Volume: 6 Issue: 5-Special Issue 2019

IJATE

International Journal of Assessment Tools in Education

http://ijate.net/

e-ISSN: 2148-7456

IJATE

e-ISSN 2148-7456

http://www.ijate.net/index.php/ijate/index

Volume 6 Issue 5 - Special Issue 2019

Dr. Izzet KARA
Editor in Chief
International Journal of Assessment Tools in Education

Pamukkale University,
Education Faculty,
Department of Mathematic and Science Education,
20070, Denizli, Turkey

Phone : +90 258 296 1036
Fax : +90 258 296 1200
E-mail : ijate.editor@gmail.com

Publisher : İzzet KARA
Frequency : 4 issues per year starting from June 2018 (March, June, September, December)
Online ISSN: 2148-7456
Website : http://www.ijate.net/index.php/ijate
http://dergipark.org.tr/ijate
Design & Graphic: IJATE

Support Contact
Dr. İzzet KARA
Journal Manager & Founding Editor
Phone : +90 258 296 1036
Fax : +90 258 296 1200
E-mail : ikara@pau.edu.tr

International Journal of Assessment Tools in Education

International Journal of Assessment Tools in Education (IJATE) is an international, peer-reviewed online journal. IJATE is aimed to receive manuscripts focusing on evaluation and assessment in education. It is expected that submitted manuscripts could direct national and international argumentations in the area. Both qualitative and quantitative studies can be accepted, however, it should be considered that all manuscripts need to focus on assessment and evaluation in education.
IJATE as an online journal is sponsored and hosted by **TUBITAK-ULAKBIM** (The Scientific and Technological Research Council of Turkey).

There is no submission or publication process charges for articles in IJATE.

IJATE is indexed in:

- Emerging Sources Citation Index (ESCI) (Web of Science Core Collection)
- TR Index (ULAKBIM),
- ERIH PLUS,
- DOAJ,
- Index Copernicus International
- SIS (Scientific Index Service) Database,
- SOBIAD,
- JournalTOCs,
- MIAR 2015 (Information Matrix for Analysis of the Journals),
- idealonline,
- CrossRef,
- ResearchBib,
- International Scientific Indexing

Table of Contents

Dear Researchers, Readers and Contributors,

As the *International Journal of Assessment Tools* (IJATE), we are thrilled to announce that the 2019 Special Issue has been published. The main theme of the special issue was "**Promoting Free/Libre Software Use in Educational Measurement**". The target was to promote and spread free/libre software use in the field of educational measurement and our aim was to set up a link between programmers and practitioners.

As the IJATE Editorial Team, we would like to thank all authors that submitted a paper, for their interest in the journal. We are happy to see that the special issue brought too much voice, and we received too many e-mails, questions and submissions. However, we had to consider the ones that met submission guidelines, aim and scope of the special issue. Our acceptance rate for this special issue was 50%.

We also would like to thank our reviewers for the effort and expertise that they contribute to reviewing, without which it would be impossible to maintain the high standards of our journal.

I am indebted to the Editor-in-Chief, Prof. Izzet Kara, Associate Editors, Drs. Eren Can Aybek and Ozen Yildirim for the tremendous support that they provided. It would not be possible to publish this issue without their encouragement, support and sincere. The success of this journal is indicative of these three researchers' efforts.

We hope you enjoy reading the papers published in the special issue and benefit from them. I encourage you to continue to send us your invaluable feedback and ideas for further improvement of your journal.

Dr. Halil Ibrahim Sari

Editor of Special Issue

International Journal of Assessment Tools in Education
2019, Vol. 6, No. 5-Special Issue, 1–19
https://dx.doi.org/10.21449/ijate.527299
Published at http://www.ijate.net http://dergipark.gov.tr/ijate *Research Article*

Computation of the Response Similarity Index M4 in R under the Dichotomous and Nominal Item Response Models

Cengiz Zopluoglu [1,*]

[1] School of Education and Human Development, University of Miami, USA

ARTICLE HISTORY

Received: 26 October 2018
Revised: 12 February 2019
Accepted: 14 February 2019

KEYWORDS

response similarity,
M4,
test fraud,
item response theory,
test security

Abstract: Unusual response similarity among test takers may occur in testing data and be an indicator of potential test fraud (e.g., examinees copy responses from other examinees, send text messages or pre-arranged signals among themselves for the correct response, item pre-knowledge). One index to measure the degree of similarity between two response vectors is M4 proposed by Maynes (2014). M4 index is based on a generalized trinomial distribution and it is computationally very demanding. There is currently no accessible tool for practitioners who may want to use M4 in their research and practice. The current paper introduces the M4 index and its computational details for the dichotomous and nominal item response models, provides an R function to compute the probability distribution for the generalized trinomial distribution, and then demonstrates the computation of the M4 index under the dichotomous and nominal item response models using R.

1. INTRODUCTION

In an era of high-stakes testing, maintaining the integrity of test scores has become an important issue and another aspect of test score validity. Unusual response similarity among test takers is a type of irregularity which may occur in testing data and be an indicator of potential test fraud such as sharing item responses among students during an exam, coaching of students by a teacher or a test proctor during an exam, or item pre-knowledge. In order to identify unusual response similarity among examinees, response similarity indices focus on the likelihood of agreement between two response vectors under the assumption of independent responding. The response indices differ in how they utilize the evidence of agreement and also in the reference statistical distribution used for computing the likelihood of observed agreement between two response vectors. For instance, while a well-known index developed by van der Linden and Sotaridona (2006) uses a generalized binomial distribution to model the number of all matching responses, the M4 index (Maynes, 2014) is using a generalized trinomial distribution to model the joint distribution of the number of matching correct responses and matching incorrect responses. In this paper, I first introduce the computational details of the M4 index as provided

CONTACT: Cengiz Zopluoglu ✉ c.zopluoglu@miami.edu School of Education and Human Development, University of Miami, USA

ISSN-e: 2148-7456 /© IJATE 2019

by Maynes (2014, 2017), then discuss how it can be computed in R and illustrate its use under the dichotomous and nominal item response models.

2. THE M4 INDEX

Suppose that P_i and Q_i are two disjoint events and $R_i = P_i{}'Q_i{}'$, where P_i represents the probability of matching correct response, Q_i represents the probability of matching incorrect response, and R_i represents the probability of nonmatching response between two test takers for the *i*th item. By definition, we know that $R_i = 1 - (P_i + Q_i)$. The probability of observing *m* correct matches and *n* incorrect matches between two test takers for *I* items is equal to

$$T_I(m,n) = \sum_{a=n}^{I-m} \sum_{b=m}^{I-a} (-1)^{a+b-m-n} \binom{b}{m} \binom{a}{n} S_{I;a,b}$$

where $S_{I;a,b} = \sum P_{u_1} P_{u_2} \ldots P_{u_b} Q_{v_1} Q_{v_2} \ldots Q_{v_a}$ and summation is extended over all possible pairs of disjoint subsets $\{u_1, u_2, \ldots, u_b\}$ and $\{v_1, v_2, \ldots, v_b\}$ of the set $\{1,2,\ldots,I\}$ (Charalambides, 2005). Maynes (2017) indicated that this quantity may be computed using a recursive formula as shown below:

$$T_{k+1}(m,n) = P_{k+1}T_k(m-1,n) + Q_{k+1}T_k(m,n-1) + R_{k+1}T_k(m,n)$$

with boundary conditions $T_0(0,0) = 1$ and $T_0(m,n) = 0$. The recursive formula starts with *k*=0 and ends with k=*I*-1. When $T_I(m,n)$ is computed for all possible combinations of *m* and *n*, the desired tail probability can be computed using a sub-ordering principle. First, the probabilities for all bivariate points (u,v) are added where *u* is greater than *m* and *v* is greater than *n*. Let this quantity be $D_{m,n}$. Then, all values of $T_I(a,b)$ where $D_{a,b} \geq D_{m,n}$ are found and summed up to obtain the desired tail probability.

2.1 Calculating the P and Q vectors using Item Response Models

In order to compute the M4 index, one has to obtain the vectors of probabilities for the correct match and incorrect match between two test takers. While these values could be empirically derived from a large dataset, they can be obtained based on item response models as this is a typical practice for other similar indices in the literature such as ω (Wollack, 1997) and generalized binomial test (van der Linden and Sotaridona, 2006).

2.2. Dichotomous Item Response Data

Suppose a researcher or practitioner has dichotomous item response data (e.g., 0/1, correct/incorrect, true/false) and wants to compute the M4 index. A variety of dichotomous IRT models are available for use depending on which one fits better to the data. The most general version of a dichotomous IRT model can be written as

$$\pi_{ij}(Y = 1|\theta_j, a_i, b_i, c_i, d_i) = c_i + (d_i - c_i)\frac{e^{a_i(\theta_j - b_i)}}{1 + e^{a_i(\theta_j - b_i)}},$$

where $\pi_{ij}(Y = 1|\theta_j, a_i, b_i, c_i, d_i)$ is the probability of correct response for the *j*th person on the *i*th item given the item and person parameters; a_i, b_i, c_i, and d_i are the discrimination, difficulty, guessing, and slipping parameters, respectively, for the *i*th item; and, θ_j is the person location parameter for the *j*th person. These parameters have to be estimated from the item response data prior to computing the M4 index. If the *d* parameter is fixed to one for all items, the model reduces to a 3-PL IRT model. In addition, if the *c* parameter is also fixed to zero for all items, the model reduces to a 2-PL IRT model. In addition, if the *a* parameter is constrained to be equal for all items, the model reduces to a 1-PL IRT model.

Once the item and person parameters are estimated, the probability of matching correct response on the ith item for two test takers, person j and person s, can be computed as

$$P_i = \pi_{ij}(Y = 1|\theta_j) \times \pi_{is}(Y = 1|\theta_s).$$

Similarly, the probability of matching incorrect response on the ith item for these test takers can be computed as

$$Q_i = \left(1 - \pi_{ij}(Y = 1|\theta_j)\right) \times (1 - \pi_{is}(Y = 1|\theta_s)).$$

Finally, the probability of not matching on the ith item can simply be computed as

$$R_i = 1 - (P_i + Q_i).$$

2.3. Nominal Response Data

Suppose that a researcher or practitioner has a multiple-choice test data with multiple response alternatives available for each item. One of these response alternatives is the correct response (key) and the remaining response alternatives are the incorrect responses (distractors). Note that there are a few number of alternative models proposed in the literature for such nominal response data (Bock, 1972; Penfield and de la Torre, 2008; Thissen & Steinberg, 1997). One can choose any of these models for modeling probabilities. We consider here the original Nominal Response Model (NRM; Bock, 1972) as it has been used in the literature for other indices and there are already available existing tools in R to reliably estimate the parameters of NRM. In NRM, the probability of selecting the kth response alternative among m alternatives of the ith item for the jth person is written as

$$\pi_{kij} = \frac{e^{\zeta_{ik}+\lambda_{ik}\theta_j}}{\sum_{k=1}^{m} e^{\zeta_{ik}+\lambda_{ik}\theta_j}},$$

where ζ_{ik} is the intercept and λ_{ik} is the slope parameter for the kth response alternative of the ith item, and θ_j is the person location parameter for the jth person.

Once the item and person parameters are estimated for NRM, the probability of matching correct response on the ith item for two test takers, person j and person s, can be computed as

$$P_i = \sum_{k=1}^{m} \pi_{kij} \times \pi_{kis} \times I(k = r_i),$$

where r_i is the correct response alternative for the ith item and $I(.)$ is an indicator variable that equals to 1 if the statement in parentheses is true, 0 otherwise. In a similar way, the probability of matching incorrect response on the ith item for these two test takers can be computed as

$$Q_i = \sum_{k=1}^{m} \pi_{kij} \times \pi_{kis} \times I(k \neq r_i).$$

The probability of not matching on the ith item can be computed as shown before.

3. R CODE FOR COMPUTING THE M4 INDEX

3.1. Computing the generalized trinomial distribution for a given P and Q vectors

Table 1 shows an R function to compute the joint distribution of matching correct and matching incorrect responses using the recursive algorithm. The function requires two vectors as input **P** and **Q**. **P** is a vector of probabilities for matching on a correct response and **Q** is a vector of probabilities for matching on an incorrect response for I items. Both vectors have a length of I. The function also requires two numbers m and n, m representing the observed number of correct matches and n is the observed number of incorrect matches between two test takers.

