattice` package and then activate it using the `library` command. Finally, we use the `xyplot` function in the `lattice` package (Sarkar, 2008) to create an interaction plot. This plot will demonstrate the relationship between bias, RMSE, sample size, and test length for each item parameter. In the `xyplot` function, we first select the variable for the y axis (either bias or RMSE) and the variable for the x axis (test length), the variable that defines multiple panels (sample size), and the group variable (parameters). In addition, `xlab` defines the label for the x axis, `type = "a"` indicates an interaction plot, the elements in `auto.key` define the position of the legend, whether or not data points should be shown, whether or not lines should be shown, and the number of columns for the legend, and the elements in `par.settings` defines the colours (`lty`) and thickness (`lwd`) of the lines in the plot. The details of the `xyplot` function can be obtained by running the `?xyplot` command in the R console. Figures 2 and 3 show the interaction plots for bias and RMSE, respectively.

```
#Reading in "parameter recovery.csv"
result <- read.csv("parameter recovery.csv", header = TRUE)
result$SampleSize <- as.factor(result$SampleSize)
result$TestLength <- as.factor(result$TestLength)

#Create interaction plots using the Lattice package
install.packages("lattice")
library("lattice")
xyplot(Bias ~ TestLength | SampleSize,result,group=Parameter,xlab="Test Length",
type = "a",auto.key=list(corner=c(1,0.9),points=FALSE,lines=TRUE,columns=1),
par.settings=simpleTheme(lty=1:3,lwd=2))

xyplot(RMSE ~ TestLength | SampleSize,result,group=Parameter,xlab="Test Length",
type = "a",auto.key=list(corner=c(1,0.9),points=FALSE,lines=TRUE,columns=1),
par.settings = simpleTheme(lty=1:3,lwd=2))
```

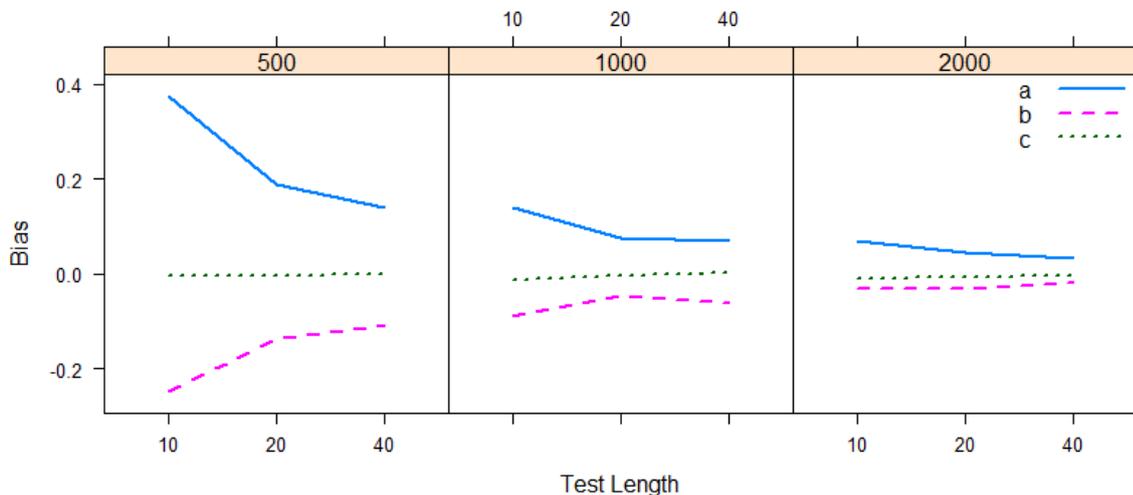


Figure 2. The Interaction Plot for Bias, Test Length, and Sample Size

Summary

This simulation study investigated the effects of sample size and test length on the recovery of item parameters from the 3PL model. To demonstrate how to evaluate the accuracy of estimated item parameters in R, this study included two simulation factors (test length and sample size) with 100 replications. The results suggested that both test length and sample size are negatively associated with the accuracy of item discrimination and item difficulty parameters. As sample size and test length increased, both bias and RMSE decreased. Unlike item discrimination and item difficulty parameters,

the lower asymptote (guessing) parameter was slightly affected by the simulation factors. Future studies can focus on item parameter recovery by expanding the simulation factors of the current study (e.g., smaller or larger sample sizes), using different IRT models, such as Graded Response Model (Samejima, 1969), adding other simulation factors (e.g., ability distribution, extreme guessing parameters for the 3PL model, and non-simple structure in multidimensionality).

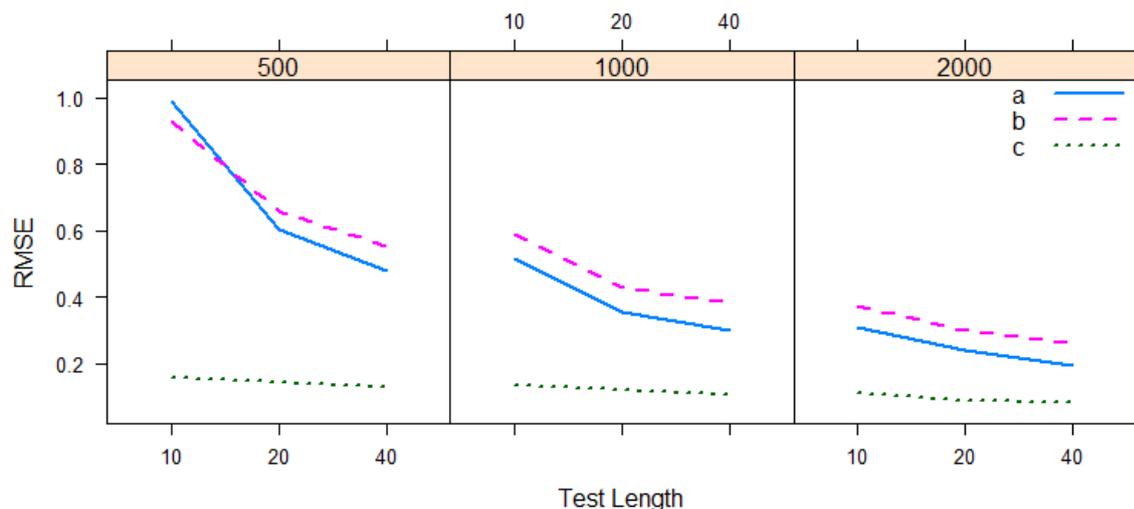


Figure 3. The Interaction Plot for RMSE, Test Length, and Sample Size

Simulation Study 2: Detecting Differential Item Functioning in Multidimensional IRT

The second simulation study aims to investigate the detection of differential item functioning (DIF) in the context of multidimensional IRT models. In educational testing, DIF occurs when the probability of responding to a dichotomous item correctly varies between focal and reference groups (e.g., male and female students), after controlling for examinees' ability levels. If the item is polytomous, then the probabilities of obtaining no credit, a partial credit (e.g., 1 point in a two-point item), or a full credit (e.g., 2 points in a two-point item) are expected to differ between the focal and reference groups, after controlling for examinees' ability levels. There are two types of DIF in test items: uniform and nonuniform DIF. If the focal group consistently underperforms or outperforms the reference group, then the item is flagged for having uniform DIF. If, however, the direction of bias changes between the focal and reference groups along the ability continuum, the item is flagged for having nonuniform DIF (Lee, Bulut, & Suh, 2016).

