International Journal of Assessment Tools in Education
2019, Vol. 6, No. 5-Special Issue, 44–56
https://dx.doi.org/10.21449/ijate.621157
Published at http://www.ijate.net http://dergipark.gov.tr ***Research Article***

Computer Adaptive Testing Simulations in R

Başak Erdem-Kara [1,*]

[1] Hacettepe University, Education Faculty, Measurement and Evaluation in Education Department, Turkey

ARTICLE HISTORY
Received: 17 September 2019
Accepted: 04 December 2019

KEYWORDS
Computer adaptive testing,
Simulation,
R programming language

Abstract: Computer adaptive testing is an important research field in educational measurement, and simulation studies play a critically important role in CAT development and evaluation. Both Monte Carlo and Post Hoc simulations are frequently used in CAT studies in order to investigate the effects of different factors on test efficiency and to compare different test designs. Although there are several softwares for CAT simulations, R is preferred since it includes many free packages and gives researchers opportunity to write their own functions according to their own requirements besides being free. The purpose of this study is to make an introduction and demonstration of how to use catR package in CAT simulations. Different examples were provided in the context of this study and R codes were presented with explanations. Then, the output files were briefly explained. It is thought that this paper is helpful for the researchers who are interested in CAT simulations.

1. INTRODUCTION

Over recent decades, advances in computer technology have made computer based tests (CBT) a popular alternative to linear tests. Especially computer adaptive test (CAT) which is a kind of CBT, have become popular since they provide more efficient and more precise measurement of test takers' performance than those that linear tests provide (Wainer & Mislevy, 2000; Weiss & Kingsbury, 1984; Yan, Lewis & von Davier, 2014). In computer adaptive tests, each new item that examinee faces with is selected from the item pool according to examinees' performance on all previous items. Since examinees only face with items appropriate for their ability levels, they do not have to spend time for easier or more difficult questions for themselves. As a result, the test results in a shorter length (Hendrickson, 2007; Wainer, 2000).

Adaptive testing is a well-established procedure and an active research field in educational and psychological assessment (Magis & Raiche, 2011; Magis, Yan & von Davier, 2017). However, while developing an adaptive test, there are lots of factors to be determined such as the properties of item bank, test length, test design, content balancing, item exposure etc. Researchers try to find an answer for the question 'In what conditions, does the test perform best?' or 'How does test performance change under different conditions?'. Collecting empirical data is not always possible or costly while manipulating different conditions. Therefore,

CONTACT: Başak ERDEM-KARA ✉ basakerdem@hacettepe.edu.tr Hacettepe University, Education Faculty, Measurement and Evaluation in Education Department, Turkey

ISSN-e: 2148-7456 /© IJATE 2019

simulation studies can be a useful way to investigate those questions (Bulut & Sünbül, 2017; Lee, Choi & Cohen, 2018; Magis, Yan & von Davier, 2017). Similarly, Han & Kosinski (2014) stated that simulation techniques have critically important roles for CAT development and evaluation.

There are several softwares for CAT simulations such as SimulCAT (Han, 2012), CATSim (Weiss & Guyer, 2010) or Firestar (Choi, 2009). In the context of this study, the demonstration of how to use R programming language (R Core Team, 2017) for CAT simulations through the usage of 'catR' package on RStudio 1.1.463 was aimed. As Magis, Yan & von Davier (2017) stated, R gives researchers the chance of adding their own functions according to their requirements and additional free packages can be installed easily. This is the reason why it was preferred in this study. In the context of the present study, firstly computer adaptive testing was explained with basic concepts and terms and the demonstration of CAT simulations with catR package was given afterwards.

2. COMPUTER ADAPTIVE TESTING

Embretson and Reise (2000) stated that the main purpose of CAT is to provide maximally efficient and informative items for each test taker. For this purpose, different items are administered to different examinees according to their proficiency levels. The ability level of test takers are estimated and updated after the administration of each item and the next item is chosen based on that updated level then. That process continues until stopping criterion is met. The schematic representation of testing process is given in Figure 1 below.

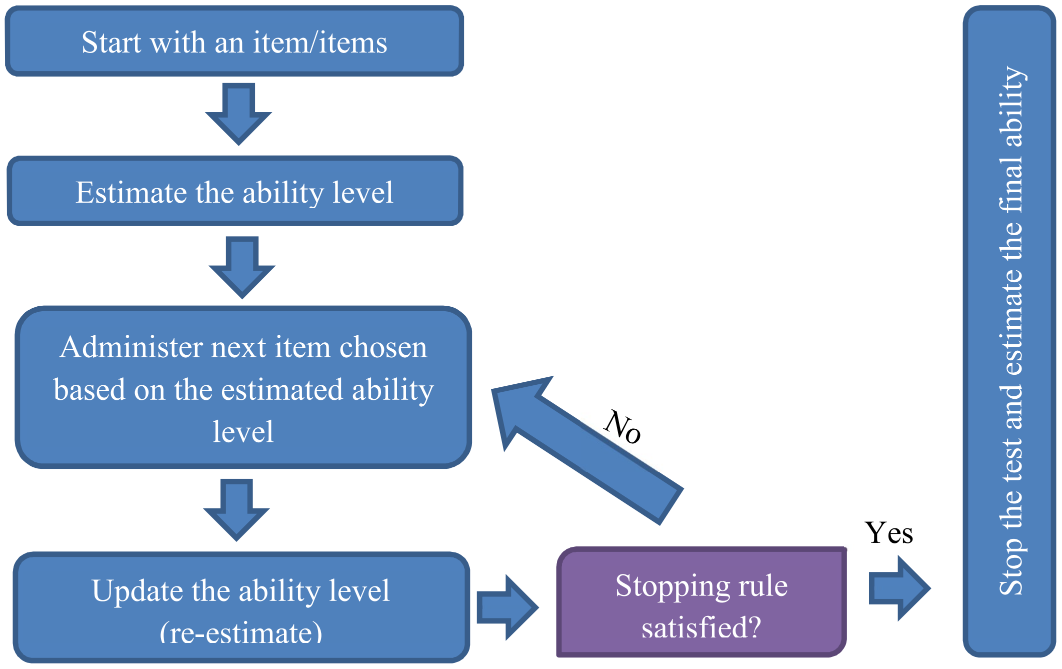

Start with an item/items
Estimate the ability level
Administer next item chosen based on the estimated ability level
No
Update the ability level (re-estimate)
Stopping rule satisfied?
Yes
Stop the test and estimate the final ability

Figure 1. Schematic representation of CAT process

Weiss and Kingsburry (1984) stated that there are six main components of computer adaptive test procedure. Those are (a) Item response model, (b) Item pool, (c) Starting rule, (d) Item selection rule, (e) Scoring rule and a (f) Termination criterion.