Table 1. *An R function to compute the generalized trinomial distribution and its tail probability given the vector of probabilities for two disjoint events and specified numbers.*

```
gtd <- function(P,Q,m,n) {

  R <- 1-(P+Q)
  I=length(P)

  rec <- vector("list",I+1)
  rec[[1]]=matrix(0,nrow=I+1,ncol=I+1)
  rec[[1]][1,1] <- 1
  for(k in 2:(I+1)){
    rec[[k]] = R[k-1]*rec[[k-1]]+
           rbind(0,P[k-1]*rec[[k-1]])[-(I+2),]+
           cbind(0,Q[k-1]*rec[[k-1]])[,-(I+2)]
  }

  for(k in 1:(I+1)){ rec[[k]]=t(rec[[k]])}

  upper <- matrix(nrow=I+1,ncol=I+1)
  for(x in 1:(I+1)){
    for(y in 1:(I+1)) {
      upper[x,y] = sum(rec[[I+1]][x:(I+1),y:(I+1)])
    }
  }

  prob.table <- expand.grid(0:I,0:I)
  colnames(prob.table) <- c("IncorrectMatch","CorrectMatch")
  prob.table <- prob.table[which(rowSums(prob.table)<=I),]
  prob.table <- prob.table[order(prob.table[,1]),]
  prob.table <- cbind(prob.table,0,0,0,0)
  prob.table[,3] <- I-(rowSums(prob.table[,1:2]))
  for(i in 1:(nrow(prob.table))){
    x=prob.table[i,1]
    y=prob.table[i,2]
    prob.table[i,4] <- upper[x+1,y+1]
    prob.table[i,5] <- rec[[I+1]][x+1,y+1]
  }

  for(i in 1:(nrow(prob.table))){
    r = prob.table[i,4]
    marked = which(prob.table[,4] <= r)
    prob.table[i,6] <- sum(prob.table[marked,5])
  }

  colnames(prob.table)[3:6] <- c("NonMatch","Upper",
                               "Probability","TailProbability")
  p = prob.table[which(prob.table[,1]==n & prob.table[,2]==m),6]
  list(prob.table[,-4],p)
}
```

This function returns a list with two elements. The first one is a table including the probabilities for the joint distribution of number of correct and incorrect matches (Probability column) and tail probabilities (Tail Probability column). The tail probability is the probability of observing the number of correct and incorrect matches or more extreme number of matches. The tail probability can be compared to an alpha level (e.g., .01) to make a decision about whether or not the observed similarity is significantly unusual under the assumption of independent responding.

Table 2. *An example of use for the R function to compute the generalized trinomial distribution and its tail probability given the vector of probabilities for two disjoint events and specified numbers.*

```
P <- c(0.45,0.60,0.30,0.55,0.58,0.42,0.60,0.25)
Q <- c(0.15,0.20,0.07,0.10,0.12,0.18,0.30,0.05)
M4 <- gtd(P, Q, m=3,n=2)
M4
 [[1]]
    IncorrectMatch CorrectMatch NonMatch   Probability TailProbability
 1               0            0        8 0.00014817600   1.00000000000
 10              0            1        7 0.00229866840   0.99985182400
 19              0            2        6 0.01383419478   0.99755315560
 28              0            3        5 0.04290090345   0.98371896082
 37              0            4        4 0.07560196143   0.88706918856
 46              0            5        3 0.07769363787   0.52528666715
 55              0            6        2 0.04534833951   0.21852241329
 64              0            7        1 0.01366565580   0.05554589926
 73              0            8        0 0.00162785700   0.00412010397
 2               1            0        7 0.00084360360   0.94081805737
 11              1            1        6 0.00964994464   0.93997445377
 20              1            2        5 0.04325532057   0.93032450913
 29              1            3        4 0.09890315828   0.81146722713
 38              1            4        3 0.12439288143   0.71256406885
 47              1            5        2 0.08552521188   0.36981797220
 56              1            6        1 0.02949871068   0.11179242607
 65              1            7        0 0.00393499620   0.01063825410
 3               2            0        6 0.00161592858   0.58817118742
 12              2            1        5 0.01406754191   0.58655525884
 21              2            2        4 0.04720104978   0.57248771693
 30              2            3        3 0.07777505708   0.44759302928
 39              2            4        2 0.06577034703   0.28429276032
 48              2            5        1 0.02674781613   0.08229371539
 57              2            6        0 0.00408353481   0.01472178891
 4               3            0        5 0.00147975807   0.17317407378
 13              3            1        4 0.00975200588   0.17169431571
 22              3            2        3 0.02374799388   0.16194230983
 31              3            3        2 0.02640188988   0.13819431595
 40              3            4        1 0.01320810993   0.04114389883
 49              3            5        0 0.00237669012   0.00670325790
 5               4            0        4 0.00073634463   0.04188024346
 14              4            1        3 0.00354190993   0.02793578890
 23              4            2        2 0.00583529943   0.02439387897
 32              4            3        1 0.00383679063   0.01855857954
 41              4            4        0 0.00084883518   0.00116303490
 6               5            0        3 0.00020646381   0.00432656778
 15              5            1        2 0.00067335552   0.00249224697
 24              5            2        1 0.00065585655   0.00181889145
 33              5            3        0 0.00019058868   0.00031419972
 7               6            0        2 0.00003170475   0.00012361104
 16              6            1        1 0.00006111576   0.00009190629
 25              6            2        0 0.00002628801   0.00003079053
 8               7            0        1 0.00000239652   0.00000450252
 17              7            1        0 0.00000203796   0.00000210600
 9               8            0        0 0.00000006804   0.00000006804

 [[2]]
 [1] 0.4476
```

Suppose that two test takers responded to eight items and we know the probability of matching correct and matching incorrect responses for each item (**P** and **Q** vectors). Also, suppose that these two test takers have the same correct answer for three items and the same incorrect response for two items. How likely this outcome would be? Table 2 presents the results obtained from the R function provided in Table 1 for this specific scenario.

The Table 2 indicates that observing three correct and two incorrect matches for a pair of test takers with the given **P** and **Q** is 0.0778. For the same pair, observing three correct and two incorrect matches or more extreme similarity is 0.4476. If we use a type-I error rate of 0.01, then we can decide that the response similarity between these two test takers is not significantly unusual because the tail probability is not smaller than .01.

3.2. Computing M4 for Dichotomous Data

For a given dichotomous dataset, the steps to compute the M4 statistics between two test takers are below:

1. Decision about the dichotomous IRT model to use. Researchers can choose a particular model based on their own judgement, or can fit all possible models and then empirically decide the best fitting model. The researchers are strongly encouraged to evaluate the plausibility of model assumptions such as unidimensionality and local independent before proceeding.
2. Estimation of item and person parameters based on the chosen dichotomous IRT model in Step 1.
3. Computation of the **P** and **Q** vectors for two test takers given their estimated person parameters and the estimated item parameters.
4. Computation of the tail probability for the observed number of correct and incorrect matches between these two test takers using the **gtd()** function introduced above.

For demonstration, I will use a dichotomous dataset that is publicly available on the following link https://itemanalysis.com/example-data-files/. This dataset includes binary responses to 56 items for 6,000 test takers. Table 3 and Table 4 shows the code to import the dataset and then fitting the 1-, 2-, and 3-PL IRT models using the **mirt** package (Chalmers, 2012) to decide the best fitting model. Based on the model fit indices, the best fitting model is the 3-PL IRT model.

Table 3. *R code to import the dataset and display the first few rows*

```
setwd("Path to file")  # Here you put the path to the folder for the dataset

exam1 <- read.csv("exam1_scored.txt") # Import the dataset

dim(exam1)      # Ask R to show the dimensions of the dataset

[1] 6000   56    # This indicates there are 6,000 rows and 56 columns


head(exam1,3)   # Ask R to display the first three rows of the dataset

   item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11
 1     1     0     0     0     1     1     1     1     0      1      1
 2     1     0     1     0     1     1     0     1     1      1      1
 3     1     0     0     1     0     0     0     1     1      1      1
   item12 item13 item14 item15 item16 item17 item18 item19 item20 item21
 1      1      1      1      1      1      1      1      1      0      0
 2      1      1      1      1      1      0      0      1      1      0
 3      1      1      0      1      1      0      0      1      1      0
```

Table 3. *Continued*

```
  item22 item23 item24 item25 item26 item27 item28 item29 item30 item31
1      1      0      1      1      1      0      0      1      0      0
2      1      1      1      0      1      0      1      0      0      1
3      0      1      1      0      1      0      0      0      1      0
  item32 item33 item34 item35 item36 item37 item38 item39 item40 item41
1      1      1      1      1      1      0      1      1      1      1
2      0      1      1      0      0      1      0      0      1      0
3      1      1      0      0      1      0      1      0      1      1
  item42 item43 item44 item45 item46 item47 item48 item49 item50 item51
1      1      1      0      1      0      0      0      0      0      1
2      0      1      0      0      0      0      0      0      0      1
3      0      1      0      0      1      1      0      1      1      1
  item52 item53 item54 item55 item56
1      0      1      0      0      1
2      0      1      1      1      0
3      1      1      0      1      1
```

Table 4. *R code to fit dichotomous IRT models, to choose the best fitting model, and to estimate item and person parameters for the best fitting model*

```
install.packages("mirt") # Install the mirt package into your computer
require(mirt)            # Load the library to the R session

# Fit 1PL model

  mod <- 'F = 1-56
         CONSTRAIN = (1-56,a1)'

    onePL <- mirt(data  = exam1, model = mod, itemtype="2PL",SE=TRUE)

# Fit 2PL  model

  twoPL <- mirt(data  = exam1, model = 1,itemtype="2PL",SE=TRUE)

# Fit 3PL model

  threePL <- mirt(data = exam1, model = 1,itemtype="3PL",SE=TRUE)

# Compare the model fit

  anova(onePL,twoPL)        # 1PL vs 2PL
  anova(twoPL,threePL)      # 2PL vs 3PL
      # 3 PL fits best.

# Item parameters for the 3PL model

  ipar <- coef(threePL,IRTpars=TRUE,simplify=TRUE)$items[,1:3]
  head(ipar,3) # display the item parameters for the first 3 rows

           a        b        g
 item1 1.0503 -0.37464 0.306052
 item2 0.6379 -0.02992 0.050127
 item3 1.5072 -1.27318 0.064474
# Estimate the ML theta estimates
```

Table 4. *Continued*

```
    mle <- fscores(threePL,method="ML") # This generates a 6000 x 1 matrix

    head(mle,3)        # display the ML theta estimates for the first 3 rows
            F1
 [1,]   0.32301
 [2,]  -0.07246
 [3,]   0.25915
```

Then, I save the estimated item parameters for the 3PL model into an object (*ipar*) and estimate the maximum likelihood person parameter estimates. Given these estimated item and person parameters based on the 3-PL model, suppose that we want to compute the M4 response similarity statistic for two test takers, subjects 1035 and 1567. We need to compute **P** and **Q** vectors. In order to compute the **P** and **Q** vectors, we have to compute the probability of correct response for each item for these two test takers using the estimated item parameters and their estimated person parameters. Table 5 shows the R code to compute the **P** and **Q** vectors for individuals 1035 and 1567 based on the estimated person parameters and item parameters.

These two test takers are matching on the correct response for 40 items and matching on the incorrect response for three items. Given their joint probability vectors for the correct and incorrect responses across all items (**P** and **Q** vectors), Table 6 shows how to use the **gtd()** function to compute the probability for the degree of the observed similarity or more extreme similarity between these two test takers. The tail probability is 0.557. This indicates that the observed similarity between these two test takers are not very unlikely. Therefore, we can conclude that there is no unusual degree of response similarity between the two test takers. It may be sometimes useful to visually present the results. Table 7 shows the R code to create a contour plot which was also discussed in Maynes (2017). In this contour plot, the boundary lines represent the likelihood of .01, .001, .0001, and .0001 for the number of correct and incorrect matches between two response vectors. In addition, the observed number of correct and incorrect matches is marked in the plot. One can easily demonstrate how likely the observed similarity is between two response vectors using this plot.