There are many methods in the literature to detect uniform and nonuniform DIF in the context of unidimensional IRT models. These methods include the Mantel-Haenszel method (Mantel & Haenszel, 1959), simultaneous item bias test (SIBTEST; Shealy & Stout, 1993), Raju's differential functioning of items and tests (DFIT; Raju, van der Linden, & Fler, 1995); and the multiple indicators multiple causes (MIMIC) model (Finch, 2005; Woods & Grimm, 2011). However, when the definition of DIF is extended to a multidimensional assessment that simultaneously measures two or more abilities, there are only a few DIF methods in the literature, such as multidimensional MIMIC-interaction model (Lee et al., 2016), IRT likelihood ratio test (Suh & Cho, 2014), and multidimensional SIBTEST (MULTISIB; Stout, Li, Nandakumar, & Bolt, 1997).

In this Monte Carlo simulation study, we use the IRT likelihood ratio test described by Suh and Cho (2014) for detecting uniform and nonuniform DIF in the context of multidimensional Graded Response Model (MGRM). The mathematical formulation of MGRM for a polytomous item with $K + 1$ response categories on an M -dimensional test becomes:

$$P_{ik}^*(\boldsymbol{\theta}) = \frac{1}{1 + e^{[-D \sum_{m=1}^M a_{im}(\theta_m - b_{ik})]}} \quad (4)$$

where $P_{ik}^*(\boldsymbol{\theta})$ is the probability of selecting the response option k ($k = 1, \dots, K$) in item i for an examinee with the ability vector of $\boldsymbol{\theta} = [\theta_1, \dots, \theta_M]$, b_{ik} is the boundary parameter of the k^{th} category of item i , a_{im} is the item discrimination parameter for item i on dimension m ($m = 1, \dots, M$), and D is a constant of 1.7 to transform the logistic IRT scale into the normal ogive scale.

In this study, we assume a two-dimensional test with a simple structure. The test consists of 30 items, where the first set of 15 items is loaded on the first dimension and the second set of 15 items is loaded on the second dimension. To simulate polytomous item responses, we use item characteristics similar to those from Jiang, Wang, and Weiss's (2016) recent study on MGRM. Item discrimination parameters are randomly drawn from a uniform distribution, $a \sim U(1.1, 2.8)$; the first category boundary parameter is randomly drawn from a uniform distribution, $b_1 \sim U(0.67, 2)$ and the other two category boundary parameters are created by subtracting a value randomly drawn from a uniform distribution, $b_2 = b_1 - U(0.67, 1.34)$ and $b_3 = b_2 - U(0.67, 1.34)$. In addition, the ability parameters are drawn from a multivariate normal distribution, $\boldsymbol{\theta} \sim MVN(0, \boldsymbol{\Sigma})$ where is $\boldsymbol{\Sigma}$ the variance-covariance matrix of the abilities.

Three simulation factors are manipulated in this study: sample size, DIF magnitude, and inter-dimensional correlation. Sample size is manipulated for the reference (R) and focal (F) groups as R1000/F200, R1500/F500, and R1000/F1000. DIF magnitude is manipulated as 0, 0.3, or 0.6 logit difference for uniform and nonuniform DIF. DIF magnitude is added to the category boundary parameters for uniform DIF and to the item discrimination parameters for nonuniform DIF. We assume that the focal group is at a disadvantage due to uniform or nonuniform DIF. Inter-dimensional correlation refers to the correlation between the two ability dimensions. Inter-dimensional correlation is manipulated as $\rho=0$, $\rho=.3$, or $\rho=.5$. Based on the value of the inter-dimensional correlation, the off-diagonal elements of the variance-covariance matrix ($\boldsymbol{\Sigma}$) are replaced with ρ , while the diagonal elements remain as "1".

For simulating polytomous item responses, estimating the item parameters based on MGRM, and running the IRT likelihood ratio tests, we will use the MASS package (Venables & Ripley, 2002) and the mirt package (Chalmers, 2012) in R (R Core Team, 2017). In addition, we will use the doParallel package (Revolution Analytics & Weston, 2015) to benefit from the parallel computing to further speed up the estimation process. To install and then activate these packages, the following commands should be first run in the R console:

```
install.packages("doParallel")
library("doParallel")
library("mirt")
library("MASS")
```

Next, we define a function called detectDIF, which generates item parameters and simulates polytomous item responses based on MGRM, estimates the item parameters using the simulated data, runs IRT likelihood ratio tests to detect uniform and nonuniform DIF on a particular set of items, and computes the true positive rates (i.e., power) and false positive rates (i.e., Type I error) as the evaluation criteria.

The detectDIF function requires four input values: sample.size defines the size of reference and focal groups (e.g., sample.size = c(1000, 200) for the reference group of 1000 examinees and the focal group of 200 examinees); DIF.size defines the magnitude of uniform DIF and nonuniform DIF (e.g., DIF.size = c(0.3, 0) for 0.3 difference in the category threshold parameters as uniform DIF and DIF.size = c(0, 0.3) for 0.3 difference in the discrimination parameters as nonuniform DIF); cor specifies the correlation between the two ability dimensions (e.g., cor = 0.5 for a correlation of 0.5 between the two abilities); and seed is the user-defined seed for the data generation process. Based on the selected input values, the function generates a 30-item, polytomously-scored test in which items 1, 7, 15, 16, 23, and 30 are tested for uniform and nonuniform DIF. These items are particularly selected because they represent a combination of low-, medium-, and high-difficulty as well as low-, medium-, and high-discrimination parameters.

The IRT likelihood ratio test examines the likelihood difference between two nested IRT models based on a chi-square test with degrees of freedom equal to the difference between the numbers of estimated item parameters between the two IRT models (see Suh and Cho [2014] for more details on this procedure). To investigate uniform DIF with the IRT likelihood ratio test, we first estimate `mod0` that assumes that all items except for items 1, 7, 15, 16, 23, 30 are invariant between the focal and reference group. Next, we estimate `mod1` that constrains the category boundary parameters `b1`, `b2`, and `b3` to be equal between the focal and reference group for each of the six DIF items and tests whether there is a significant change in the model likelihood due to the equality constraints. Significant likelihood changes from `mod0` to `mod1` indicate that the items being tested exhibit uniform DIF. Nonuniform DIF is examined by comparing `mod0` against `mod1` and `mod2`, which constrain the item discrimination parameters `a1` and `a2` to be equal between the focal and reference groups, respectively. Significant likelihood changes between these models indicate that the items being tested exhibit nonuniform DIF.