Item response model: In adaptive testing, different sets of items are administered to the examinees since each examinee takes items according to his/her proficiency level. Thus, their abilities are estimated independently of the particular selection of test items. IRT models are important since they give the opportunity to make the ability estimation independently of item selection. Thus, examinees taking different item sets can be compared (Lord, 1984; Magis, Duanli & von Davier, 2017).

Item pool: An item pool of large number of items including a wide range of item parameters is necessary. Since every test taker takes the item suitable for her/his ability, there should be items providing efficient measurement through all ability levels.

Starting rule: In that step, first items to start the test are defined. If the initial theta of examinee is known, test starts with the item/s suitable for that examinee's ability level. However, when that initial value is not known, researchers should assign an initial theta value in order to choose the first item/s. The most common way of that assignment is to assign the average ability level of population as initial theta (Thompson, 2007).

Item selection rule: After initial item/s are administered and the ability value is estimated, an item selection rule is required in order to choose the next item. There are lots of item selection methods but the most common ones are Maximum Fisher Information (MFI) and Bayesian methods (Wang, 2017; Weiss &Kingsbury, 1984).

Scoring rule: Ability estimation method should be specified in this step. Although different methods are used for ability estimation purposes, the most preferred ones are Maximum Likelihood Estimation (MLE) and Bayesian methods (EAP and MAP).

Termination criteria: As the last step, a criterion should be specified on where to stop the testing process. The possible criteria are test length or a value of standard error. If the length is specified, the testing process stops when the specified length is met. On the other hand, test continue until the desired measurement precision level is satisfied with 'standard error' criteria. Therefore, test takers may take different number of items on that criterion since different people can reach the criterion with different number of items (Thissen and Mislevy, 2000; Wang, 2017).

After that brief introduction on computer adaptive tests and its main components, demonstration of how to use R in CAT simulations was made with example R codes.

3. USING R IN CAT SIMULATIONS

In CAT simulations *catR* 3.16 version was used for demonstration purposes. The main functions for CAT simulations in catR are *randomCAT()* and *simulateRespondents()* functions. While *simulateRespondents()* function allows the multiple generation of CAT response patterns, *randomCAT()* results in generation of only one single response pattern. Since we are interested in the multiple generations at the same time in the context of this study, *simulateRespondents()* functions are explained and exemplified below.

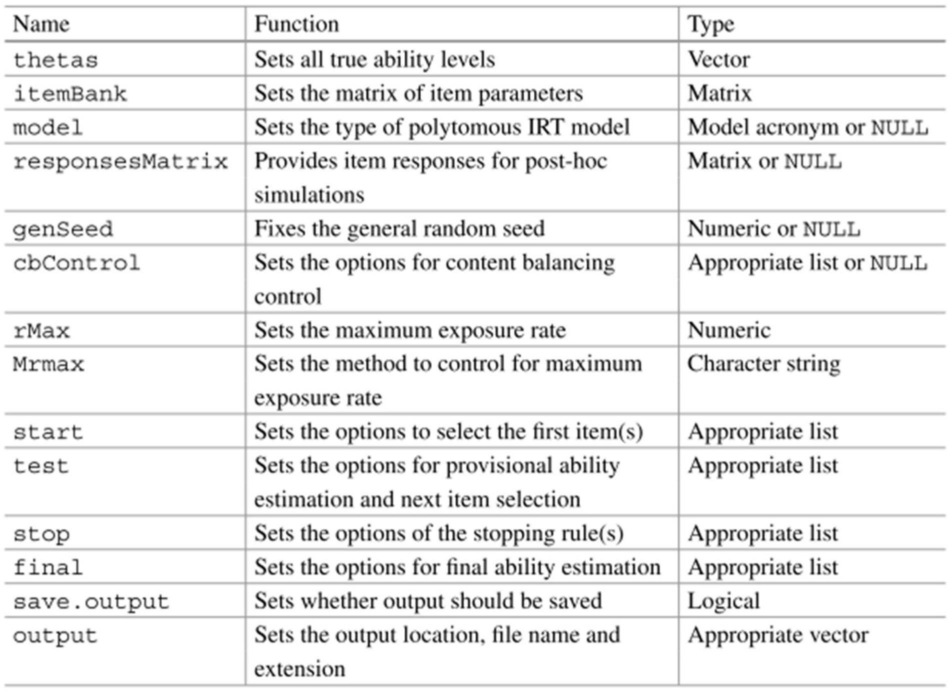
Input arguments in *simulateRespondents()* function are summarized by Magis, Yan and von Davier (2017) and given in Figure 2. Detailed information is provided for the related arguments inside the function.

In *thetas* argument, true ability levels of all examinees are provided in a vector form.

itemBank argument should be a matrix including the item parameters and include as many rows as item numbers. The column number of matrix differs according to the used IRT model.

model argument is NULL if the IRT model is dichotomous.

responsesMatrix includes item responses. If it is not specified, that matrix is randomly generated by using given item parameters. If the researcher is interested in Post-Hoc simulation, responses of each examinee to all questions should be provided in a matrix.