Table 5. *R code to compute the* ***P*** *and* ***Q*** *vectors for individuals 1035 and 1567 based on the estimated person parameters and item parameters*

```
# Estimated theta for Person 1035 and person 1567

   th1 <- mle[1035,1]

   th1
    F1
 1.415

   th2 <- mle[1567,1]

   th2
    F1
 1.577

 # A small function to compute the probability of correct response
 # given the item and person parameters for the 3-PL model



   prob <- function(ip,th){
     # ip - n x 3 item parameter matrix. Columns are a, b, g respectively
     # th - a numeric value

     ip[,3]+((1-ip[,3])*(1/(1+exp(-ip[,1]*(th-ip[,2])))))
   }


 # Probability of correct response across items for the two test takers

   P1 <- prob(ip=ipar,th=th1)    # Test taker 1035

   P1
  item1  item2  item3  item4  item5  item6  item7  item8  item9 item10
 0.9081 0.7296 0.9840 0.7483 0.8505 0.9322 0.7347 0.9950 0.7968 0.9924
 item11 item12 item13 item14 item15 item16 item17 item18 item19 item20
 0.9939 0.7983 0.9855 0.7202 0.9288 0.7388 0.6960 0.7229 0.9820 0.8225
 item21 item22 item23 item24 item25 item26 item27 item28 item29 item30
 0.8964 0.8104 0.7232 0.9633 0.8764 0.9615 0.3099 0.6112 0.7061 0.7084
 item31 item32 item33 item34 item35 item36 item37 item38 item39 item40
 0.5605 0.5130 0.9455 0.8687 0.9789 0.9653 0.8835 0.6294 0.5555 0.8618
 item41 item42 item43 item44 item45 item46 item47 item48 item49 item50
 0.9160 0.7597 0.8911 0.7167 0.9202 0.9097 0.8501 0.8702 0.7808 0.9943
 item51 item52 item53 item54 item55 item56
 0.9947 0.8833 0.8771 0.5813 0.7557 0.9771
   P2 <- prob(ip=ipar,th=th2)    # Test taker 1567
   P2
  item1  item2  item3  item4  item5  item6  item7  item8  item9 item10
 0.9208 0.7491 0.9874 0.7646 0.8650 0.9421 0.7658 0.9963 0.8196 0.9944
 item11 item12 item13 item14 item15 item16 item17 item18 item19 item20
 0.9955 0.8186 0.9889 0.7465 0.9420 0.7773 0.7228 0.7476 0.9874 0.8556
 item21 item22 item23 item24 item25 item26 item27 item28 item29 item30
 0.9173 0.8493 0.7807 0.9750 0.9040 0.9686 0.3673 0.6409 0.7358 0.7469
 item31 item32 item33 item34 item35 item36 item37 item38 item39 item40
 0.5960 0.5637 0.9583 0.8986 0.9857 0.9754 0.9059 0.6572 0.5921 0.8797
 item41 item42 item43 item44 item45 item46 item47 item48 item49 item50
 0.9302 0.7822 0.9076 0.7676 0.9413 0.9284 0.8868 0.9025 0.8103 0.9968
```

Table 5. *Continued*

```
  item51 item52 item53 item54 item55 item56
  0.9969 0.9092 0.9061 0.6091 0.7960 0.9841
# Joint probability of correct response across items (P vector)

    P <- P1*P2
    P
   item1  item2  item3  item4  item5  item6  item7  item8  item9 item10
  0.8362 0.5466 0.9716 0.5722 0.7357 0.8782 0.5626 0.9914 0.6530 0.9868
  item11 item12 item13 item14 item15 item16 item17 item18 item19 item20
  0.9894 0.6535 0.9746 0.5376 0.8750 0.5743 0.5031 0.5405 0.9696 0.7037
  item21 item22 item23 item24 item25 item26 item27 item28 item29 item30
  0.8223 0.6882 0.5647 0.9392 0.7923 0.9313 0.1138 0.3918 0.5195 0.5292
  item31 item32 item33 item34 item35 item36 item37 item38 item39 item40
  0.3341 0.2892 0.9061 0.7806 0.9649 0.9416 0.8003 0.4137 0.3290 0.7581
  item41 item42 item43 item44 item45 item46 item47 item48 item49 item50
  0.8521 0.5943 0.8088 0.5501 0.8662 0.8446 0.7539 0.7854 0.6326 0.9911
  item51 item52 item53 item54 item55 item56
  0.9916 0.8032 0.7947 0.3541 0.6015 0.9616

# Joint probability of incorrect response across items (Q vector)

    Q <- (1-P1)*(1-P2)
    Q
       item1      item2      item3      item4      item5      item6
  0.00727836 0.06782913 0.00020138 0.05923844 0.02017187 0.00392514
       item7      item8      item9     item10     item11     item12
  0.06213472 0.00001822 0.03665862 0.00004270 0.00002744 0.03658889
      item13     item14     item15     item16     item17     item18
  0.00016096 0.07092701 0.00412716 0.05816974 0.08425009 0.06993389
      item19     item20     item21     item22     item23     item24
  0.00022726 0.02563475 0.00855995 0.02857902 0.06068258 0.00091665
      item25     item26     item27     item28     item29     item30
  0.01186580 0.00120889 0.43662558 0.13959138 0.07765921 0.07378293
      item31     item32     item33     item34     item35     item36
  0.17755613 0.21249479 0.00227399 0.01332128 0.00030215 0.00085127
      item37     item38     item39     item40     item41     item42
  0.01096429 0.12702758 0.18128293 0.01663444 0.00586350 0.05232573
      item43     item44     item45     item46     item47     item48
  0.01005836 0.06584107 0.00468136 0.00646589 0.01696561 0.01265535
      item49     item50     item51     item52     item53     item54
  0.04159484 0.00001832 0.00001658 0.01059035 0.01153943 0.16365438
      item55     item56
  0.04983442 0.00036321
 # Observed number of correct matches between the two test takers

    m = sum (exam1[1035,]==1 & exam1[1567,]==1)
    m
  [1] 40

 # Observed number of incorrect matches between the two test takers

    n = sum (exam1[1035,]==0 & exam1[1567,]==0)
    n
  [1] 3
```

Table 6. *R code to compute the M4 response similarity index between examinees 1035 and 1567 based on the P and Q vectors computed from dichotomous item response data*

```
M4 <- gtd(P=P,Q=Q,m=m,n=n)

M4[[1]] # Probabilities for the trinomial distribution

     IncorrectMatch CorrectMatch NonMatch Probability TailProbability

1                 0            0       56   3.278e-46       1.000e+00
58                0            1       55   2.769e-43       1.000e+00
115               0            2       54   1.070e-40       1.000e+00
.............................................................................
3136              0           55        1   5.728e-09       4.383e-08
3193              0           56        0   2.057e-10       7.333e-10
2                 1            0       55   2.259e-45       9.985e-01
59                1            1       54   1.904e-42       9.985e-01
116               1            2       53   7.336e-40       9.985e-01
.............................................................................
3080              1           54        1   4.626e-08       3.585e-07
3137              1           55        0   1.814e-09       8.515e-09
3                 2            0       54   7.426e-45       9.685e-01
60                2            1       53   6.242e-42       9.685e-01
117               2            2       52   2.399e-39       9.685e-01
.............................................................................
3024              2           53        1   1.505e-07       9.657e-07
3081              2           54        0   6.304e-09       3.612e-08
.............................................................................
54               53            0        3  2.169e-106      6.136e-101
111              53            1        2  4.364e-104      6.136e-101
168              53            2        1  2.826e-102      6.131e-101
225              53            3        0  5.848e-101      5.849e-101
55               54            0        2  2.227e-109      1.082e-105
112              54            1        1  3.108e-107      1.082e-105
169              54            2        0  1.051e-105      1.051e-105
56               55            0        1  1.391e-112      1.021e-110
113              55            1        0  1.007e-110      1.007e-110
57               56            0        0  3.968e-116      3.968e-116

M4[[2]] # Tail Probability

[1] 0.5571
```

Table 7. *R code to create a contour plot for a visual representation of the result provided by the M4 index*

```
install.packages("lattice")

library(lattice)

obs <- c(40,3)

contourplot(TailProbability ~ CorrectMatch + IncorrectMatch,
            data=M4[[1]],
            labels=FALSE,
            xlab="Number of Correct Matches",
            ylab="Number of Incorrect Matches",
            panel=function(at,lty,...){
              panel.contourplot(at = .00001, lty = 1,...)
              panel.contourplot(at = .0001, lty = 2,...)
              panel.contourplot(at = .001, lty = 3,...)
              panel.contourplot(at = .01, lty = 4,...)
              panel.points(x=obs[1],y=obs[2], pch=15, cex=1)
             },
            key=list(corner=c(1,.9),lines=list(lty=c(1,2,3)),
            text=list(c("p=0.00001","p=0.0001","p=0.001","p=0.01"))),
            scales=list(y=list(at=seq(0,56,4)),x=list(at=seq(0,56,4)))
```

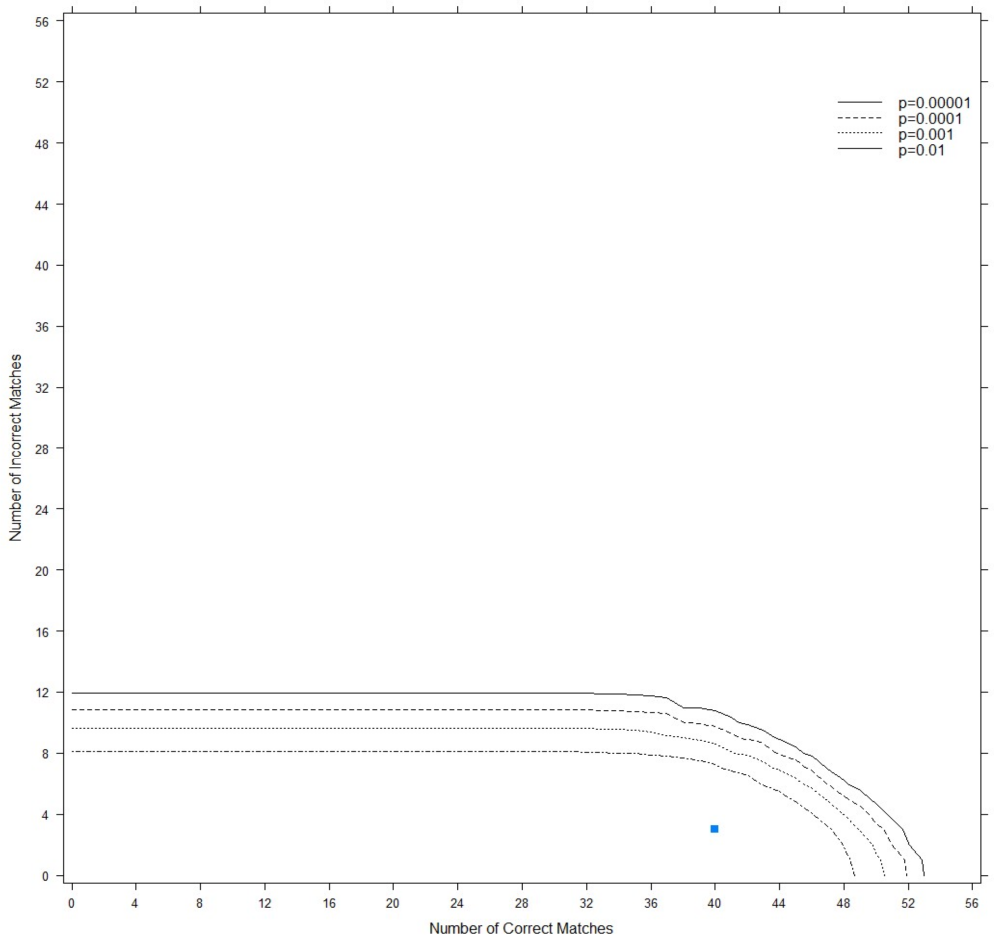


Figure 1. Contour plot for the joint probability distribution of correct and incorrect matches

3.3. Computing M4 for Nominal Response Data

The steps to compute the M4 index for nominal response data are identical to the dichotomous dataset. In particular, we are interested in multiple-choice test data where one of the response options is considered as the correct response (key) and other response options are considered as the incorrect responses (distractors). As mentioned before, there are a few number of alternative models proposed in the literature for multiple-choice test data (Bock, 1972; Penfield and de la Torre, 2008; Thissen & Steinberg, 1997). One can choose any of these models for modeling probabilities. We consider here the original Nominal Response Model (NRM; Bock, 1972). For this section, I will use the nominal version of the dichotomous dataset used before. We will also need a vector of correct response option for these 56 items. In order to fit NRM in the **mirt** package, we first transform these nominal A, B, C, and D response categories in the dataset to numbers 1, 2, 3, and 4, respectively. Then, we also need to recode data such that the correct response option is always assigned to the highest number possible (e.g., four in this case). Table 8 shows a compilation of R code to prepare the dataset for the data analysis.