The `detectDIF` function returns a data frame in which the aforementioned six items and the p-values from the IRT likelihood ratio test for each item are listed. When one of the input values for `DIF.size` is larger than zero, the function returns the p-values for detecting DIF correctly (i.e., power) for the six items listed above. If, however, `DIF.size = c(0, 0)`, then the function returns the p-values for detecting DIF falsely (i.e., Type I error) for the items. In this study, we assume that DIF occurs either in the category boundary parameters or in the item discrimination parameters. Therefore, one of the values in `DIF.size` will always be zero when running the analysis for power.

```
detectDIF <- function(sample.size, DIF.size, cor, seed) {

  require("mirt")
  require("MASS")
  set.seed(seed)

  #Define multidimensional abilities for reference and focal groups
  theta.ref <- mvrnorm(n = sample.size[1], rep(0, 2), matrix(c(1,cor,cor,1),2,2))
  theta.foc <- mvrnorm(n = sample.size[2], rep(0, 2), matrix(c(1,cor,cor,1),2,2))

  #Generate item parameters for reference and focal groups
  a1 <- c(runif(n = 15, min = 1.1, max = 2.8), rep(0,15))
  a2 <- c(rep(0,15), runif(n = 15, min = 1.1, max = 2.8))
  a.ref <- as.matrix(cbind(a1, a2), ncol = 2)
  b1 <- runif(n = 30, min = 0.67, max = 2)
  b2 <- b1 - runif(n = 30, min = 0.67, max = 1.34)
  b3 <- b2 - runif(n = 30, min = 0.67, max = 1.34)
  b.ref <- as.matrix(cbind(b1, b2, b3), ncol = 3)

  #Uniform and nonuniform DIF for items 1, 7, 15, 16, 23, and 30
  b.foc <- b.ref
  b.foc[c(1,7,15,16,23,30),] <- b.foc[c(1,7,15,16,23,30),]+DIF.size[1]
  a.foc <- a.ref
  a.foc[c(1,7,15),1] <- a.foc[c(1,7,15),1]+DIF.size[2]
  a.foc[c(16,23,30),2] <- a.foc[c(16,23,30),2]+DIF.size[2]

  #Generate item responses according to MGRM
  ref <- simdata(a = a.ref, d = b.ref, itemtype = 'graded', Theta = theta.ref)
  foc <- simdata(a = a.foc, d = b.foc, itemtype = 'graded', Theta = theta.foc)
  dat <- rbind(ref, foc)
  #Define the group variable (0=reference; 1=focal) and test DIF using mirt
  group <- c(rep("0", sample.size[1]), rep("1", sample.size[2]))
  itemnames <- colnames(dat)
  model <- 'f1 = 1-15
           f2 = 16-30
           COV = F1*F2'
  model.mgrm <- mirt.model(model)
```

```

#Test uniform DIF
if(DIF.size[1]>0 & DIF.size[2]==0) {
  mod0 <- multipleGroup(data = dat, model = model.mgrm, group = group,
    invariance = c(itemnames[-c(1,7,15,16,23,30)],
    'free_means', 'free_var'), verbose = FALSE)
  mod1 <- DIF(mod0, c('d1','d2','d3'), items2test = c(1,7,15,16,23,30))

  result <- data.frame(items=c(1,7,15,16,23,30),
    DIF=c(mod1[[1]][2,8], mod1[[2]][2,8], mod1[[3]][2,8],
    mod1[[4]][2,8], mod1[[5]][2,8], mod1[[6]][2,8]))
} else

#Test nonuniform DIF
if(DIF.size[1]==0 & DIF.size[2]>0) {
  mod0 <- multipleGroup(data = dat, model = model.mgrm, group = group,
    invariance = c(itemnames[-c(1,7,15,16,23,30)],
    'free_means', 'free_var'), verbose = FALSE)

  mod1 <- DIF(mod0, c('a1'), items2test = c(1,7,15))
  mod2 <- DIF(mod0, c('a2'), items2test = c(16,23,30))
  result <- data.frame(items=c(1,7,15,16,23,30),
    DIF=c(mod1[[1]][2,8], mod1[[2]][2,8], mod1[[3]][2,8],
    mod2[[1]][2,8], mod2[[2]][2,8], mod2[[3]][2,8]))
} else

#Test type I error
if(DIF.size[1]==0 & DIF.size[2]==0) {
  mod0 <- multipleGroup(data = dat, model = model.mgrm, group = group,
    invariance = c(itemnames[-c(1,7,15,16,23,30)],
    'free_means', 'free_var'), verbose = FALSE)

  mod1 <- DIF(mod0, c('a1','d1','d2','d3'), items2test = c(1,7,15))
  mod2 <- DIF(mod0, c('a2','d1','d2','d3'), items2test = c(16,23,30))
  result <- data.frame(items=c(1,7,15,16,23,30),
    DIF=c(mod1[[1]][2,8], mod1[[2]][2,8], mod1[[3]][2,8],
    mod2[[1]][2,8], mod2[[2]][2,8], mod2[[3]][2,8]))
}
return(result)
}

```

For this study, we use 30 replications, as in Jiang et al.'s (2016) simulation study with MGRM. Despite using only 30 replications, this simulation study is more complex compared to the first simulation study presented earlier because in addition to estimating item parameters from a two-dimensional MGRM, we conduct a series of IRT likelihood ratio tests to examine uniform and nonuniform DIF across six items (items 1, 7, 15, 16, 23, and 30). To increase the speed of the entire simulation process, we use the `doParallel` package. First, we generate a set of 30 random seeds ranging from 0 to 10000 and save the generated seeds in a data set called "myseed". Next, using a computer with a multi-core processor, we allocate multiple cores for our simulation study. To check the number of processors in a computer, the researcher can first run `detectCores()`. To assign a particular number of cores, `registerDoParallel()` should be used. For example, to allocate 8 cores for the simulation study, `registerDoParallel(8)` should be used. Once this command is executed, the simulation process can be completed using 8 cores rather than a single core, which is the default setting in R. Although it is possible to use all available cores in a computer, this could be problematic because using all available cores would slow down the operation of the computer significantly, especially when performing other tasks in addition to the simulation study. The parallel computing is particularly useful when an iterative computing process – such as a simulation study – is implemented. Because the current simulation study requires 30 replications, estimating multiple replications simultaneously is expected to reduce the duration of simulation significantly (e.g., with 8 cores, it would be theoretically 8 times faster than a regular estimation with a single core).

```

myseed <- sample.int(n=10000, size = 30)
detectCores()
registerDoParallel(8)

```

Once the parallel computing process is set up, the final step of this simulation study is to run the simulation study by changing the input values in the `detectDIF` function. The `foreach` function from the `doParallel` package is used for creating a loop of 30 replications. After each replication, the results will be combined in a single data set called `result`. We use the `ifelse` function to create a binary variable for the items in the `result` data set that have a p-value less than .05 (i.e., significant DIF based on the IRT likelihood ratio test). The `mean` function will return the proportion of the items that have indicated significant DIF (i.e., average power or Type I error depending on the condition). The following R commands demonstrate an example for running a particular combination of the simulation factors. For each replication, a random seed from `myseed` is selected. Then, the `detectDIF` function is executed using sample sizes of 1000 and 200 for the reference and focal groups, the DIF magnitude of 0.3 for uniform DIF, and a correlation of $\rho=.3$ between the two ability dimensions. After 30 replications are complete, the average proportion of significant IRT likelihood ratio tests across six items is reported with two decimal points (using the `round` function).

```
result <- foreach (i = 1:30, .combine=rbind) %dopar% {
  detectDIF(sample.size=c(1000, 200), DIF.size=c(0.3, 0), cor=0.3, seed=myseed[i])
}
round(mean(ifelse(result$DIF < 0.05, 1, 0)),2)
```