Name	Function	Type
`thetas`	Sets all true ability levels	Vector
`itemBank`	Sets the matrix of item parameters	Matrix
`model`	Sets the type of polytomous IRT model	Model acronym or `NULL`
`responsesMatrix`	Provides item responses for post-hoc simulations	Matrix or `NULL`
`genSeed`	Fixes the general random seed	Numeric or `NULL`
`cbControl`	Sets the options for content balancing control	Appropriate list or `NULL`
`rMax`	Sets the maximum exposure rate	Numeric
`Mrmax`	Sets the method to control for maximum exposure rate	Character string
`start`	Sets the options to select the first item(s)	Appropriate list
`test`	Sets the options for provisional ability estimation and next item selection	Appropriate list
`stop`	Sets the options of the stopping rule(s)	Appropriate list
`final`	Sets the options for final ability estimation	Appropriate list
`save.output`	Sets whether output should be saved	Logical
`output`	Sets the output location, file name and extension	Appropriate vector

Figure 2. Input arguments in *simulateRespondents()* function (Magis, Yan &von Davier, 2017)

Start function allows you to define the starting rule on the CAT process. In the start list,

fixItems: either researcher selects the items to be administered as the first items with that command or items are selected by another function and, in this case, that command is useless.

nrItems helps researchers to define number of starting items to be chosen randomly. Default is 1.

startSelect helps researchers to choose the item selection method.

randomesque is about item exposure issue. The number of specified items are picked up optimally before one single item is selected randomly.

theta a vector of the initial ability levels for selecting the first items

In the test list; details about ability estimators, the item selection rule and the item exposure issue are specified. Most commonly used arguments in the test list are given and explained below:

method: the ability estimation method is specified with method argument. There are four possible ability estimators; "BM", "ML", "EAP", "ROB". If the selected method is BM or EAP, priorDist and PriorPAr arguments are necessary.

itemSelect: Item selection rule during the process is chosen with itemSelect argument. Possible choices are "MFI", "bOpt", "thOpt", "MLWI", "MPWI", "MEPV", "MEI", "KL", "KLP", "progressive", "proportional" and "random".

randomesque: As in the start list, this argument is related to the item exposure issue. Specified number of items are chosen with the item selection rule and next item to be administered is pickep up among those randomly. Default value is 1.

The stop list consists of termination rule components. Most commonly used arguments are;

rule: This argument is necessary to specify how to stop the test. Possible choices are "length", "precision", "classification" and "minInfo". If we want to stop the test after the specified item number, "length" argument; to stop the test according to pre-specified precision level, "precision" argument is used.

thr: thr specifies the threshold related to the specified stopping rule. When the rule is length, thr indicates the maximal number of items to be used in the test.

rmax argument indicates the maximum exposure rate of an item. For instance, if it is set to 0.8, it means that an item in the pool can be administered to maximum 800 examinee out of 1000.

Output options will be explained by the examples.

3.1. Example 1

In the first example, a Monte Carlo simulation with 2000 examinees on an item pool of 300 items was presented step by step.

Since there was no item pool or theta values, those two were generated firstly.

Item Pool and Theta Generation

In order to generate item parameters *genDichoMatrix()* function can be used. The number of items, IRT model and parameter distributions are specified in that function. Although dichotomous IRT models were used for illustration in this study, *catR* package included polytomous models as well for those interested.

Dichotomous IRT model options: "1PL", "2PL", "3PL" or "4PL"

Distribution options: Normal distribution with *c("norm",mean, sd)*
Log-normal distribution with *c("lnorm", mean, sd)*
Uniform distribution with *c("unif", min, max)*
Beta distribution with *c("beta", alpha, beta)*

- *aPrior* may have normal (default), log-normal or uniform
- *bPrior* may have normal (default), or uniform
- *cPrior and dPrior* may have uniform or Beta distributions.

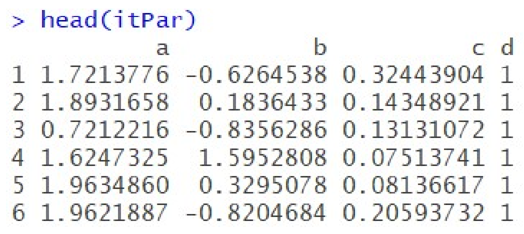
An item pool of 300 items was generated according to three-parameter logistic model. R codes are presented in Table 1.

Table 1. R codes used in item parameters and theta generation

```
#Firstly install package catR
install.packages("catR")

#Activate installed package
library("catR")

#The function genDichoMatrix creates a matrix of item parameters for dichotomous IRT models.
#Item pool is generated with 3PL. So a,b,c parameters should be specified.
itPar<-genDichoMatrix(items=300, model = "3PL", aPrior = c("unif", 0.5, 2),
                      bPrior = c("norm", 0, 1), cPrior = c("beta", 4, 16),
                      seed = 1)

#theta values were generated by using the standard normal distribution and stored in theta o
bject.
theta<-rnorm(2000,0,1)
```

Standard normal distribution for difficulty parameters, uniform distribution for discrimination parameters with min: 0.5 and max: 2.0 values and beta distribution for guessing parameters β(4,16) was used. *seed* function was used to get reproducible results. If random numbers are generated without seed function, different results are obtained in each attempt. If researchers want to get the same results in each trial, they should use the *seed* function.

As a result of running those codes, item parameters of 300 items generated according to 3 PL with specified parameters were obtained and those parameters were stored in 'itPar' object. Besides, theta values of 2000 people generated with standard normal distribution were stored in 'theta' object. Parameters of the first 6 items in the pool were obtained with *head()* function and given in Figure 3.



```
> head(itPar)
          a          b          c d
1 1.7213776 -0.6264538 0.32443904 1
2 1.8931658  0.1836433 0.14348921 1
3 0.7212216 -0.8356286 0.13131072 1
4 1.6247325  1.5952808 0.07513741 1
5 1.9634860  0.3295078 0.08136617 1
6 1.9621887 -0.8204684 0.20593732 1
```