Table 8. *R code to prepare nominal response data for data analysis*

```
# Import dataset

 exam1_nom <- read.csv("exam1_nominal.txt")

 dim(exam1_nom)
 [1] 6000   56

 head(exam1_nom,3) # display the first 3 rows of the dataset

   item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11
 1     A     A     D     D     C     B     C     D     D      D      C
 2     A     B     C     C     C     B     A     D     A      D      C
 3     A     C     D     B     D     D     D     D     A      D      C
   item12 item13 item14 item15 item16 item17 item18 item19 item20 item21
 1      A      D      C      A      B      D      B      A      B      B
 2      A      D      C      A      B      A      D      A      C      B
 3      A      D      D      A      B      A      A      A      C      B
   item22 item23 item24 item25 item26 item27 item28 item29 item30 item31
 1      A      D      B      C      B      A      C      A      C      A
 2      A      C      B      D      B      C      A      D      B      C
 3      C      C      B      B      B      B      D      D      A      A
   item32 item33 item34 item35 item36 item37 item38 item39 item40 item41
 1      B      B      A      B      D      C      A      D      C      D
 2      A      B      A      D      B      D      B      C      C      A
 3      B      B      C      C      D      B      A      C      C      D
   item42 item43 item44 item45 item46 item47 item48 item49 item50 item51
 1      A      B      C      C      C      D      B      D      D      D
 2      D      B      C      A      B      A      D      A      D      D
 3      D      B      D      A      D      B      D      C      B      D
   item52 item53 item54 item55 item56
 1      D      C      A      D      D
 2      C      C      B      A      C
 3      A      C      C      A      D

# Key response vector (correct responses for 56 items)

 key <- c("A","D","C","B","C","B","C","D","A","D","C","A","D","C",
         "A","B","D","B","A","C","A","A","C","B","C","B","D","A",
```

Table 8. *Continued*

```
         "A","A","C","B","B","A","B","D","D","A","D","C","D","A",
         "B","B","C","D","B","C","C","B","D","A","C","B","A","D")


# Recode A,B,C,D to 1,2,3,4

 for(i in 1:ncol(exam1_nom)){

   exam1_nom[,i]=ifelse(exam1_nom[,i]=="A",1,
                  ifelse(exam1_nom[,i]=="B",2,
                   ifelse(exam1_nom[,i]=="C",3,
                    ifelse(exam1_nom[,i]=="D",4,NA))))
 }

# Recode the vector of key responses

 new.key <- ifelse(key=="A",1,
                  ifelse(key=="B",2,
                   ifelse(key=="C",3,
                    ifelse(key=="D",4,NA))))

# Recode the data so that the correct option is always scored as 4

for(i in 1:ncol(exam1_nom)) {

  hold1 <- which(exam1_nom[,i]==new.key[i])
  hold2 <- which(exam1_nom[,i]==4)

  exam1_nom[hold1,i]= 4
  exam1_nom[hold2,i]= new.key[i]
}

head(exam1_nom,3) # display the first 3 rows of recoded data

   item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11
 1     4     1     3     2     4     4     4     4     1      4      4
 2     4     2     4     3     4     4     1     4     4      4      4
 3     4     3     3     4     3     2     3     4     4      4      4
   item12 item13 item14 item15 item16 item17 item18 item19 item20 item21
 1      4      4      4      4      4      4      4      4      2      2
 2      4      4      4      4      4      1      2      4      4      2
 3      4      4      3      4      4      1      1      4      4      2
   item22 item23 item24 item25 item26 item27 item28 item29 item30 item31
 1      4      3      4      4      4      1      3      4      3      1
 2      4      4      4      3      4      3      4      1      2      4
 3      3      4      4      2      4      2      1      1      4      1
   item32 item33 item34 item35 item36 item37 item38 item39 item40 item41
 1      4      4      4      4      4      3      4      4      4      4
 2      1      4      4      2      2      4      2      3      4      1
 3      4      4      3      3      4      2      4      3      4      4
   item42 item43 item44 item45 item46 item47 item48 item49 item50 item51
 1      4      4      3      4      3      2      2      3      2      4
 2      1      4      3      1      2      1      3      1      2      4
 3      1      4      2      1      4      4      3      4      4      4
   item52 item53 item54 item55 item56
 1      1      4      1      1      4
 2      3      4      4      4      3
 3      4      4      3      4      4
```

Once the dataset is prepared, we fit the nominal response model using the **mirt** package and extract the item parameters. In the nominal response model, each response category has one slope and one intercept parameter. Table 9 shows the R code to fit the model and estimate the item and person parameters. As it is seen, the item parameter matrix has eight columns with the first four columns (labeled as a1, a2, a3, and a4) are response category slope parameters and the last four columns (labeled as c1, c2, c3, and c4) are response category intercept parameters. Once these item parameters are obtained, we also estimate a person parameter for each individual based on maximum likelihood estimation.

Table 9. *R code to fit the nominal response model and estimate item and person parameters*

```
# Fit the Nominal Response Model

nrm <- mirt(exam1_nom, 1, 'nominal')

# Item parameter estimates

ipar.nrm <- coef(nrm, simplify=T, IRTpars = TRUE)$item

head(ipar.nrm,3) # display the first 3 rows of item parameter matrix

              a1       a2       a3     a4         c1      c2       c3     c4
 item1  0.29203 -0.36571 -0.69963 0.7733  0.0001801 -0.7811 -0.95435 1.7352
 item2  0.14899 -0.17617 -0.43625 0.4634 -0.3189671 -0.3362 -0.29905 0.9542
 item3 -0.71860 -0.45263 -0.17098 1.3422 -1.3477243 -0.8299 -0.37141 2.5490

# Person parameter estimates

theta.ML <- fscores(nrm,method="ML")

head(theta.ML,3) # display the first 3 rows of the person parameter matrix
             F1
 [1,]   0.21598
 [2,]  -0.32331
 [3,]  -0.05163
```

In order to compute the **P** and **Q** vectors based on nominal response data, we first need to create a function to compute the probability of selecting each response category on each item for a person given the nominal response model item parameter and the person parameter estimates. The R code in Table 10 takes the nominal response model estimated item parameter matrix obtained from the **mirt** package and the person parameter estimates for an individual as inputs and returns a matrix of probabilities for each response option on each item for the individual. For instance, we can see that the model predicted probabilities of choosing response categories 1, 2, 3, and 4 (correct response) for subject 1035 on the first item are .071, .011, .005, and .913, respectively. Similarly, the model probabilities of choosing response categories 1, 2, 3, and 4 (correct response) for subject 1567 on the first item are .050, .004, .002, and .944. Once the probability matrix for each subject is obtained, then the **P** vector, joint probability of matching on the correct response for each item, and **Q** vector, joint probability of matching on an incorrect response for each item, are computed. For instance, the probability of matching on the response category 4 (correct response) for subject 1035 and subject 1567 would be equal to 0.913*0.944 = 0.861 and the probability of matching on the response category 1, 2, or 3 would be (0.071*0.050 + 0.011*0.004 + 0.005*0.002) = 0.004. You can see at the end of Table 10 that we do this computation for each item and create the **P** and **Q** vectors.

After we obtain the **P** and **Q** vectors, we can now compute the M4 index for the same two test takers 1035 and 1567 using the nominal response data. Table 11 shows the R code to run *gtd()* function again taking the **P** and **Q** vectors, observed number of matches on correct responses (*m*), and observed number matches on incorrect responses (*n*) as inputs, and returns the generalized trinomial distribution for the number of correct and incorrect matches for every possible outcome. Also, the function returns the tail probability for observing more extreme similarity between two test takers. The tail probability is 0.9378 and can be compared to a conventional alpha level (e.g., 0.01) to make a decision about the degree of unusual similarity.

4. FINAL REMARKS

A very nice theoretical introduction and discussion of the M4 index have been provided by Maynes (2017); however, there has not been an accessible tool to compute the M4 index for other practitioners and researchers in the field of educational testing. The M4 index is a computationally demanding method. Its computation requires recursive algorithms that may not very easy to understand and implement. In this paper, I introduced an R function to compute the probabilities of the generalized trinomial distribution for two disjoint events, and demonstrated how this function can be used along with other R item response theory packages (e.g., mirt) to compute the M4 index under the dichotomous and nominal item response models. The availability of an open source computational tool will help the practitioners and the consumers of this index understand the nature of the M4 index better and will also help researchers conduct deeper investigations in the future about the properties of the M4 index under different conditions with real and simulated datasets.

Table 10. *R code to compute the **P** and **Q** vectors based on nominal response dataset for individuals 1035 and 1567 based on the estimated person parameters and item parameters*

```
# An internal function to compute the probability of choosing each response
# category of each item given the item parameter matrix and a person
# parameter estimate

irtprob <- function(th, item.param) {

      # Inputs:

            # item.param - n x 4 item parameter matrix.
              # First four columns are slopes, and the last four columns
are
              # intercepts

      # th - ability - a numeric value

   n.opt = ncol(item.param)/2
   prob <- matrix(nrow = nrow(item.param), ncol = ncol(item.param)/2)
     for (j in 1:ncol(prob)) {
              prob[,j] = exp((item.param[, j] * th) + item.param[,j +
n.opt])
     }
   prob <- prob/rowSums(prob)
   prob
  }

# Estimated theta for Person 1035 and person 1567 based on nominal response
# model
```

Table 10. *Continued*

```
   th1 <- theta.ML[1035,1]

   th1
   F1
1.692

   th2 <- theta.ML[1567,1]
   th2
   F1
2.514

 # Probability matrices. These have 56 rows, each row is representing an
item
 # They have four columns, each column is representing a response category

   P1 <- irtprob(item.param=ipar.nrm,th=th1)    # Test taker 1035
   P1
            [,1]       [,2]       [,3]    [,4]
  [1,] 0.07131223 0.0107270 0.0051265 0.9128
  [2,] 0.12456912 0.0706226 0.0472016 0.7576
  ..........................................
 [55,] 0.15651175 0.0552107 0.0473266 0.7410
 [56,] 0.00444109 0.0067377 0.0093141 0.9795
   P2 <- irtprob(item.param=ipar.nrm,th=th2)    # Test taker 1567
   P2
             [,1]       [,2]        [,3]    [,4]
  [1,] 0.049684924 0.0043540 0.00158164 0.9444
  [2,] 0.104793552 0.0454847 0.02455257 0.8252
  ............................................
 [55,] 0.102024390 0.0296282 0.02506444 0.8433
 [56,] 0.001038142 0.0017010 0.00266656 0.9946

 # Joint probability of correct and incorrect responses across items
 #(P and Q vector)

     # Note that in this recoded dataset used to fit the model,
     # we re-coded the correct response as 4 for all items
     # So, 1, 2, and 3 are incorrect responses.

      P <- P1[,4]*P2[,4]

      Q <- rowSums(P1[,1:3]*P2[,1:3])

 # Observed number of correct matches between the two test takers

   m  <- sum((exam1_nom[1035,]==exam1_nom[1567,] &
             exam1_nom[1035,]==4)*1,na.rm=TRUE)

 # Observed number of incorrect matches between the two test takers

   n  <- sum((exam1_nom[1035,]==exam1_nom[1567,] &
             exam1_nom[1035,]!=4)*1,na.rm=TRUE)
```

Table 11. *R code to compute the M4 response similarity index between examinees 1035 and 1567 based on the P and Q vectors computed from nominal response data*

```
M4 <- gtd(P=P,Q=Q,m=m,n=n)

M4[[1]] # Probabilities for the trinomial distribution

     IncorrectMatch CorrectMatch NonMatch Probability TailProbability
1                 0            0       56   1.251e-50       1.000e+00
58                0            1       55   2.847e-47       1.000e+00
115               0            2       54   2.717e-44       1.000e+00
......................................................................
3136              0           55        1   5.428e-08       2.222e-07
3193              0           56        0   1.836e-09       8.750e-09
2                 1            0       55   3.301e-50       9.034e-01
59                1            1       54   7.505e-47       9.034e-01
116               1            2       53   7.153e-44       9.034e-01
......................................................................
3080              1           54        1   1.661e-07       9.390e-07
3137              1           55        0   6.320e-09       2.739e-08
3                 2            0       54   4.188e-50       5.360e-01
60                2            1       53   9.512e-47       5.360e-01
117               2            2       52   9.054e-44       5.360e-01
......................................................................
3024              2           53        1   2.111e-07       1.202e-06
3081              2           54        0   8.794e-09       5.272e-08
4                 3            0       53   3.405e-50       2.439e-01
61                3            1       52   7.727e-47       2.439e-01
118               3            2       51   7.345e-44       2.439e-01
......................................................................
2968              3           52        1   1.538e-07       7.729e-07
3025              3           53        0   6.912e-09       3.431e-08
......................................................................
56               55            0        1  3.362e-144      1.259e-141
113              55            1        0  1.256e-141      1.256e-141
57               56            0        0  9.791e-149      9.791e-149

M4[[2]] # Tail Probability
[1] 0.9378
```

ORCID

Cengiz Zopluoglu https://orcid.org/0000-0002-9397-0262

5. REFERENCES

Bock, R. D. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, *37*(1), 29-51.

Chalmers, R.P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software, 48(6)*, 1 - 29. URL http://www.jstatsoft.org/v48/i06/

Charalambides, C. A. (2005). *Combinatorial methods in discrete distributions* (Vol. 600). John Wiley & Sons.

Gabriel, T. (2010, December 27). Cheaters find an Adversary in Technology. *The New York Times*. Retrieved from https://www.nytimes.com/2010/12/28/education/28cheat.html

Maynes, D.D. (2014). Detection of non-independent test taking by similarity analysis. In N. M. Kingston and A. K. Clark (Eds.) Test fraud: Statistical detection and methodology (pp. 53-82). Routledge: New York, NY.

Maynes, D. D. (2017). Detecting potential collusion among individual examinees using similarity analysis. In GJ Cizek and JA Wollack (eds.), *Handbook of quantitative methods for detecting cheating on tests*, Chapter 3, 47-69. Routledge, New York, NY.

Penfield, R. D., de la Torre, J., & Penfield, R. (2008). A new response model for multiple-choice items. Presented at the annual meeting of the National Council on Measurement in Education, New York.

Thissen, D., & Steinberg, L. (1984). A response model for multiple choice items. *Psychometrika*, *49*(4), 501-519.

van der Linden, W. J., & Sotaridona, L. (2006). Detecting answer copying when the regular response process follows a known response model. *Journal of Educational and Behavioral Statistics*, *31*(3), 283-304.