Table 2. Power and Type I Error Rates for Detecting DIF in MGRM

Sample Size	Correlation	Power for Uniform DIF		Power for Nonuniform DIF		Type I Error
		0.3	0.6	0.3	0.6	
R1000/F200	0	.23	.80	.21	.51	.09
	.3	.28	.84	.23	.46	.07
	.5	.31	.82	.22	.49	.06
R1500/F500	0	.53	1	.39	.83	.07
	.3	.56	.98	.40	.82	.04
	.5	.55	.98	.44	.81	.04
R1000/F1000	0	.60	.98	.44	.66	.06
	.3	.56	.98	.37	.68	.03
	.5	.64	.97	.43	.73	.06

Table 2 shows a summary of the findings across all simulation factors. The results show that sample size and DIF magnitude are positively associated with the power of the IRT likelihood ratio test when detecting uniform and nonuniform DIF in MGRM. As sample size and DIF magnitude increased, power rates of the IRT likelihood ratio for detecting both uniform and nonuniform DIF test also increased. Unlike sample size and DIF magnitude, the effect of inter-dimensional correlation does not seem to be consistent regarding power rates. When DIF magnitude is large (i.e., 0.6), the correlation between the two dimensions has no effect on power rates for uniform DIF. However, the correlation between the dimensions affects power rates slightly for nonuniform DIF. Overall, the IRT likelihood ratio test appears to detect uniform DIF more precisely than nonuniform DIF. Type I error rates appear to be reasonable, although small sample size condition (R1000/F200) seems to have higher Type I error rates than the other two sample size conditions. As for the power rates, the effect of inter-dimensional correlation is not consistent regarding Type I error rates. The results in Table 2 can also be summarized with a scatterplot to demonstrate the relationships between the simulation factors more visually (see the graphical example in Simulation Study 1 for more details.).

Summary

This simulation study investigated the effects of sample size, DIF magnitude, and inter-dimensional correlation in detecting uniform and nonuniform DIF for a multidimensional polytomous IRT model (i.e., MGRM). Power rates in detecting DIF correctly and Type I error rates in detecting DIF falsely are used as the evaluation criteria. For the demonstration purposes, we only used 30 replications but

the number of replications could be easily increased with the help of parallel computing. This would result in more reliable simulation results. Future studies can expand the simulation factors of the current study to have a more comprehensive analysis. For example, different values for DIF magnitude, sample size, and inter-dimensional correlations can be used. Furthermore, new simulation factors can be included. For example, three or higher dimensional structures can be used to see the impact of the number of dimensions, which would also allow examining the impact of varying inter-dimensional correlations. In addition, instead of a simple structure, a complex test structure with items associated with multiple dimensions can be assumed.

Simulation Study 3: Investigating Unidimensionality

The third simulation study aims to investigate test dimensionality. Unidimensionality is essential for the test theories like as Classical Test Theory or Item Response Theory. Therefore, the investigation of test dimensionality is very important. The unidimensionality assumption requires that there is a single latent trait underlying a set of test items. As Hambleton, Swaminathan, and Rogers (1991) pointed out, the unidimensionality assumption may not hold for most measurement instruments in education, psychology, and other social sciences due to complex cognitive and non-cognitive factors, such as motivation, anxiety, and ability to work quickly. Therefore, we can expect that at least one minor extra factor confounds unidimensionality. However, finding a major component or a factor underlying the data is adequate to meet the unidimensionality assumption.

Monte Carlo studies can be very convenient for investigating the factors affecting unidimensionality under various conditions. In the following Monte Carlo study, we aim to examine the impact of sample size, the number of items associated with a secondary (nuisance) dimension, and inter-dimensional correlations on the detection of unidimensionality. Two-dimensional response data with a simple structure are generated. While most items are assumed to be associated with the first ability dimension, the number of items (10, 20, or 30 items) associated with a secondary dimension is manipulated as a simulation factor. Second, the correlation between the two ability dimensions ($\rho=.3$, $\rho=.6$, or $\rho=.9$) is manipulated. As the third simulation factor, sample size (500, 1000, or 3000 examinees) is modified because sample size is considered an important factor for the accuracy of dimensionality analyses. The three simulation factors are fully crossed, resulting in $3 \times 3 \times 3 = 27$ cells in total. One hundred replications are conducted for each cell.

A multidimensional two-parameter logistic IRT model (M2PL) is used for data generation. The M2PL model can be written as follows:

$$P_i(\boldsymbol{\theta}) = \frac{\exp(\sum_{m=1}^M a_{im}\theta_m + d_i)}{1 + \exp(\sum_{m=1}^M a_{im}\theta_m + d_i)} \quad (5)$$

where $P_i(\boldsymbol{\theta})$ is the probability of responding to item i correctly for an examinee with the ability vector of $\boldsymbol{\theta} = [\theta_1, \dots, \theta_M]$, a_{im} is the discrimination parameter of item i related to ability dimension m ($m = 1, 2, \dots, M$), and d_i is the difficulty parameter of item i . In this study, the item discrimination parameters are randomly drawn from a uniform distribution, $a \sim U(1.1, 2.8)$, the item difficulty parameters are randomly drawn from a uniform distribution, $d \sim U(0.67, 2.00)$, and the ability values are obtained from a multivariate normal distribution, $\boldsymbol{\theta} \sim MVN(0, \boldsymbol{\Sigma})$ where is $\boldsymbol{\Sigma}$ the variance-covariance matrix of the abilities. Each generated data set is analyzed with NOHARM explanatory factor analysis (McDonald, 1997), which is an effective method for finding the number of underlying dimensions in item response data (Finch & Habing, 2005). NOHARM is implemented with one factor restriction using the sirt package (Robitzsch, 2017). The average root mean square error of approximation (RMSEA) is used as the evaluation criterion. RMSEA values smaller than .05 are usually considered a close fit, whereas RMSEA values equal or greater than .10 are considered a poor fit (Browne & Cudeck, 1993; Hu & Bentler, 1999).

For this study, we define a function called `detectDIM`, which draws item difficulty and item discrimination parameters according the distributions explained earlier, simulates two-dimensional

response data with a simple structure based on the M2PL model, fits an exploratory factor model with a one-factor restriction (i.e., unidimensional model) to the simulated data, and extracts the RMSEA value as the evaluation criterion. Before using the `detectDIM` function, the following packages must be installed and activated:

```
install.packages("sirt")
library("sirt")
library("mirt")
library("MASS")
```