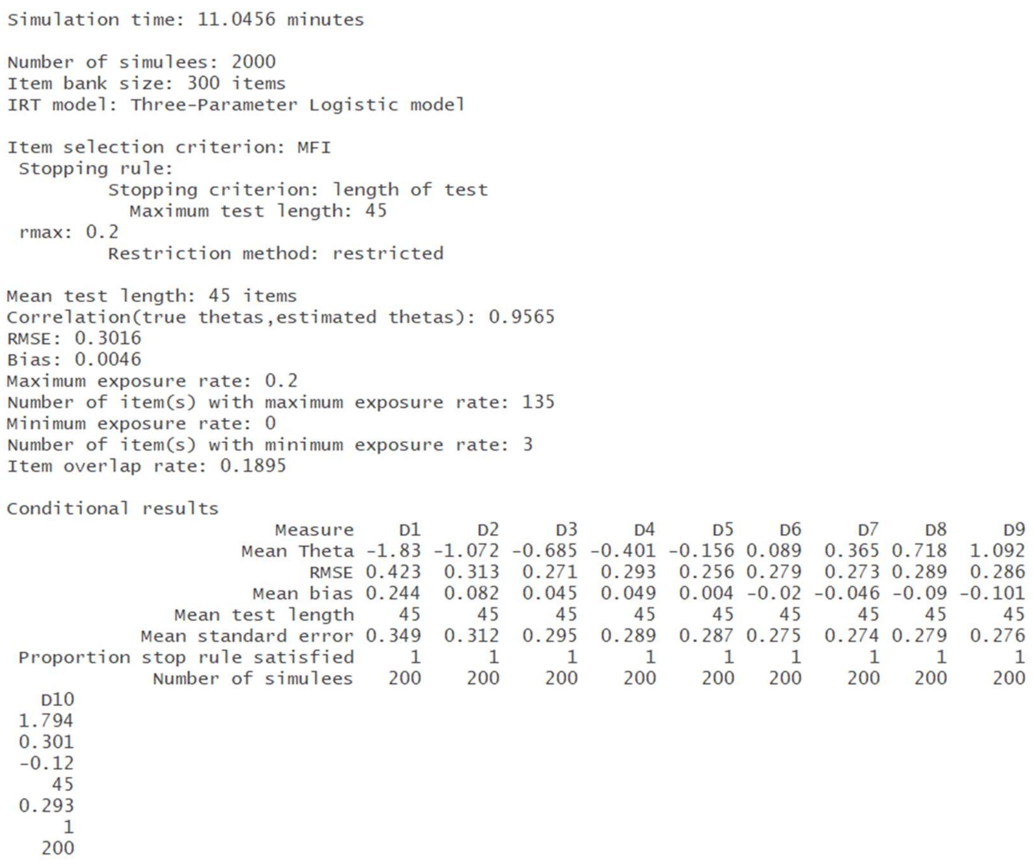
Figure 3. Generated item parameters of first six items

We have generated theta values and item parameters before and stored the data in 'theta' and 'itPar' objects respectively. Now, by using them, a CAT simulation was started with the codes below (Table 2). That simulation is about 45 item CAT application on 2000 examinees on an item pool of 300 items.

Table 2. R codes for CAT simulation

```
start <- list(nrItems=1, theta = 0, startSelect="MFI", randomesque = 10)
test <- list(method = "EAP", itemSelect = "MFI",  priorDist = "norm",
             priorPar = c(0, 1), randomesque = 10)
stop <- list(rule ="length", thr = 45)
final <- list(method = "EAP",  priorDist = "norm", priorPar = c(0, 1))

catResults1<- simulateRespondents(thetas = theta, itemBank = itPar,
                      rmax = 0.2, start = start, test = test, stop = stop,
                      final = final, save.output = TRUE,
                      output = c("","catR","txt"))
```

In this simulation; start, test, stop and final rules were specified first. Examinees started with one randomly chosen item among the 10 most informative items at the starting ability level zero since Maximum Fisher Information was used. In the test list, EAP and MFI were selected as ability estimation and item selection methods respectively. Again, 10 most optimal items were chosen and one of them was randomly chosen to be administered. As a stopping rule, length was specified and it was defined that test should be stopped when 45 items were reached. Lastly, the final ability was estimated with the EAP method again. Inside *simulateRespondents()* function, previously generated 'theta' and 'itPar' objects were used for theta and itemBank arguments. The maximum exposure rate was restricted to 0.2 which means that an item could be administered to 400 out of 2000 examinees. *save.output* argument was chosen TRUE since I wanted to save the output file and details were specified in *output* argument. *output* is a vector of three components and the first one is to define either the file path or "" (default). Second one is either the initial part of the output file name or "catR" and the third one is the file type which will be saved (either "txt" or "csv" (default)). Saving process results in three different files: "main", "responses", and "tables". "tables" and "responses" files include more detailed knowledge than "main" file. "responses" file indicates administered items, response pattern and estimated thetas after each item for each of 2000 examinees. "tables" file shows the true theta, estimated theta, final standard error and total number of administered items for each examinee. Lastly, "main" file includes general information about the process which is given in Figure 4. In case you don't use the save option, that is the same output come up in 'Console' in RStudio when simulation have been finished. Besides, simulation results were stored in 'catResults1' object.



```
Simulation time: 11.0456 minutes

Number of simulees: 2000
Item bank size: 300 items
IRT model: Three-Parameter Logistic model

Item selection criterion: MFI
 Stopping rule:
         Stopping criterion: length of test
           Maximum test length: 45
 rmax: 0.2
         Restriction method: restricted

Mean test length: 45 items
Correlation(true thetas,estimated thetas): 0.9565
RMSE: 0.3016
Bias: 0.0046
Maximum exposure rate: 0.2
Number of item(s) with maximum exposure rate: 135
Minimum exposure rate: 0
Number of item(s) with minimum exposure rate: 3
Item overlap rate: 0.1895

Conditional results
                        Measure    D1     D2     D3     D4     D5    D6     D7    D8     D9
                     Mean Theta -1.83 -1.072 -0.685 -0.401 -0.156 0.089  0.365 0.718  1.092
                           RMSE 0.423  0.313  0.271  0.293  0.256 0.279  0.273 0.289  0.286
                      Mean bias 0.244  0.082  0.045  0.049  0.004 -0.02 -0.046 -0.09 -0.101
               Mean test length    45     45     45     45     45    45     45    45     45
            Mean standard error 0.349  0.312  0.295  0.289  0.287 0.275  0.274 0.279  0.276
 Proportion stop rule satisfied     1      1      1      1      1     1      1     1      1
              Number of simulees  200    200    200    200    200   200    200   200    200
   D10
 1.794
 0.301
 -0.12
    45
 0.293
     1
   200
```