Wollack, J. A. (1997). A nominal response model approach for detecting answer copying. *Applied Psychological Measurement*, *21*(4), 307-320.

International Journal of Assessment Tools in Education
2019, Vol. 6, No. 5-Special Issue, 20–36
https://dx.doi.org/10.21449/ijate.627361

Published at http://www.ijate.net http://dergipark.gov.tr **Research Article**

Educational data mining: A tutorial for the *rattle* package in R

Okan Bulut [1,*], **Hatice Cigdem Yavuz** [2]

[1] Centre for Research in Applied Measurement and Evaluation, University of Alberta, Edmonton, AB, Canada
[2] Cukurova University, Faculty of Education, Sarıçam/Adana, Turkey

ARTICLE HISTORY

Received: 01 October 2018

Accepted: 05 December 2019

KEYWORDS

Educational data mining,
rattle,
Decision tree,
Random forest,
Support vector machines

Abstract: Educational data mining (EDM) has been a rapidly growing research field over the last decade and enabled researchers to discover patterns and trends in education with more sophisticated methods. EDM offers promising solutions to complex educational problems. Given the rapid increase in the availability of big data in education and software programs to analyze big data, the demand for user-friendly, free software programs to implement EDM methods also continues to increase. The R programming language has become a popular environment for data mining due to its availability and flexibility. The *rattle* package in R contains a set of functions to implement data mining with a graphical user interface. This study demonstrates three widely used data mining algorithms (classification and regression tree, random forest, and support vector machine) in EDM using real data from the 2015 administration of the Programme for International Student Assessment (PISA). First, a brief introduction to EDM is provided along with the description of the selected data mining algorithms. Then, how to perform data mining analysis using the *rattle*'s graphical user interface is demonstrated. The study concludes by comparing the results of the selected data mining algorithms and highlighting how those algorithms can be utilized in the context of educational research.

1. INTRODUCTION

As an interdisciplinary field, educational data mining (EDM) refers to the development and use of advanced statistical methods to explore and identify patterns and relationships in data derived from educational settings. EDM aims to implement advanced machine learning and data mining algorithms (1) to exploit unprocessed data from educational settings (e.g., large-scale assessments, records of students' academic progress in school, and log data from e-learning systems), (2) to discover relations, patterns, and trends in education, and (3) to use the discovered information in order guide and improve the decision-making process in educational practices. The increasing availability and popularity of big data in education has created new pathways for educational researchers who are interested in applying EDM methods to find solutions for various problems in education – such as enhancing the quality of online learning environments (Ducange, Pecori, Sarti, & Vecchio, 2016), early prediction of student dropouts

CONTACT: Okan Bulut ✉ bulut@ualberta.ca Centre for Research in Applied Measurement and Evaluation, University of Alberta, Edmonton/Alberta, Canada

(Aulck, Velagapudi, Blumenstock, & West, 2016), and forecasting students' academic performance and identifying students who might be at risk of academic failure (Hussain, Zhu, Zhang, Abidi, & Ali, 2019).

Educational researchers who intend to use the EDM methods typically follow a deductive reasoning approach in which they first collect or get access to large volumes of data, explore the data visually and statistically, and then do further investigations in order to discover hidden patterns and relationships in the data. Unlike theory-driven educational research that usually aims to obtain evidence supporting a priori hypothesis, the primary goal of a typical EDM process is to find and extract new knowledge from the data without a particular priori hypothesis and to use the discovered information for the purpose of building new theory, if possible. In the context of EDM, educational researchers' interests are mainly focused on several dimensions, such as learning, predictive, behavioral, and visual analytics (Aldowah, Al-Samarraie, & Fauzy, 2019). Recent systematic review studies have also highlighted a vast and growing body of research on EDM and its applications in various areas of education (e.g., Aldowah et al., 2019; Baker, Martin, & Rossi, 2017; Dutt, Ismail, & Herawan, 2016; Peña-Ayala, 2014). The findings of these review studies reveal that educational researchers will continue to harness the power of EDM for solving complex problems in education with the availability of big data in education.

From the methodological point of view, EDM methods are the same as the data mining methods utilized in other scientific fields (e.g., business, finance, medicine, and agriculture). The current data mining methods can be categorized into two main types according to the availability of a target (i.e., dependent) variable in the data: supervised (also known as predictive) and unsupervised (also known as descriptive). Supervised data mining methods are appropriate when the researcher wants to predict a specific target variable that is already available in the data. Typical examples of supervised data mining applications include regression and classification tasks where the researcher wants to predict either a categorical (classification) or continuous (regression) variable using a set of predictors (i.e., features) available in the data. Unsupervised data mining methods are appropriate when the goal is to find hidden structures or relations in the data instead of predicting a target variable. Common examples of unsupervised data mining applications include clustering, association rule mining, and dimensionality reduction. A detailed review of data mining methods commonly used in educational research can be found in Aldowah et al. (2019) and Peña-Ayala (2014).

In education, researchers and practitioners are often interested in research problems in which the primary goal is the prediction of an outcome (i.e., dependent) variable from a set of predictors (Berland, Baker, & Blikstein, 2014; Sinharay, 2016). Therefore, EDM applications mostly involve supervised data mining methods, instead of unsupervised data mining methods. Previous research indicated that the supervised data mining methods often provide higher prediction accuracy than traditional methods, such as multiple linear and logistic regression (e.g., Fernández-Delgado, Cernadas, Barro, & Amorim, 2014; Koon & Petscher, 2015, 2016; Spikol, Ruffaldi, Dabisias, & Cukurova, 2018). This study focuses on three data mining algorithms that can be used for both classification and regression problems: classification and regression trees (CART; Breiman, Friedman, Olshen, & Stone, 1984), random forest (RF; Breiman, 2001), and support vector machines (SVM; Cortes & Vapnik, 1995). These algorithms have been widely used in previous EDM research due to their relatively lower complexity and ease of implementation and interpretation (e.g., Guruler, Istanbullu, & Karahasan, 2010; Ivancevic, Celikovic, & Lukovic, 2011; Mccuaig & Baldwin, 2012; Pardos, Wang, & Trivedi, 2012).

The CART algorithm relies on stratifying a large dataset into a number of smaller subsets in which separate regression models can be built for either continuous or categorical outcome variables. Then, the model provides a set of classification or regression rules in a decision tree based on the nodes generated from the utilized predictors (Agarwal, Pandey, & Tiwari, 2012). As a nonparametric approach, CART does not make explicit assumptions about the distributions of variables, and thus it can produce relatively more accurate predictions (e.g., Strobl, 2013). The RF algorithm is similar to the CART algorithm in terms of relying on a regression or classification tree model for prediction. However, unlike CART, the RF algorithm generates many decision trees and combines all of them for making a final prediction (Breiman, 2001). Therefore, the RF algorithm can overcome many estimation issues (e.g., instability, high bias, and under-representation of classifications) in the CART algorithm because predictions are made based on the combination of many tree models that are generated differently using bootstrap samples, instead of a single decision tree model based on the entire sample (Sinharay, 2016; Williams, 2011). Differently from the previous two algorithms, the SVM algorithm relies on creating a separating hyperplane in an *N*-dimensional prediction space where *N* refers to the number of available predictors in the data. A hyperplane can be considered as a decision boundary that helps separate or classify the data points. If the outcome variable is categorical, then the hyperplane aims to create classes having the maximum distance between each other (Williams, 2011). If, however, the outcome variable is continuous, then the hyperplane creates a regression line (or plane) that can minimize the difference between the predicted and original values of the outcome variable.

Currently, there are many software programs that are capable of implementing the data mining algorithms mentioned above – such as RapidMiner, Weka, KEEL, KNIME, Orange, Python, R, and IBM SPSS Modeler (see Slater, Joksimović, Kovanovic, Baker, and Gasevic [2017] for a detailed review). Some of these programs (e.g., RapidMiner, Weka, and IBM SPSS Modeler) provide a graphical user interface (GUI) for users to easily select an algorithm along with the type of data mining analysis that they want to perform. Compared to these software programs, advanced programming languages such as Python and R (R Core Team, 2019) can provide users with more sophisticated tools to explore, organize, visualize, and model the data within the same computing environment. However, the amount of time that it takes to learn a new programming language and to achieve expertise in it can be very long for novice users who do not have any previous experience in programming. An exception in this situation is the *rattle* package (Williams, 2011) that provides a user-friendly GUI to perform data mining analysis within the R statistical computing environment (R Core Team, 2019). The *rattle* package can perform data mining analysis using a variety of advanced algorithms. The purpose of this study is to demonstrate how to use the *rattle* package for performing data mining analysis. Using a real dataset from a large-scale international assessment, the implementation of the CART, RF, and SVM algorithms using the *rattle* package is demonstrated. The steps for building and evaluating a predictive model in the *rattle* are also described in detail.

2. METHOD

2.1. Study Group

The sample of this study comes from the 2015 administration of the Organisation for Economic Co-operation and Development's (OECD) Programme for International Student Assessment (PISA). PISA is a large-scale, international assessment program that assesses the extent to which 15-year-old students have acquired adequate competency in various subject areas such as reading, mathematics, and science (OECD, 2018). The 2015 administration of PISA involved approximately 540,000 15-year-old students from 72 participating countries and economies. The sample of this study consists of 5896 students (49.83 % female) who participated in PISA 2015 from Turkey. This study uses the PISA dataset for the demonstration of the *rattle* package because the dataset is publicly available through the OECD website (http://www.oecd.org/pisa/) and it consists of many categorical and continuous variables from students and schools – which creates a large-size database suitable for an EDM research study.

2.2. Measures

2.2.1. *Scientific Literacy Test in PISA 2015*

The primary focus of PISA 2015 was to assess students' scientific literacy as well as their attitudes and preferences regarding learning experiences in science. The results of PISA 2015 suggest that there is a large variation in students' competency levels in science and that this variation can be explained by many factors, such as demographic variables, socioeconomic status, students' participation in science-related activities, and the opportunity to learn science at school (Mostafa, Echazarra & Guillou, 2018; OECD, 2018). In this study, students' performance levels in scientific literacy were obtained from the PISA 2015 Scientific Literacy Test. The scientific literacy test was designed to assess three major competencies: explaining phenomena scientifically, evaluating and designing scientific inquiry, and interpreting data and evidence scientifically (OECD, 2017). Moreover, 36% of the items in the test were in physical, 36% in living, 28% in earth and space context. Students' scores obtained from the test were scaled with a mean of 500 and a standard deviation of 100. The average scientific literacy score in PISA 2015 was 493 across all participating countries. Using this score as a cutoff value, a categorical variable (science_perf) was created. For students whose scores were equal or higher than 493, science_perf was labeled as "High". If, however, students' scores were less than 493, then the label of "Low" was assigned to science_perf. The resulting categorical variable was used as the outcome variable in the data mining analysis.

2.2.2. *The student questionnaire*

The other variables regarding students (i.e., predictors) were obtained from the student questionnaire of PISA 2015. Table 1 shows the complete list of the variables used in this study.

Table 1. The list of the variables used in this study

Variable	Data type	Description
gender	Categorical	Female=1, Male=0
computer	Categorical	Owning a computer at home; Yes=1, No=0
software	Categorical	Owning software at home; Yes=1, No=0
internet	Categorical	Owning internet at home; Yes=1, No=0
desk	Categorical	Owning a desk at home; Yes=1, No=0
own.room	Categorical	Owning a room at home; Yes=1, No=0
quiet.study	Categorical	Owning a quiet study area at home; Yes=1, No=0
ANXTEST	Numeric	Test anxiety
COOPERATE	Numeric	Enjoying cooperation
EMOSUPS	Numeric	Parents emotional support
PARED	Numeric	Highest education of parents in years
TMINS	Numeric	Learning time in total
ESCS	Numeric	Index of economic, social and cultural status
TEACHSUP	Numeric	Teacher support in a science class
TDTEACH	Numeric	Teacher-directed science instruction
IBTEACH	Numeric	Inquiry-based science teaching and learning practices
SCIEEFF	Numeric	Science self-efficacy
science_perf	Categorical	If science scores >= 493, *High*; *Low* otherwise

2.3. Procedure

To use the *rattle* package, readers first need to download and install the R software program into their computers. Readers who have no experience regarding downloading, installing, and using R are recommended to check the program manual on the CRAN website (https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf) prepared by Venables, Smith, and the R Core Team (2019). The *rattle* package contains a set of functions to implement data mining with a GUI. The latest installation instructions can be found at http://rattle.togaware.com. In this study, Rattle version 5.2.0 was used for data mining analysis. To download and install the *rattle* (Williams, 2011) and its required extension *RGtk2* (Lawrence & Lang, 2010), the following codes must be executed in the R console (note that this step requires Internet connection):

```
#Installing the packages
install.packages("rattle")
install.packages("RGtk2")
```