The `detectDIM` function requires five input values: `sample.size` for the number of examinees, `testLength1` for the number of items with a non-zero loading on the first ability dimension, `testLength2` for the number of items with a non-zero loading on the second ability dimension, `cor` for the correlation between the two ability dimensions, and `seed` for setting the random number generator. The `detectDIM` function is shown below:

```
detectDIM <- function(sample.size, testLength1, testLength2, cor, seed) {

  require("mirt")
  require("MASS")
  require("sirt")
  set.seed(seed)

  # Generate item discrimination and difficulty parameters
  a1 <- c(runif(n = testLength1, min = 1.1, max = 2.8), rep(0, testLength2))
  a2 <- c(rep(0, testLength1), runif(n = testLength2, min = 1.1, max = 2.8))
  disc.matrix <- as.matrix(cbind(a1, a2), ncol = 2)
  difficulty <- runif(n = (testLength1 + testLength2), min = 0.67, max = 2)

  # Specify inter-dimensional correlations
  sigma <- matrix(c(1, cor, cor, 1), 2, 2)

  # Simulate data
  dataset <- simdata(disc.matrix, difficulty, sample.size, itemtype = 'dich',
                    sigma = sigma)

  # Analyze the simulated data by fitting a unidimensional model
  noharmOneFactorSolution <- noharm.sirt(dat = dataset, dimensions = 1)

  # Summarize the results
  result <- data.frame(sample.size = sample.size, testLength1 = testLength1,
                       testLength2 = testLength2, cor = cor,
                       RMSEA = noharmOneFactorSolution$rmsea)

  return(result)
}
```

To start the simulation study, we first generate 100 random seeds ranging from 0 to 1,000,000 and save the seeds into a file called “simulation seeds.csv”. The numbers stored in this file will be useful if we want to replicate the findings of this study in the future. Next, we create an empty data frame called “results.csv” and save this file into the current directory. This is a comma-separated-values file, which can be opened with any text editor or Microsoft Excel. This file will store all average RMSEA values across the 27 simulation cells. For each cell, 100 replications will be temporarily stored in a data set called “result” and the average RMSEA values from this data set will be stored in the results.csv file. Unlike in the first two simulation studies presented earlier, the input values in this simulation study are entered into nested loops so that these input values do not have to be modified manually. For example, for `(ss in 1:length(sample.size))` creates a loop with three sample

size values (see `sample.size <- c(500, 1000, 3000)`). These values will be used iteratively in the `detectDIM` function via `sample.size = sample.size[ss]`. In addition to the loop for sample size, there are three other loops for test length for the second dimension, correlations between the ability dimensions, and the replications based on the random seeds, respectively. The simulation process will stop automatically after 100 replications are completed for each of the 27 cells, and it will write the average RMSEA values into the `results.csv` file.

```
myseed <- sample.int(n = 1000000, size = 100)
write.csv(myseed, "simulation seeds.csv", row.names = FALSE)
write.table(matrix(c("Sample Size", "Test Length 1", "Test Length 2",
                    "Correlation", "Mean RMSEA"), 1, 5), "results.csv", sep = ",",
            col.names = FALSE, row.names = FALSE)
# Create an empty data frame to save the results
result <- data.frame(sample.size = 0, testLength1 = 0, testLength2 = 0,
                    cor = 0, RMSEA = 0)

# Run all the input values through loops
sample.size <- c(500, 1000, 3000)
test.length2 <- c(10, 20, 30)
correlation <- c(0.3, 0.6, 0.9)
for (ss in 1:length(sample.size)) {
  for (tl in 1:length(test.length2)) {
    for (k in 1:length(correlation)) {
      for (i in 1:length(myseed)) {
        result[i, ] <- detectDIM(sample.size = sample.size[ss], testLength1 = 30,
                                testLength2 = test.length2[tl], cor = correlation[k],
                                seed = myseed[i])
      }
      meanRMSEAs <- round(colMeans(result), 3)
      write.table(matrix(meanRMSEAs, 1, 5), "results.csv", sep = ",",
                      col.names = FALSE, row.names = FALSE, append = TRUE)
    }
  }
}
```

After all of the iterations (27 cells x 100 replications = 2700 iterations in total) are completed, we read “`results.csv`” and call the data set “`summary`” in R. Then, we use the `summary` data set to create a graphical summary of the findings through the `dotplot` function in the `lattice` package. To consider the simulation values as labels in the graph, we use the `factor` function, which saves a numerical variable as a character variable. In addition, we use the `paste0` function to make the labels more clear. For example, instead of using 500, 1000, and 3000 as the labels, we combine these values with the text “`Sample Size=`”, and create the following labels: `Sample Size=500`, `Sample Size=1000`, and `Sample Size=3000`. With the `levels` option, it is possible to set the order of the created labels, which changes in which order the labels will appear in the graph. The `dotplot` function creates a scatterplot of the correlation between the dimensions and average RMSEA values. The vertical line “`|`” between `Mean.RMSEA` and `Sample.Size*Test.Length.2` allows us to create a separate scatterplot for each sample size and test length 2 combination.

```
# Read in the summary results in
summary <- read.csv("results.csv", header = TRUE)
summary$Sample.Size <- factor(paste0("Sample Size=", summary$Sample.Size),
                             levels = c("Sample Size=500", "Sample Size=1000",
                                           "Sample Size=3000"))
summary$Test.Length.2 <- factor(paste0("Test Length 2=", summary$Test.Length.2))
summary$Correlation <- factor(summary$Correlation)

library(lattice)
```

```
dotplot(Correlation ~ Mean.RMSEA | Sample.Size*Test.Length.2 , data = summary,
        pch=c(2), cex=1.5, xlab = "Average RMSEA", aspect=0.5, layout = c(3,3),
        ylab="Correlation", xlim=c(0,0.3))
```

Figure 4 shows the results of the Monte Carlo simulation study across the three simulation factors. The results indicate that test length and inter-dimensional correlations can influence the dimensional structure of item response data. When the two ability dimensions are highly correlated (i.e., $\rho=.9$), average RMSEA values are less than 0.05, suggesting a close fit for the one-factor model. However, as the correlation between the ability dimensions decreases and the number of items associated with the secondary dimension increases, RMSEA values become substantially larger, suggesting a poor fit for the one-factor model. Unlike the test length and correlation factors, sample size does not appear to affect the magnitude of RMSEA. In Figure 4, as the sample size increases from 500 (first column from the left) to 3000 (last column from the left), the average RMSEA values remain nearly the same, holding the other two simulation factors constant.