Figure 4. Main output come up after the simulation process in Example 1

This output includes summary statistics on average test length, correlation, RMSE and bias values and item exposure issues. The simulation process took 11.046 minutes on a computer with IntelCore i7-6700HQ processor. The mean test length was 45 since the test was specified as the fixed test length with 45 items. Correlation, bias and RMSE values were computed through all test takers and computed as 0.957, 0.005 and 0.302 respectively. Those values can be used in order to get information about test efficiency. Besides, different combinations for test designs for instance the effect of changing the item selection method and/or stopping criteria can be compared with the help of those values in terms of test efficiency. 135 out of 300 items in the pool had maximum exposure rate of 0.2 which meant that 135 items administered on 400 examinees and the usage of these items was restricted after that point. On the other hand, three items which had minimum exposure rate were not administered to anyone. In the conditional results part, examinees are divided into 10 equal intervals according to their ability levels; then mean theta, RMSE, bias, test length, standard error values were provided. As displayed in Figure 4, examinees were divided into ten subgroups of 200 people and detailed information on each interval was provided.

Another feature provided by catR package is graphical demonstration. Graphics related to the simulation results 'catResults1' can be obtained with the following code given in Table 3;

Table 3. R codes for simulation graphs

```
#There are different graph types. If type="all" is used, six different graphs come up. In c
ase only one specific graph is demanded, type should be changed.
plot(catResults1, type = "all")
```

The result of that command is given in Figure 5.

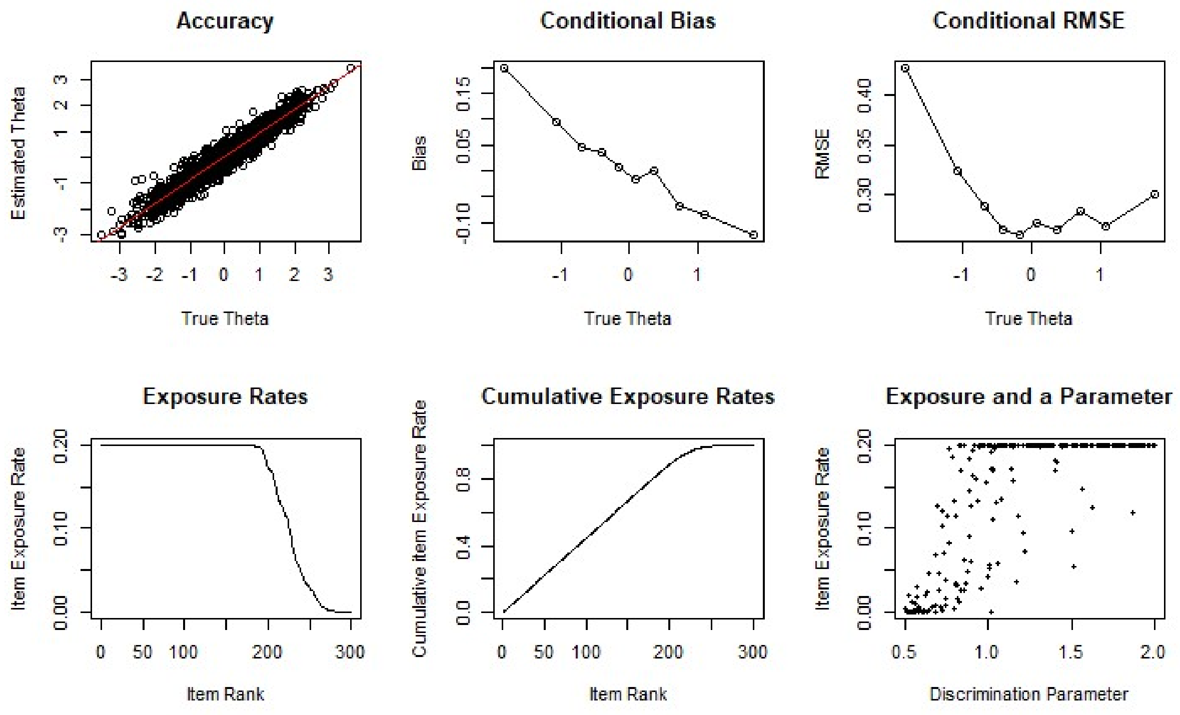


Figure 5. The graphical representation of simulation

Those are graphs which can be plotted for fixed test length. Some other graph types can also show up if stopping rule is changed. Graph types in Figure 5 are briefly explained below.

- Accuracy: In that plot, the scatter plot of true versus estimated abilities is presented. As can be seen in Figure 5, there is a strong positive relationship between true and estimated abilities.
- Conditional bias: True theta values' deciles are taken as x axis values and the graph is plotted as a function of bias and those deciles. As can be seen in the graph, bias value decreases with the increase of theta.
- Conditional RMSE: As in conditional bias graph, true theta deciles are in the x axis and the graph is plotted by using RMSE and the deciles. While RMSE had the lowest value at the left end of the true theta axis, it decreased with the increase of theta. After value of 0, it followed a waved way towards the right end of true thetas.
- Exposure rates: This graph indicates that the first 200 items in the item bank had maximum item exposure rate which was set as 0.20. After the first 200 items, the mentioned rate started to fall and it was zero finally, which means that last items were not used in the process.
- Cumulative exposure rates: As indicated in Figure 5, most of the required items were chosen through the first 200 items and the remaining ones were chosen from the last 100 items in the bank.
- Exposure and a parameter: That graph provided information on how discrimination values of the items in the bank affected the item exposure rate. As it can be seen, the items with low discrimination index have generally the lowest item exposure rate. High discriminating items were preferred more. It can be said that the higher item discrimination index, the higher item exposure rate.