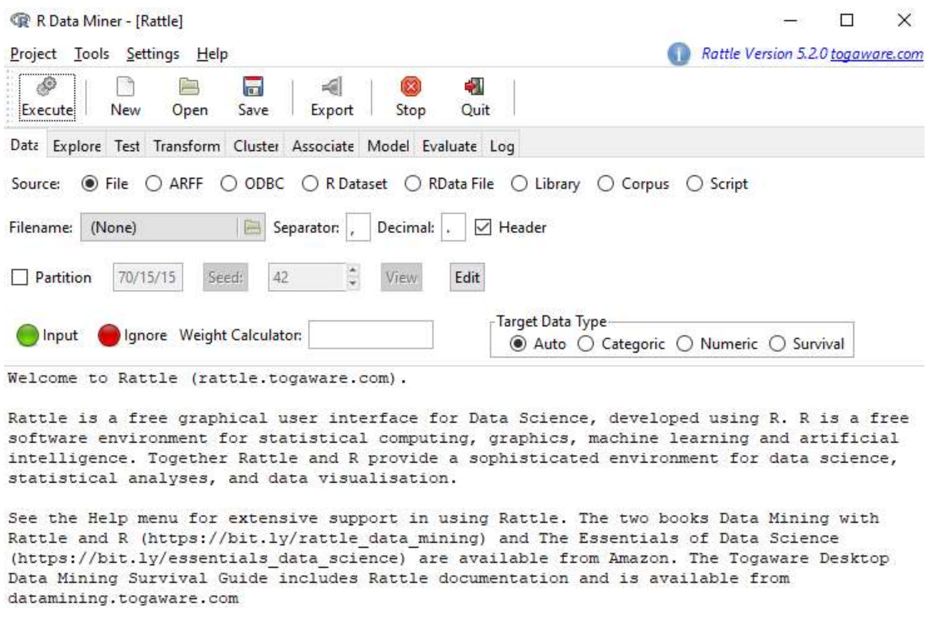
Once the packages have been installed successfully, both packages must be activated using the **library** command in R:

```
#Activating the packages
library("rattle")
library("RGtk2")
```

The next step is to the **rattle** command, which will open the *rattle* GUI as demonstrated in Figure 1.

```
#Opening the rattle GUI
rattle()
```

The *rattle* GUI can read several data formats, such as text files with .txt, .dat, or .csv extensions, RData files, and Open Database Connectivity (ODBC) files. This study uses “pisa_turkey.csv”, which consists of the variables listed in Table 1. To open the pisa_turkey.csv in the *rattle*, the first step is to click “Filename” under the “Data” tab and look for the data file in the computer. Once the data file is found, the “Open” and “Execute” buttons should be clicked, respectively. This process will open the pisa_turkey.csv file in the *rattle* and load the dataset into the program (see Figure 2). For other types of data formats, the same procedure can be followed by selecting a specific file format available under “Source”. Once a dataset is properly read and loaded into the *rattle*, a summary screen of the dataset becomes available (see Figure 3). The summary menu shows all the variables in the dataset, types of variables (numeric or categorical), and the role of the variables (e.g., input, target, and identity). Furthermore, the “Comment” column in the summary screen can help users identify potential issues in the variables (e.g., extreme missingness). Using the summary menu, users can change the default preferences regarding the variables. For example, the outcome variable must be labeled as “Target” so that this variable can be used as the outcome variable in the modeling stage. If the user wants to exclude some variables from the dataset, these variables should be labeled as “Ignore”. Note that changes made on the summary screen will be saved only after the user clicks the “Execute” button. Otherwise, changes made on the variables will be lost.


R Data Miner - [Rattle]
Project Tools Settings Help
Rattle Version 5.2.0 togaware.com
Execute New Open Save Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
Input
Ignore
Weight Calculator:
View
Edit

Source: [✓ File] ARFF ODBC R Dataset RData File Library Corpus Script

Filename: (None) Separator: , Decimal: . [✓ Header]

[ ] Partition 70/15/15 Seed: 42

Target Data Type: [✓ Auto] Categoric Numeric Survival

```
Welcome to Rattle (rattle.togaware.com).

Rattle is a free graphical user interface for Data Science, developed using R. R is a free
software environment for statistical computing, graphics, machine learning and artificial
intelligence. Together Rattle and R provide a sophisticated environment for data science,
statistical analyses, and data visualisation.

See the Help menu for extensive support in using Rattle. The two books Data Mining with
Rattle and R (https://bit.ly/rattle_data_mining) and The Essentials of Data Science
(https://bit.ly/essentials_data_science) are available from Amazon. The Togaware Desktop
Data Mining Survival Guide includes Rattle documentation and is available from
datamining.togaware.com
```

Figure 1. The graphical user interface (GUI) of the *rattle*

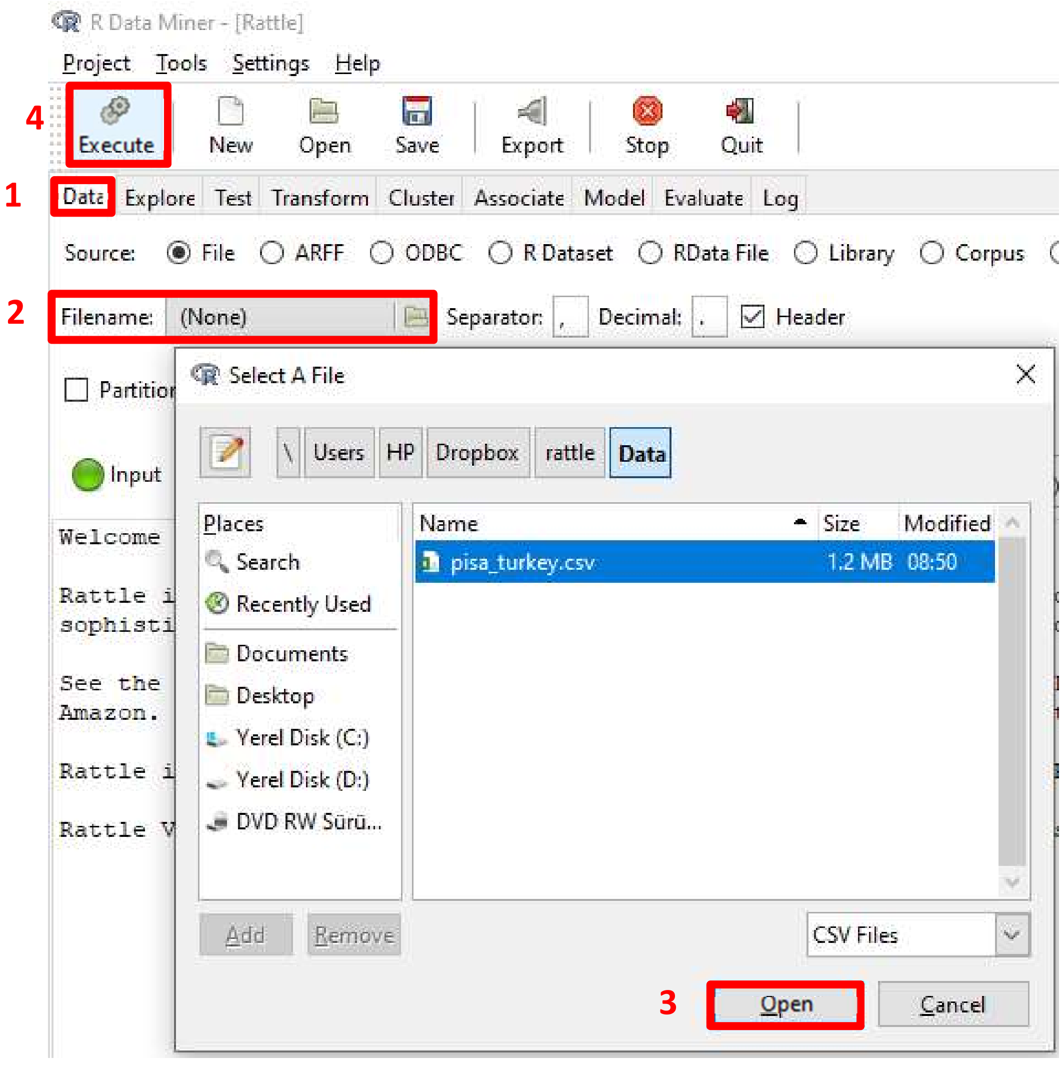
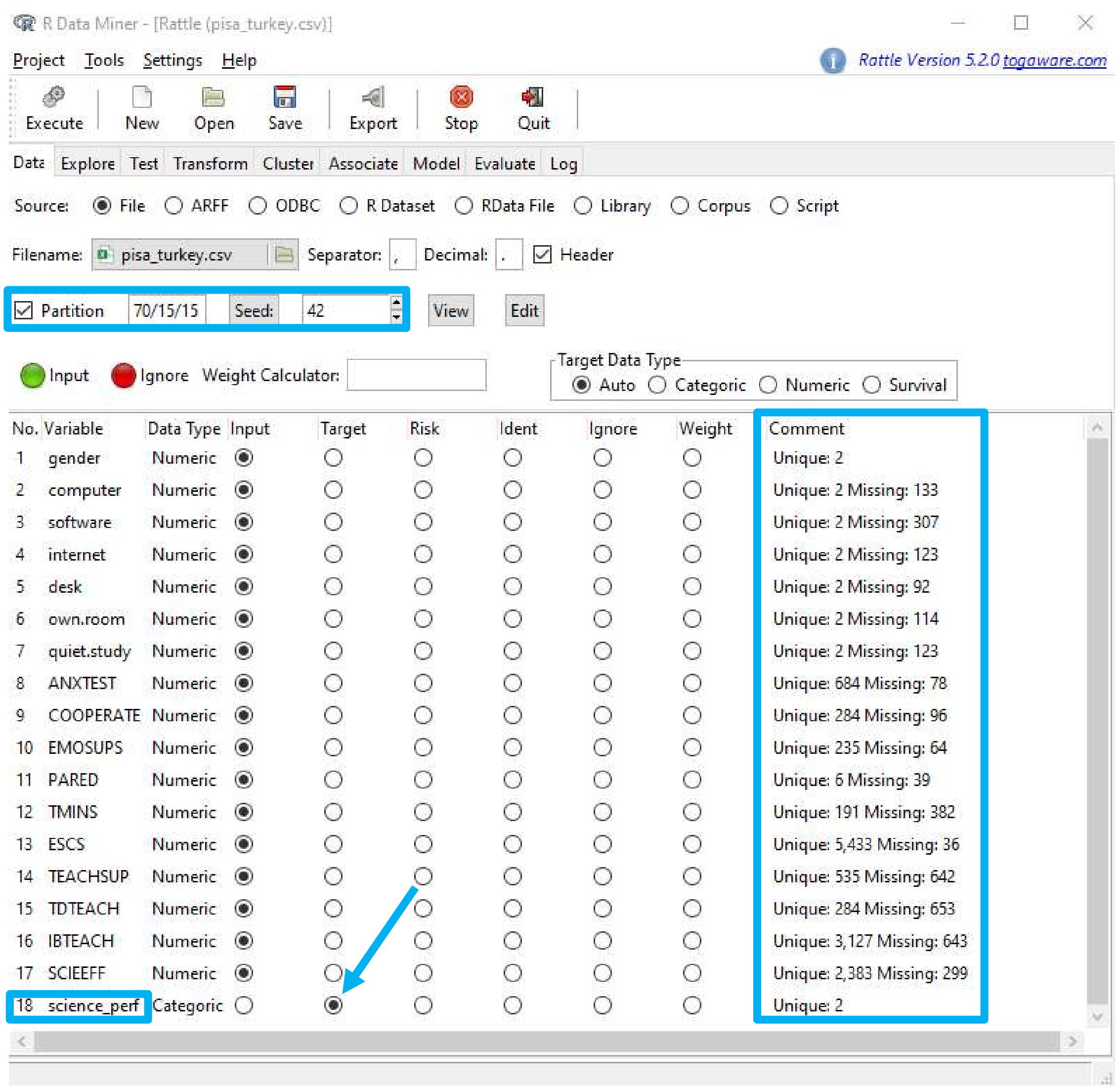

R Data Miner - [Rattle]
Project Tools Settings Help
4
Execute New Open Save Export Stop Quit
1
Data Explore Test Transform Cluster Associate Model Evaluate Log
Source: File ARFF ODBC R Dataset RData File Library Corpus
2
Filename: (None) Separator: , Decimal: . Header
Partition
Input
Welcome
Rattle i
sophisti
See the
Amazon.
Rattle i
Rattle V
Select A File
\ Users HP Dropbox rattle Data
Places
Search
Recently Used
Documents
Desktop
Yerel Disk (C:)
Yerel Disk (D:)
DVD RW Sürü...
Name Size Modified
pisa_turkey.csv 1.2 MB 08:50
Add Remove
CSV Files
3
Open Cancel

Figure 2. Loading the data


R Data Miner - [Rattle (pisa_turkey.csv)]
Project Tools Settings Help
Rattle Version 5.2.0 togaware.com
Execute New Open Save Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log

Source: [✓ File] ARFF ODBC R Dataset RData File Library Corpus Script

Filename: pisa_turkey.csv Separator: , Decimal: . [✓ Header]