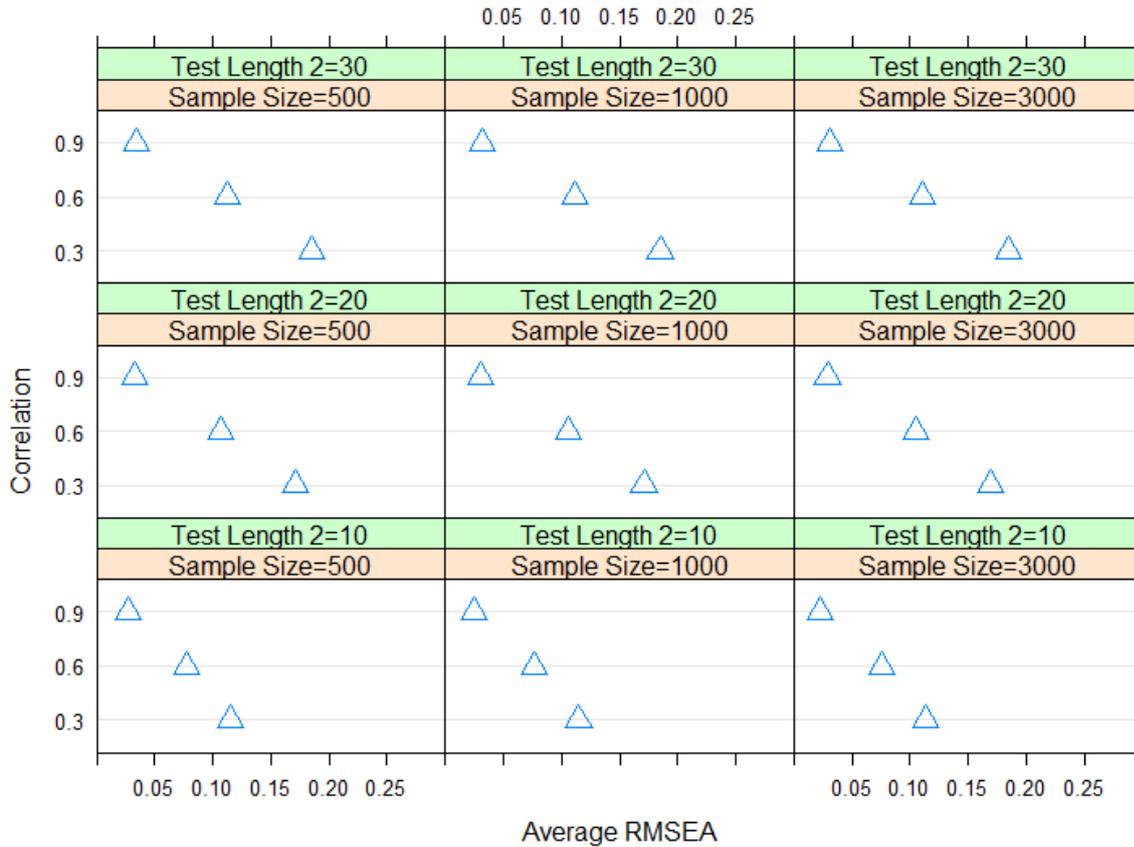


Figure 4. Average RMSEA Values across the Three Simulation Factors

Summary

This simulation study investigated the impact of a secondary dimension on the detection of unidimensionality. The simulation study involved three simulation factors: the number of items related to the secondary dimension, the correlation between the ability dimensions, and sample size. The M2PL model was used as the underlying model for data generation. A one-factor model was fit to each simulated data set using the `noharm.sirt` function in the `sirt` package and the RMSEA values were extracted as the evaluation criterion. Future studies can expand the scope of the current study. For example, different values for test length (e.g., fewer or more items) and inter-dimensional correlations (e.g., $\rho=0$) can be used. In addition, new simulation factors can be included to investigate different research questions. For example, the number of secondary (nuisance) dimensions can be influential on the detection of unidimensionality. This would also enable the use of varying

correlations between dimensions since there would be more values to modify in the variance-covariance matrix of the abilities. Finally, instead of a simple test structure, secondary dimensions can be used for generating a complex test structure where most items are dominantly loaded on the first dimension but the items also share some variance with the secondary dimensions.

DISCUSSION and CONCLUSION

Academic researchers and practitioners often use Monte Carlo simulation studies for investigating a wide range of research questions related to IRT. During the last decade, R has become one of the most popular software programs for designing and implementing Monte Carlo simulation studies in IRT. The R programming language allows researchers to simulate various types of data, analyze the generated data based on a particular model or method of interest, and summarize the results statistically and graphically. Given the growing popularity of R among researchers and practitioners, this study provided a brief introduction to the R programming language and demonstrated the use of R for conducting Monte Carlo studies in IRT. Each simulation study presented in this study focuses on a different aspect of IRT, involves a variety of simulation factors, and uses various criteria to evaluate the outcomes of the simulations. We recommend the readers either to conduct the same Monte Carlo simulation studies or to design their own simulation studies by following the R codes provided in this study. In addition, the readers who are interested in the nuts and bolts of the R programming language are encouraged to check out many R resources available on the internet (e.g., <https://journal.r-project.org/> and <http://www.statmethods.net/>).

REFERENCES

- Albano, A. D. (2016). equate: An R package for observed-score linking and equating. *Journal of Statistical Software*, 74(8), 1–36.
- Brown, T. A. (2006). *Confirmatory factor analysis for applied research*. New York: Guilford.
- Browne, M., & Cudeck, R. (1993). Alternative ways of assessing model fit. In K. A. Bollen, & J. S. Long (Eds.). *Testing structural equation models*. Newbury Park, CA: Sage.
- Bulut, O. (2013). *Between-person and within-person subscore reliability: Comparison of unidimensional and multidimensional IRT models* (Unpublished doctoral dissertation, University of Minnesota).
- Bulut, O., Davison, M. L., & Rodriguez, M. C. (2017). Estimating between-person and within-person subscore reliability with profile analysis. *Multivariate Behavioral Research*, 52(1), 86–104.
- Cai, L. (2013). *flexMIRT version 2.00: A numerical engine for flexible multilevel multidimensional item analysis and test scoring* [Computer software]. Chapel Hill, NC: Vector Psychometric Group.
- Cai, L., Thissen, D., & du Toit, S. H. C. (2011). *IRTPRO: Flexible, multidimensional, multiple categorical IRT modeling* [Computer software]. Lincolnwood, IL: Scientific Software International.
- Camilli, G. (1994). Origin of the scaling constant $d = 1.7$ in item response theory. *Journal of Educational and Behavioral Statistics*, 19(3), 293–295.
- Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, 48(6), 1–29.
- Chalmers, R. P. (2016). Generating adaptive and non-adaptive test interfaces for multidimensional item response theory applications. *Journal of Statistical Software*, 71(5), 1–39.
- Choi, S. W., Gibbons, L. E., & Crane, P. K. (2016). lordif: Logistic ordinal regression differential item functioning using IRT [Computer software]. Available from <https://CRAN.R-project.org/package=lordif>.
- Crocker, L., & Algina, J. (1986). *Introduction to classical and modern test theory*. New York: Holt, Rinehart and Winston, Inc.
- Feinberg, R. A., & Rubright, J. D. (2016). Conducting simulation studies in psychometrics. *Educational Measurement: Issues and Practice*, 35(2), 36–49.
- Finch, H. (2005). The MIMIC model as a method for detecting DIF: Comparison with Mantel-Haenszel, SIBTEST, and the IRT likelihood ratio. *Applied Psychological Measurement*, 29, 278–295.
- Finch, H., & Habing, B. (2005). Comparison of NOHARM and DETECT in item cluster recovery: Counting dimensions and allocating items. *Journal of Educational Measurement*, 42, 149–169.
- Hallgren, K. A. (2013). Conducting simulation studies in the R programming environment. *Tutorials in Quantitative Methods for Psychology*, 9(2), 43–60.