In this example, a brief explanation of how to use R codes for a Monte Carlo simulation in computer adaptive testing was presented and related outputs were briefly mentioned. Response pattern was generated during CAT simulation by using the given item parameters and theta values. Next, another example the response pattern of which was not generated during the process and given by the researcher, was presented.

3.2. Example 2

Another example is presented here in order to illustrate how to use catR with a specific existing response pattern matrix that the researcher had. Besides, the termination rule was changed in order to see the different aspect other than those in Example 1. Since we didn't have a real response pattern matrix, a matrix was generated as an example in the first step.

Response Pattern Generation

After the generation of item parameters, item responses can be generated with *genPattern()* function as long as ability values are available. Theta values and item parameters are specified inside that function. By using 'theta' and 'itPar' objects generated in Example 1, the response patterns of 2000 people were generated and stored in *data* object. Related codes are presented in Table 4.

Table 4. R codes used in response pattern generation

```
#Before data generation, a 'data' object was defined in order to save generated patterns in
to it. Data matrix should have as many row as the length of theta and as many columns as th
e number of items.
data<-matrix(NA, length(theta), nrow(itPar))
for (i in 1:length(theta)){
  data[i,]<-genPattern(th = theta[i], itPar)}
```

genPattern function generates only one pattern according to the given item parameters and the specified theta in it. In order to generate 2000 examinees' response patterns *"for"* loop was used. A brief information on for loops and how they can be used are presented here.

for loops:

A loop is a way of automating a multi-step process that need to be repeated. It gives the chance of automate parts of the code. There are several ways of this automatization. Since *for* loops were used in the context of this study, an example of it was presented (Table 5).

Table 5. R codes for *for* loops

```
for (i in c(1:10)) print(i^2) #run

#output is like:

[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
[1] 36
[1] 49
[1] 64
[1] 81
[1] 100
```

In the example above, the square of the integers from 1 to 10 was calculated in one command. There is no need to write separate ten codes for all of the integers.

When we go back to our own example, data generation process was replicated 2000 times (the number of examinee) and the generated patterns were stored in *data* object. Now, *data* object has a matrix of 2000 rows (the number of examinees) and 300 columns (the number of items). Each row represents the response pattern of an examinee on 300 items.

There was no *responseMatrix* argument in Example 1 since it was generated during the process. However, in Example 2, it was thought that researcher had the test takers' responses to full item bank and tested the performance of CAT on these responses. So *responseMatrix* should be defined as indicating each row represents the response of an examinee. Missing responses are not accepted. The generated response pattern matrix 'data' was defined as *responsesMatrix*.

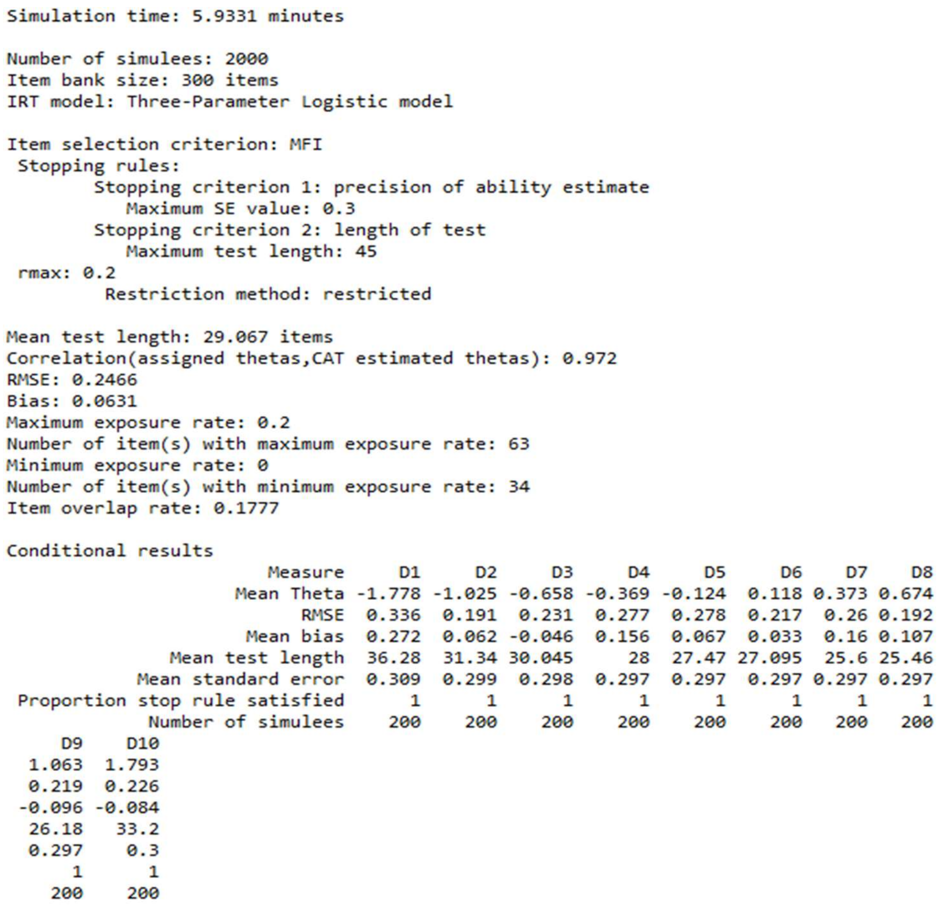
In order to see a different aspect, stopping rule was updated and rather than that everything was same such as ability estimation rule, item selection rule, item exposure issues etc. The code for the simulation process is given below in Table 6. The results were given in the output in Figure 6 below.