[✓ Partition] 70/15/15 Seed: 42 View Edit

Input Ignore Weight Calculator:

Target Data Type: [✓ Auto] Categoric Numeric Survival

No.	Variable	Data Type	Input	Target	Risk	Ident	Ignore	Weight	Comment
1	gender	Numeric	●	○	○	○	○	○	Unique: 2
2	computer	Numeric	●	○	○	○	○	○	Unique: 2 Missing: 133
3	software	Numeric	●	○	○	○	○	○	Unique: 2 Missing: 307
4	internet	Numeric	●	○	○	○	○	○	Unique: 2 Missing: 123
5	desk	Numeric	●	○	○	○	○	○	Unique: 2 Missing: 92
6	own.room	Numeric	●	○	○	○	○	○	Unique: 2 Missing: 114
7	quiet.study	Numeric	●	○	○	○	○	○	Unique: 2 Missing: 123
8	ANXTEST	Numeric	●	○	○	○	○	○	Unique: 684 Missing: 78
9	COOPERATE	Numeric	●	○	○	○	○	○	Unique: 284 Missing: 96
10	EMOSUPS	Numeric	●	○	○	○	○	○	Unique: 235 Missing: 64
11	PARED	Numeric	●	○	○	○	○	○	Unique: 6 Missing: 39
12	TMINS	Numeric	●	○	○	○	○	○	Unique: 191 Missing: 382
13	ESCS	Numeric	●	○	○	○	○	○	Unique: 5,433 Missing: 36
14	TEACHSUP	Numeric	●	○	○	○	○	○	Unique: 535 Missing: 642
15	TDTEACH	Numeric	●	○	○	○	○	○	Unique: 284 Missing: 653
16	IBTEACH	Numeric	●	○	○	○	○	○	Unique: 3,127 Missing: 643
17	SCIEEFF	Numeric	●	○	○	○	○	○	Unique: 2,383 Missing: 299
18	science_perf	Categoric	○	●	○	○	○	○	Unique: 2

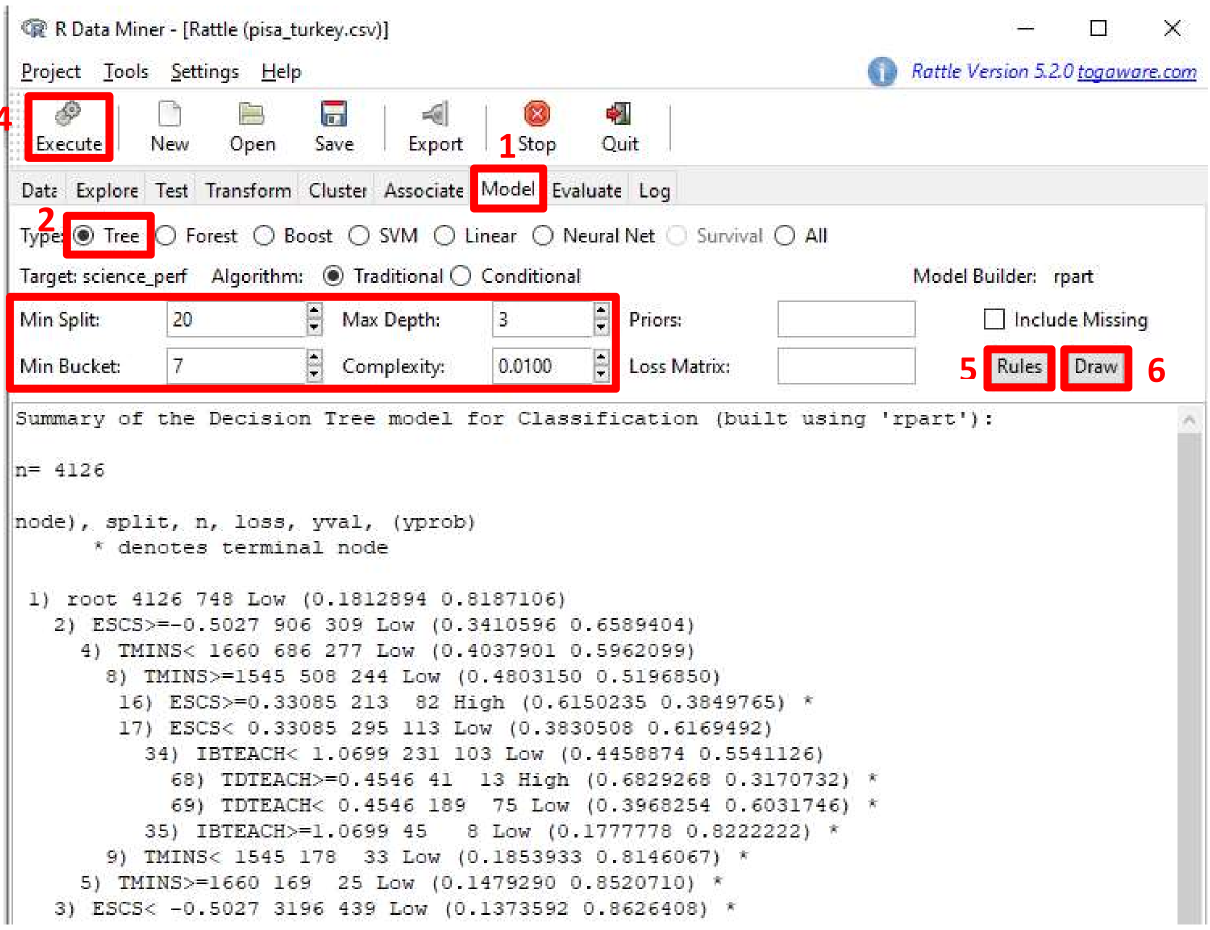
Figure 3. The view of the "pisa_turkey.csv" dataset

3. RESULTS/FINDINGS

This section demonstrates how to implement the CART, RF, and SVM algorithms for the prediction of students' proficiency status in the scientific literacy test. For each algorithm, the *rattle* will require users to download and install the required packages for the first-time implementation. Therefore, users should accept and install the suggested packages if the *rattle* shows any warning messages about downloading and installing such packages. By default, the *rattle* should be able to recognize "science_perf" as the target variable. However, if this is not the case, it must be specified as "Target" under the Data option before running the subsequent analyses (see Figure 3). Once the "Partition" option is checked, the *rattle* splits the dataset into three parts: training dataset (70% of the dataset), test dataset (15% of the dataset), and the validation dataset (the remaining 15% of the dataset). These partitions are created using random sampling based on the seed value (default = 42) under the Data tab. Using the same seed ensures that the user can get the same randomly drawn training, test, and validation datasets every time the *rattle* is used for the same dataset. The training dataset is used for model building and the other two datasets are used for evaluating the accuracy of predictions made from the model. Alternatively, two datasets (e.g., training with 70% and validation with 30%) can be created by typing 70/30 inside the "Partition" box.

3.1. Classification and regression trees (CART)

The first two steps to build a decision tree using the CART approach are to switch to the "Model" tab in the *rattle* and to select the "Tree" option (see Figure 4). Then, the third step is to set the model parameters. "Min Split" is the minimum number of observations that must exist in a node (default = 20); "Min Bucket" is the minimum number of observations in any terminal node (default is Min Split/3); "Max Depth" is the maximum depth of any node of the final tree (default = 3); and "Complexity" is the complexity parameter to prune the subtrees that do not improve the overall model fit (default = 0.01). If this parameter is set to zero, then the CART algorithm keeps all the estimated nodes and typically creates a highly complex model that might be hard to interpret. However, a large value for the complexity parameter might also be detrimental to the model because it would remove many useful nodes from the model and leave a simple model with a very low predictive accuracy. Therefore, users are recommended to build several models by tuning the model parameters based on resulting model evaluation indices (e.g., accuracy, sensitivity, and recall). The fourth step is to click on the "Execute" button – which runs the CART algorithm based on the requested settings. The CART algorithm uses all the variables selected as "input" under the Data tab to predict the target variable (science_perf). Once the estimation is complete, the results can be printed on the screen by clicking on the "Rules" button. Furthermore, visualizations can be drawn for the final decision tree model by clicking the "Draw" button. Figure 4 illustrates the steps to be followed to implement the CART approach and the output returned from the *rattle*.


R Data Miner - [Rattle (pisa_turkey.csv)]
Project Tools Settings Help
Rattle Version 5.2.0 togaware.com
4 Execute New Open Save Export 1 Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
2
3
5 Rules Draw 6

Type: [✓ Tree] Forest Boost SVM Linear Neural Net Survival All

Target: science_perf Algorithm: [✓ Traditional] Conditional Model Builder: rpart

Min Split: 20 Max Depth: 3 Priors: Include Missing

Min Bucket: 7 Complexity: 0.0100 Loss Matrix:

```
Summary of the Decision Tree model for Classification (built using 'rpart'):

n= 4126

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 4126 748 Low (0.1812894 0.8187106)
   2) ESCS>=-0.5027 906 309 Low (0.3410596 0.6589404)
     4) TMINS< 1660 686 277 Low (0.4037901 0.5962099)
       8) TMINS>=1545 508 244 Low (0.4803150 0.5196850)
        16) ESCS>=0.33085 213  82 High (0.6150235 0.3849765) *
        17) ESCS< 0.33085 295 113 Low (0.3830508 0.6169492)
          34) IBTEACH< 1.0699 231 103 Low (0.4458874 0.5541126)
            68) TDTEACH>=0.4546 41  13 High (0.6829268 0.3170732) *
            69) TDTEACH< 0.4546 189  75 Low (0.3968254 0.6031746) *
          35) IBTEACH>=1.0699 45   8 Low (0.1777778 0.8222222) *
       9) TMINS< 1545 178  33 Low (0.1853933 0.8146067) *
     5) TMINS>=1660 169  25 Low (0.1479290 0.8520710) *
   3) ESCS< -0.5027 3196 439 Low (0.1373592 0.8626408) *
```

Figure 4. Building a predictive model with the CART algorithm

The output returned from the CART model shows the rules that were used to create the nodes in the decision tree. The output shows the decision nodes and the terminal nodes that were specified with *. For example, a decision node was created based on ESCS (index of economic, social and cultural status) at the beginning of the tree. Based on whether students' ESCS index values were equal or larger than -0.5027, two branches were created in the decision tree model. Then, the group of students who meet the ESCS condition is split into two additional branches depending on whether their total learning time (TMINS) is less than 1660 minutes. The remaining nodes can be interpreted in a similar manner. A relatively easier way to see all the nodes in the model is to draw a decision tree plot. The "Draw" option under the Model tab generates a decision tree plot based on the nodes summarized in the output. Figure 5 shows the decision tree plot returned from the *rattle* for the prediction of the proficiency status in scientific literacy (i.e., science_perf).