- Hambleton, R. K., Swaminathan, H., & Rogers, J. H. (1991). *Fundamentals of item response theory*. New York: Sage publications.
- Harwell, M. R., & Baker, F. B. (1991). The use of prior distributions in marginalized Bayesian item parameter estimation: A didactic. *Applied Psychological Measurement, 15*(4), 375–389.
- Harwell, M. R., Stone, C. A., Hsu, T., & Kirisci, L. (1996). Monte Carlo studies in item response theory. *Applied Psychological Measurement, 20*(2), 101–125.
- Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indices in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling, 6*, 1–55.
- Jiang, S., Wang, C., & Weiss, D. J. (2016). Sample size requirements for estimation of item parameters in the multidimensional graded response model. *Frontiers in Psychology, 7*(109), 1–10.
- Lee, S., Bulut, O., & Suh, Y. (2016). Multidimensional extension of multiple indicators multiple causes models to detect DIF. *Educational and Psychological Measurement*. Advance online publication. DOI: 10.1177/0013164416651116.
- Mair, P., Hatzinger, R., & Maier M. J. (2016). *eRm: Extended Rasch modeling* [Computer software]. Available from <http://CRAN.R-project.org/package=eRm>.
- Magis, D., Beland, S., Tuerlinckx, F., & De Boeck, P. (2010). A general framework and an R package for the detection of dichotomous differential item functioning. *Behavior Research Methods, 42*, 847–862.
- Magis, D., & Raiche, G. (2012). Random generation of response patterns under computerized adaptive testing with the R Package catR. *Journal of Statistical Software, 48*(8), 1–31.
- McDonald, R. P. (1997). Normal-ogive multidimensional model. In W. van der Linden & R. K. Hambleton, *Handbook of modern item response theory* (pp. 257-269). New York: Springer.
- Mislevy, R. J., & Stocking, M. L. (1989). A consumer's guide to LOGIST and BILOG. *Applied Psychological Measurement, 13*, 57–75.
- Mooney, C. Z. (1997). *Monte Carlo simulations*. Sage University Paper series on Quantitative Applications in the Social Sciences, 07-116. Thousand Oaks, CA: Sage.
- Muthén, L. K., & Muthén, B. O. (1998-2015). *Mplus user's guide (7th Edition)*. Los Angeles, CA: Muthén & Muthén.
- Partchev, I. (2016). *irtoys: A collection of functions related to item response theory (IRT)* [Computer software]. Available from <https://CRAN.R-project.org/package=irtoys>.
- R Core Team (2017). *R: A language and environment for statistical computing* [Computer software]. Vienna, Austria: R Foundation for Statistical Computing. Available from <https://cran.r-project.org/>.
- Raju, N. S., van der Linden, W. J., & Fleer, P. F. (1995). An IRT-based internal measure of test bias with application of differential item functioning. *Applied Psychological Measurement, 19*, 353–368.
- Revolution Analytics, & Weston, S. (2015). *doParallel: Foreach parallel adaptor for the 'parallel' package* [Computer software]. Available from <https://CRAN.R-project.org/package=doParallel>.
- Rizopoulos, D. (2006). ltm: An R package for latent variable modelling and item response theory analyses. *Journal of Statistical Software, 17*(5), 1–25.
- Robitzsch, A. (2017). *sirt: Supplementary item response theory models* [Computer software]. Available from <https://CRAN.R-project.org/package=sirt>.
- Robitzsch, A., & Rupp, A. A. (2009). The impact of missing data on the detection of differential item functioning. *Educational Psychological Measurement, 69*, 18–34.
- Rusch, T., Mair, P., & Hatzinger, R. (2013). *Psychometrics with R: A review of CRAN packages for item response theory*. Retrieved from <http://epub.wu.ac.at/4010/1/resrepIRThandbook.pdf>
- Samejima, F. (1969). *Estimation of latent ability using a response pattern of graded scores* (Psychometric Monograph No. 17). Richmond, VA: Psychometric Society.
- SAS Institute Inc. (2014). *Version 9.4 of the SAS system for Windows*. Cary, NC: SAS Institute Inc.
- Shealy, R., & Stout, W. (1993). A model-based standardization approach that separates true bias/DIF from group ability differences and detects test bias/DIF as well as item bias/DTF. *Psychometrika, 58*, 159–194.
- Sinharay, S., Stern, H. S., & Russell, D. (2001). The use of multiple imputation for the analysis of missing data. *Psychological Methods, 6*, 317–329.
- StataCorp. (2015). *Stata statistical software: Release 14*. College Station, TX: StataCorp LP.
- Stout, W., Li, H., Nandakumar, R., & Bolt, D. (1997). MULTISIB – A procedure to investigate DIF when a test is intentionally multidimensional. *Applied Psychological Measurement, 21*, 195–213.
- Suh, Y., & Cho, S. J. (2014). Chi-square difference tests for detecting functioning in a multidimensional IRT model: A Monte Carlo study. *Applied Psychological Measurement, 38*, 359–375.
- Ünlü, A., & Yanagida, T. (2011). R you ready for R?: The CRAN psychometrics task view. *British Journal of Mathematical and Statistical Psychology, 64*(1), 182–186.
- Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S*. New York, NY: Springer.
- Woods, C. M., & Grimm, K. J. (2011). Testing for nonuniform differential item functioning with multiple indicator multiple cause models. *Applied Psychological Measurement, 35*, 339–361.

Yao, L. (2003). *BMIRT: Bayesian multivariate item response theory* [Computer software]. Monterey, CA: Defense Manpower Data Center. Available from <http://www.bmirt.com>.

UZUN ÖZET

Giriş

Son yıllarda eğitimde ölçme ve psikometri alanlarındaki çalışmalar incelendiğinde Monte Carlo simülasyon çalışmalarının oldukça büyük yer tuttuğu görülmektedir. Bu çalışmalar, özellikle teoriye yönelik olan psikometrik çalışmalar için vazgeçilmez bir rol üstlenmiş konumdadır. Bunun başlıca nedenleri Monte Carlo simülasyon çalışmalarının sağlamış olduğu avantajlardan kaynaklanmaktadır. Bu avantajlar; simülasyon koşullarını değişimleyecek şekilde görgül veri toplamanın uygulamada mümkün olmaması veya maliyetinin yüksek olması, madde ve/veya birey parametrelerinin gerçek değerlerinin görgül veri kapsamında tam olarak bilinmemesinden kaynaklı oluşan sıkıntılar, görgül verinin kayıp verilerin yarattığı sorunlar, görgül verilerde karıştırıcı değişkenlerden arınık bir ortam sağlanamamasından kaynaklı olarak neden-sonuç ilişkilerinin kurulmasında yaşanan zorluklar olarak sıralanabilir. Son yıllarda yaşanan bir diğer gelişme ise R programı ve programlama dilinde oluşan gelişmelerdir. Ücretsiz ve açık kaynak kodlu olan R, başta istatistik olmak üzere, birçok alanda yaygın olarak kullanılmaya başlanmıştır. Psikometride R'ın yaygın olarak kullanılmaya başlandığı alanlar arasında yer almaktadır. R'ın bel kemiğini oluşturan CRAN web sayfasında yer alan dizinlerden biri olan "Task View", R'da yer alan paketlerin konulara göre kategorize edildiği bir alt alandır. Bu dizinde yer alan [Psychometrics](#) bağlantısı açıldığında, psikometriye yönelik olarak birçok çalışmanın yapılmış olduğu görülebilir. Bu çalışmalar kapsamında madde tepki kuramına (MTK) dair yer alan kestirimlere yönelik başlıca paketlerin, mirt (Chalmers, 2012), eRm (Mair, Hatzinger, & Maier, 2016), irtoys (Partchev, 2016), and ltm (Rizopoulos, 2006) olduğu söylenebilir. Bunun yanında bazı özelleştirilmiş MTK paketlerinden de söz etmek mümkündür: Değişen madde fonksiyonu için lordif (Choi, Gibbons, & Crane, 2016) ve difR (Magis, Beland, Tuerlinckx, & De Boeck, 2010); bilgisayar ortamında bireye uyarlanmış testler için catR (Magis & Raiche, 2012) ve mirtCAT (Chalmers, 2016); test eşitleme için equate (Albano, 2016) ve MTK için birçok açıdan destekleyici ve bağlantı kurucu nitelikte olan sirt (Robitzsch, 2017) bu paketler arasında gösterilebilir. R programlama dili kullanarak, R'da psikometriye yönelik paketler kullanarak birçok IRT tabanlı Monte Carlo çalışması yürütülmüştür ve hala da yürütülmektedir. Bu çalışmanın amacı IRT'ye dayalı veri üretimi ve incelemesini içeren simülasyon çalışmalarının R'da nasıl yapılması gerektiğine dair bilgilendirme sağlamak ve örnekler üzerinden elde edilmiş olan bulguları paylaşmaktır.