Table 6. R codes for the simulation

```
start <- list(nrItems=1, theta = 0, startSelect="MFI", randomesque = 10)
test <- list(method = "EAP", itemSelect = "MFI",  priorDist = "norm",
             priorPar = c(0, 1), randomesque = 10)

#The test stops either when the standard error reaches 0.3 or when 45 item is administered.
It is terminated on when one of the criterion is met.
stop <- list(rule =c("precision", "length"), thr = c(0.3, 45))
final <- list(method = "EAP",  priorDist = "norm", priorPar = c(0, 1))

# Maximum exposure rate was restricted to 0.2
catResults2<-simulateRespondents(thetas = theta, itemBank = itPar,
                                 responsesMatrix=data, rmax = 0.2,
                                 start = start, test = test, stop = stop,
                                 final = final)
```

As displayed in Figure 6, the mean test length was 29.067 items. Since both precision and length criteria were specified for the termination rule, the test was stopped when one of those criteria was met. Not all of the test takers needed to take 45 items and this decreased the average test length. The correlation between assigned thetas and the estimated thetas are 0.972 which indicates a strong positive relationship. 63 items had the maximum exposure rate and 34 of the items in the bank were never used. Plots of this simulation were obtained by using the *plot()* function as in the previous example and were given in Figure 7.

Since "precision" was added as a termination rule, four different graph types that could not be obtained for "length" rule were also obtained: "Stop rule satisfied", "Test length", "Conditional test length" and "Conditional standard error".



```
Simulation time: 5.9331 minutes

Number of simulees: 2000
Item bank size: 300 items
IRT model: Three-Parameter Logistic model

Item selection criterion: MFI
 Stopping rules:
        Stopping criterion 1: precision of ability estimate
           Maximum SE value: 0.3
        Stopping criterion 2: length of test
           Maximum test length: 45
 rmax: 0.2
         Restriction method: restricted

Mean test length: 29.067 items
Correlation(assigned thetas,CAT estimated thetas): 0.972
RMSE: 0.2466
Bias: 0.0631
Maximum exposure rate: 0.2
Number of item(s) with maximum exposure rate: 63
Minimum exposure rate: 0
Number of item(s) with minimum exposure rate: 34
Item overlap rate: 0.1777

Conditional results
                        Measure     D1     D2     D3     D4     D5     D6    D7    D8
                     Mean Theta -1.778 -1.025 -0.658 -0.369 -0.124  0.118 0.373 0.674
                           RMSE  0.336  0.191  0.231  0.277  0.278  0.217  0.26 0.192
                      Mean bias  0.272  0.062 -0.046  0.156  0.067  0.033  0.16 0.107
               Mean test length  36.28  31.34 30.045     28  27.47 27.095  25.6 25.46
            Mean standard error  0.309  0.299  0.298  0.297  0.297  0.297 0.297 0.297
 Proportion stop rule satisfied      1      1      1      1      1      1     1     1
             Number of simulees    200    200    200    200    200    200   200   200
     D9    D10
  1.063  1.793
  0.219  0.226
 -0.096 -0.084
  26.18   33.2
  0.297    0.3
      1      1
    200    200
```

Figure 6. Main output come up after the simulation process in Example 2

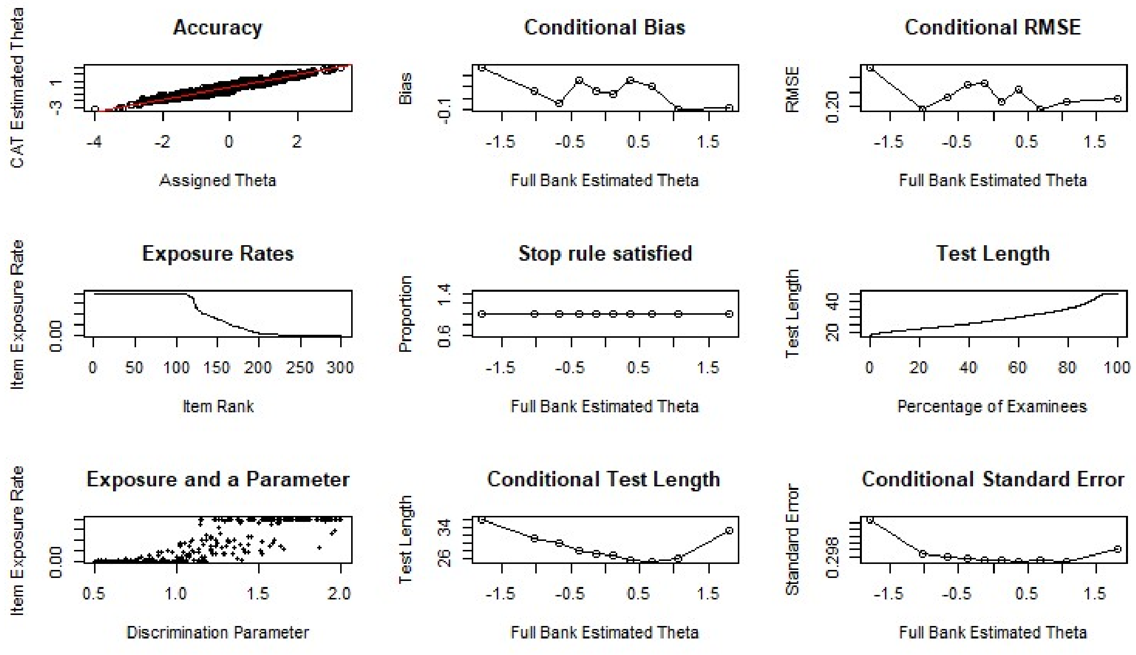


Figure 7. The graphical representation of simulation results

- Stop rule satisfied: That graph gives information about the proportion of test takers that the stopping rule is satisfied for. Since the length rule was included with the precision criteria, that proportion was 1 through all thetas.

- Test length: It indicates that how many percent of examinees took how many items.
- Conditional test length: Theta deciles are in the x axis and the graph is plotted by using average test length and the deciles. The maximum test length was obtained on the theta value interval less than -1.5. On the other hand, minimum test length was around the point of theta=0.5
- Conditional standard error: It provides information on average conditional standard error. As it can be seen in the graph, the average standard error was highest in the left end of theta continuum and the lowest one was around 0.5 value.

3.3. Example 3

Lastly, an example R code for replication purposes was illustrated. Let's think that we want to make 10 replications on the first example of which codes are given in Table 2. Everything is same with that code; however random generation of the item responses should be fixed with *genSeed* argument since the purpose was replication. By the way, several different functions can be used for replication purposes but the replication was made by using for loop in this example.

As illustrated in Table 7; start, test, stop and final functions were same as the first example. The difference appeared in the *simulateRespondents* function. The function was written in a *for* loop; and then *genSeed* argument was added in order to get the same item responses for each replication. *genSeed* should have the same length as thetas. Outputs were saved for each replication separately with *output* function. For instance, the output files of the first replication were saved as "cat-1.main", "cat-1.tables" and "cat-1.responses". Thus, results of each replication can be examined with those files.

Table 7. R codes for replication purposes

```
start <- list(nrItems=1, theta = 0, startSelect="MFI", randomesque = 10)
test <- list(method = "EAP", itemSelect = "MFI",  priorDist = "norm",
             priorPar = c(0, 1), randomesque = 10)
stop <- list(rule ="length", thr = 45)
final <- list(method = "EAP",  priorDist = "norm", priorPar = c(0, 1))

# r indicates the number of replication
# Different results for each replication is saved in different files with output command.
for (r in 1:10) {
  catResults3<-simulateRespondents(thetas=theta, itemBank=itPar,
                                   start = start, test = test, stop=stop,
                                   final = final, genSeed=1:length(theta),
                                   rmax=0.20, save.output = T,
                                   output=c("cat",-r,"catR","dat"))}
```

4. DISCUSSION and CONCLUSION

With its growing popularity, computer adaptive testing has an important place on educational measurement and psychometry. Since it provides a greater measurement precision with shorter test length in comparison to the linear tests, it attracts the attention of researchers and practitioners much. Besides, it is difficult to work with the real data by manipulating different conditions to find the best conditions. So, simulation studies have an important place in both CAT development and evaluation. In this study, some examples on how to use catR package in CAT simulations have been demonstrated. Given examples included how to make a simulation without having a response pattern, with an existing response pattern and how to make replication studies. Mainly used functions and the way how they were used were explained and the outputs were interpreted briefly. It is thought that this paper will help researchers interested in CAT simulations.

ORCID

Başak ERDEM-KARA https://orcid.org/0000-0003-3066-2892

5. REFERENCES

Bulut, O. & Sünbül, Ö. (2017). R programlama dili ile madde tepki kuramında monte carlo simülasyon çalışmaları [Monte carlo simulation studies in item response theory with the R programming language]. *Journal of Measurement and Evaluation in Education and Psychology*, *8*(3), 266-287. DOI: 10.21031/epod.305821

Choi, S. W. (2009). Firestar: Computerized adaptive testing simulation program for polytomous item response theory models. *Applied Psychological Measurement*, *33*(8), 644-645.

Embretson, S. E. & Reise, S. P. (2000). *Item response theory for psychologists*. Mahwah N.J.: L. Erlbaum Associates.

Han, K. T. (2012). SimulCAT: Windows software for simulating computerized adaptive test administration. *Applied Psychological Measurement*, *36*(1), 64-66.

Han, K. T. & Kosinski, M. (2014). Software tools for multistage testing simulations. In D. Yan, A. A. von-Davier & C. Lewis (Eds.), *Computerized multistage testing: Theory and applications* (p. 411–420). CRC Press: Taylor&Francis Group.

Hendrickson, A. (2007). An NCME instructional module on multistage testing. *Educational Measurement: Issues and Practice*, Summer 2007, 44-52.

Lee, S., Choi, Y.J., & Cohen, A. (2018). Automating simulation research for item response theory using R. *International Journal of Assessment Tools in Education*, *5*(4), 682-700. Retrieved from http://ijate.net/index.php/ijate/article/view/596

Lord, F. M. (1984). Standard errors of measurement at different ability levels. *ETS Research Reports*, *1984*(1), I-11. Retrieved from https://onlinelibrary.wiley.com/doi/epdf/10.1002/j.2330-8516.1984.tb00048.x

Magis D. & Raiche G. (2011). catR: An R package for computerized adaptive testing. *Applied Psychological Measurement*, 35, 576-577.

Magis, D., Yan, D. & von-Davier, A. (Eds.). (2017). *Computerized adaptive and multistage testing with R: Using packages catr and mstr*. Switzerland: Springer.

R Core Team. (2014). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from http://www.R-project.org/

Thissen, D. & Mislevy, R. J. (2000). Testing algorithms. In H. Wainer (Ed.), *Computerized adaptive testing: A primer (2. ed.)*. Mahwah N.J.: Lawrence Erlbaum Associates.

Thompson, N. A. (2007). A practitioner's guide for variable-length computerized classification testing. *Practical Assessment Research & Evaluation*, *12*(1). Retrieved from http://pareonline.net/getvn.asp?v=12&n=1

Wainer, H. (2000). Introduction and history. In H. Wainer (Ed.), *Computerized adaptive testing: A primer (2.ed., p. 1–22)*. Mahwah N.J.: Lawrence Erlbaum Associates.

Wainer, H. & Mislevy, R. J. (2000). Item response theory, item calibration, and proficiency estimation. In H. Wainer (Ed.), *Computerized adaptive testing: A primer (2. ed.)*. Mahwah N.J.: Lawrence Erlbaum Associates.

Wang, K. (2017). *A fair comparison of the performance of computerized adaptive testing and multistage adaptive testing* (Doctoral Dissertation). Michigan State University.

Weiss, D. J. & Kingsbury, G. G. (1984). Application of computer adaptive testing to educational problems. *Journal of Educational Measurement*, *21*(4), 361–375. https://doi.org/10.1111/j.1745-3984.1984.tb01040.x

Weiss, D. J. & Guyer, R. (2010). Manual for CATSim: Comprehensive simulation of computerized adaptive testing. St. Paul MN: Assessment Systems Corporation.

Yan, D., von-Davier, A. A. & Lewis, C. (2014b). Overview of computerized multistage tests. In D. Yan, A. A. von-Davier & C. Lewis (Eds.), *Computerized multistage testing*. CRC Press: Taylor&Francis Group.