Yöntem

Bu çalışma kapsamında üç adet Monte Carlo simülasyon çalışması yürütülmüştür. Bu çalışmalardan birincisi parametre yeniden elde edimine yönelik olup, ikincisi değişen madde fonksiyonuna yönelik bir çalışmadır. Üçüncü çalışma ise çeşitli faktör yapılarındaki tek boyutluluğun incelenmesi üzerinedir.

Birinci simülasyon çalışmasında; tek boyutlu 3 parametrelili lojistik modele dayalı olarak veri üretimi yapılmış ve bu üretimlerden elde edilen madde parametrelerinin orijinal madde parametrelerine olan benzeşikliği incelemiştir. İncelemeler yanlılık ve RMSE kapsamında gerçekleştirilmiştir. Çalışma kapsamında örneklem büyüklüğü (500, 1000, 2000) olacak şekilde ve test uzunluğu (10, 20, 40) olacak şekilde değişimlenmiştir. Simülasyon çerçevesinde b parametresi standart normal dağılımdan, $b \sim N(0, 1)$; ayırt edicilik parametresi olan a parametresi, log-normal dağılımdan, $a \sim \ln N(0.3, 0.2)$; ve düşük asimptot parametresi olan c beta dağılımından, $c \sim \text{Beta}(20, 90)$ elde edilmiştir.

İkinci simülasyon çalışmasında; tek biçimli ve tek biçimli olmayan değişen madde fonksiyonu içeren çoklu puanlanan maddeler barındıran veri üretimi gerçekleştirilmiştir. Üretim çok boyutlu aşamalı tepki modeline dayalı olarak 30 madde için yapılmıştır. İlk 15 madde birinci faktöre ait olup ikinci 15 madde ise ikinci faktöre ait olacak şekilde basit yapı formatında üretim yapılmıştır. Tek biçimli değişen madde fonksiyonu için odak ve referans gruplarının verilerinin eşik değerleri farklılaştırılmıştır. Tek düzeyli olmayan değişen madde fonksiyonu verisi üretirken ise belirtilen

grupların madde ayırt edicilik değerleri de farklılaştırılmıştır. Simülasyon kriteri olarak; örneklem büyüklüğü (R1000/O200, R1500/O500 ve R1000/F1000), Değişen madde fonksiyonu logit büyüklüğü (0, 0.3, 0.6). Olabilirlik testi kullanılarak güç ve birinci tip hata incelemesi gerçekleştirilmiştir.

Üçüncü simülasyon çalışmasında ise çok boyutlu MTK'ya dayalı olarak 2 boyutlu, basit yapı formatında, iki kategorik maddelerden oluşan üretim gerçekleştirilmiştir. İlk boyutta yer alan madde sayısı sabit tutulmakla beraber ikinci boyutta yer alan maddelerin sayısı (10, 20, 30) olacak şekilde değişimlenmiştir. Diğer simülasyon kriterleri ise örneklem büyüklüğü (500,1000, 3000) ve boyutlar arasındaki korelasyondur (0.3, 0.6, 0.9). Simülasyon için kullanılan a ve d parametreleri uniform dağılımdan elde edilirken $a \sim U(1.1, 2.8)$, $d \sim U(0.67, 2.00)$, örtük özelliklere ilişkin yetenek dağılımları tanımlanan korelasyonlar bağlı olarak, ortalamaları 0 olan çok değişkenli normal dağılımdan $\theta \sim MVN(0, \Sigma)$ elde edilmiştir. Üretilmiş olan veriler sirt (Robitzsch, 2017) paketi kullanılarak analiz edilmiştir. Verilere, tek faktör sınırlandırması altında açımlyıcı NOHARM faktör analizi uygulanmıştır. Yapılmış olan çözümlerden elde edilen RMSEA değerleri boyutluluk değerlendirmesi için kullanılmıştır.

Sonuçlar ve Tartışma

Birinci simülasyon çalışması sonucunda, madde ayırt edicilik parametreleri ve madde güçlük parametrelerinin kestirim uygunluğunun, test uzunluğu ve örneklem büyüklüğü ile negatif yönde bağlantı gösterdiği gözlenmiştir. Başka bir deyişle test uzunluğu ve örneklem büyüklüğü arttıkça elde edilen yanlılık ve RMSE değerleri düşmüştür. Buna ek olarak düşük asimptot parametresi olan c parametresinin kestirim uygunluğu simülasyon kriterlerinden oldukça az etkilenmiştir.

İkinci simülasyon çalışmasının sonucunda, örneklem büyüklüğünün ve değişen madde fonksiyonu büyüklüğünün, olabilirlik testi sonuçlarının gücüyle pozitif yönde ilişki gösterdiği söylenebilir. Başka bir deyişle örneklem büyüklüğü ve değişen madde fonksiyonu büyüklüğü arttıkça olabilirlik testinin de gücü artmaktadır. Boyutlar arası korelasyon ve güç arasında tutarlı bir ilişkilendirme sağlanamamıştır. Çalışmadan elde edilen diğer bir sonuç ise tek biçimli olan değişen madde fonksiyonu için olan sonuçların güç açısından daha yüksek olmasıdır. 1. Tip Hata sonuçları için, genel olarak kabul edilebilir sınırlar etrafında olduğu, görece daha düşük örneklem büyüklüklerinde hata oranının arttığı, boyutlar arası korelasyona dayalı incelemelerin güç çalışmasındakine benzer bir şekilde tutarlı bir sonuç vermediği söylenebilir.

Üçüncü simülasyon çalışmasından elde edilen sonuçlar incelendiğinde, ikinci boyutta yer alan madde sayısının ve boyutlar arasındaki korelasyonun verinin boyutluluk yapısını etkilediği görülmüştür. Boyutlar arasındaki korelasyon arttıkça, veri için yapılan tek boyutlu çözümlenin daha uygun olduğu görülmüştür. Bununla beraber boyutlular arasındaki korelasyon azaldığında ve ikinci boyuttaki madde sayısı arttığında verinin tek boyutluluktan uzaklaştığı gözlemlenmiştir. Son olarak örneklem büyüklüğünün, boyutluluğu diğer faktörler kadar etkilemediği görülmüştür.