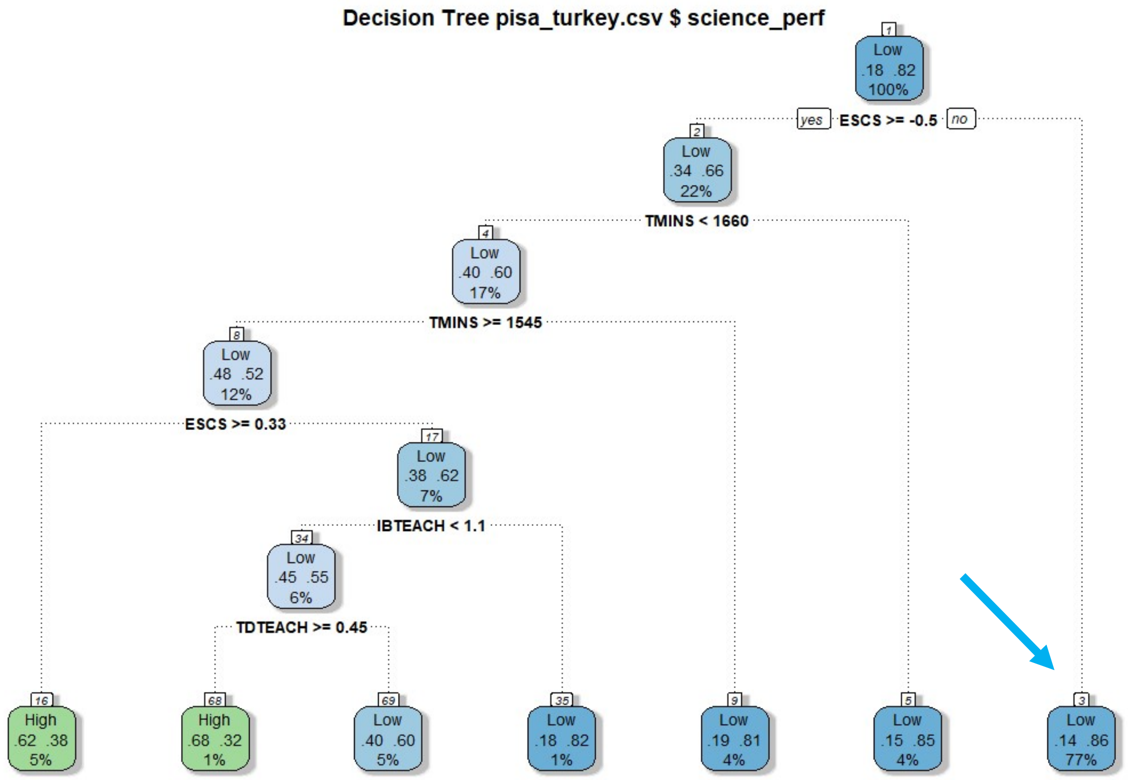
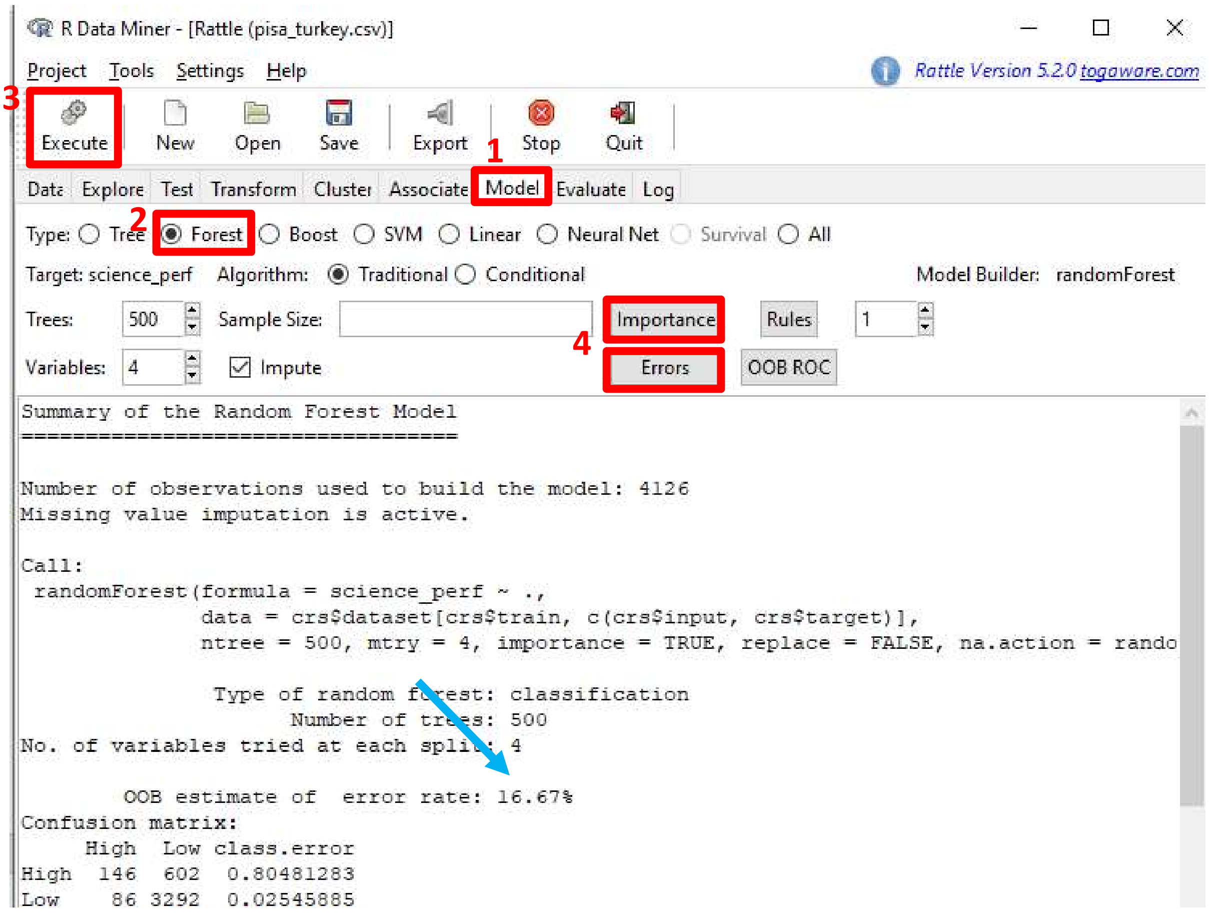


Figure 5. The decision tree plot for the prediction of science_perf

In Figure 5, the categories of science_perf are color-coded where the blue color boxes represent the "Low" category and the green color boxes represent the "High" category. Within each box, the two values in the middle represent the probabilities of the first and second categories. For example, the first terminal node on the right-hand side of the plot shows that students who have ESCS index values smaller than -0.5 have the probabilities of 14% of being in the "Low" category and 86% of being in the "High" category. The number at the bottom of each box represents the percentage of observations in the node. Focusing on the same blue box from the previous example, 77% of the students in the training dataset fall into the node where ESCS is smaller than -0.5. Figure 5 also shows that only four of the input variables (ESCS, IBTEACH, TDTEACH, and TMINS) were used as the predictors. This is because the CART algorithm keeps the predictors that can significantly contribute to the prediction, depending on the selected model parameters (e.g., complexity, min split, and max depth).

3.2. Random forest (RF)

To implement the RF algorithm for the same classification task (i.e., predicting science_perf) in the *rattle*, the "Forest" option must be selected under the "Model" tab. Then, the model parameters need to be determined. "Trees" refer to the numbers of decision (or regression) trees to be built (default = 500); "Variables" is the number of predictors randomly sampled as candidates at each split (default = square root of the number of predictors for classification and the number of predictors / 3 for regression); and "Sample Size" is the sizes of sample to draw (default = 0.632 * the number of observations in the training dataset). Once the model parameters are determined, the next step is to click on the "Execute" button to perform the analysis. Like the CART algorithm, the RF algorithm also uses all of the input variables to predict the target variable (science_perf). Once the estimation is complete, the results can be printed on the screen by clicking the "Rules" and "Importance" buttons. Figure 6 shows the steps to be followed to implement the RF algorithm and to view the output in the *rattle*.


R Data Miner - [Rattle (pisa_turkey.csv)]
Project Tools Settings Help
Rattle Version 5.2.0 togaware.com
3
Execute New Open Save Export Stop Quit
1
Data Explore Test Transform Cluster Associate Model Evaluate Log
2
4
Importance
Rules
Errors
OOB ROC

Type: ○ Tree [✓ Forest] ○ Boost ○ SVM ○ Linear ○ Neural Net ○ Survival ○ All

Target: science_perf Algorithm: [✓ Traditional] ○ Conditional Model Builder: randomForest

Trees: 500 Sample Size:

Variables: 4 [✓ Impute]

```
Summary of the Random Forest Model
==================================

Number of observations used to build the model: 4126
Missing value imputation is active.

Call:
 randomForest(formula = science_perf ~ .,
              data = crs$dataset[crs$train, c(crs$input, crs$target)],
              ntree = 500, mtry = 4, importance = TRUE, replace = FALSE, na.action = rando

               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 16.67%
Confusion matrix:
     High  Low class.error
High  146  602  0.80481283
Low    86 3292  0.02545885
```

Figure 6. Building a predictive model with the RF algorithm

The output returned from the RF algorithm shows the number of observations used for building the model, the formula used to build the predictive model, and the selected model parameters. The output also shows additional information, such as the out-of-bag (OOB) estimate of the error rate and the confusion matrix. OBB is a method of measuring the prediction error of a predictive model estimated with the RF algorithm. In this example, the OOB estimate of error rate is 16.67 %, suggesting that 83.33 % of the predictions made for science_perf is correct within the training dataset. Additionally, the visual output returned from the "Importance" and "Errors" indicates the importance of the predictors in the prediction process (Figure 7) and error rates across all of the decision trees built for the model (Figure 8). Based on the variable importance measures shown in Figure 7, ESCS, IBTEACH, TMINS, and computer appear to be the strongest predictors in the estimated model since they have higher importance values, compared to the other variables. Although there is no particular cut-off value to determine which predictors are more important, the predictive power of these variables appears to be relatively higher than the other variables (see the "High" category of the top-left corner of Figure 7). The findings also suggest that the model error rates did not change after 100 trees. That is, the same model could be estimated with only 100 trees to obtain the final model more efficiently.

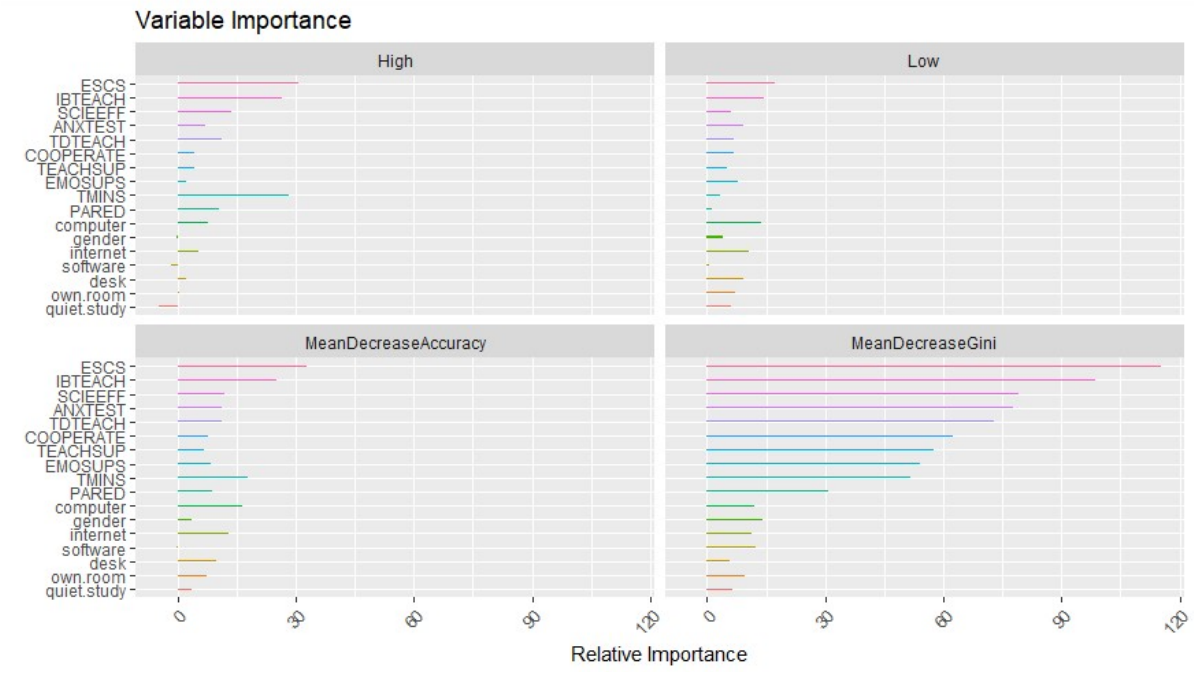


Figure 7. A plot of variable importance for the RF algorithm

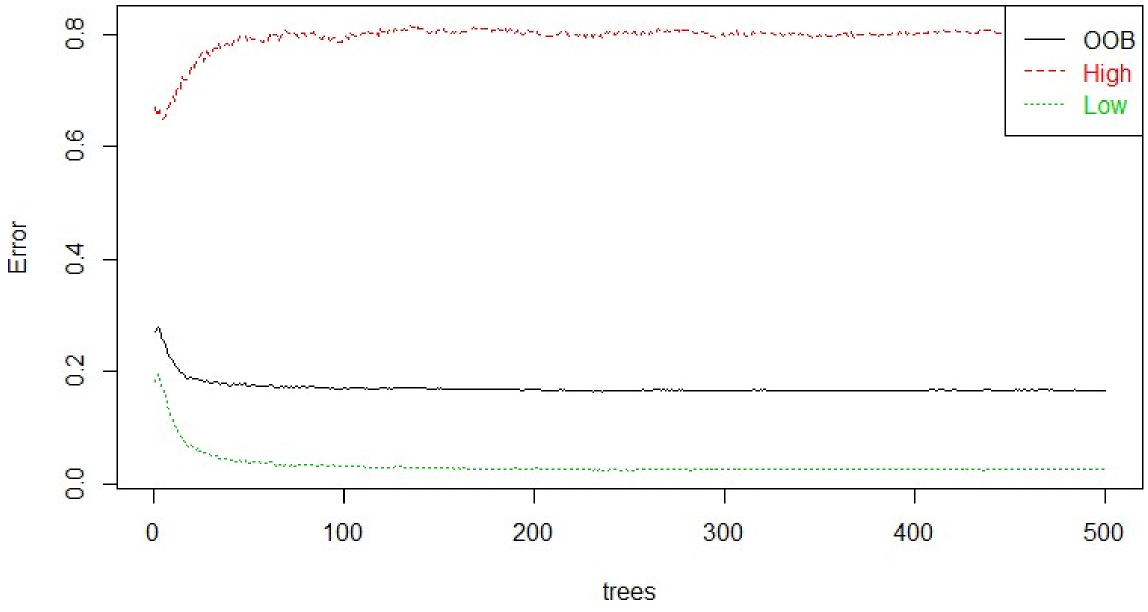
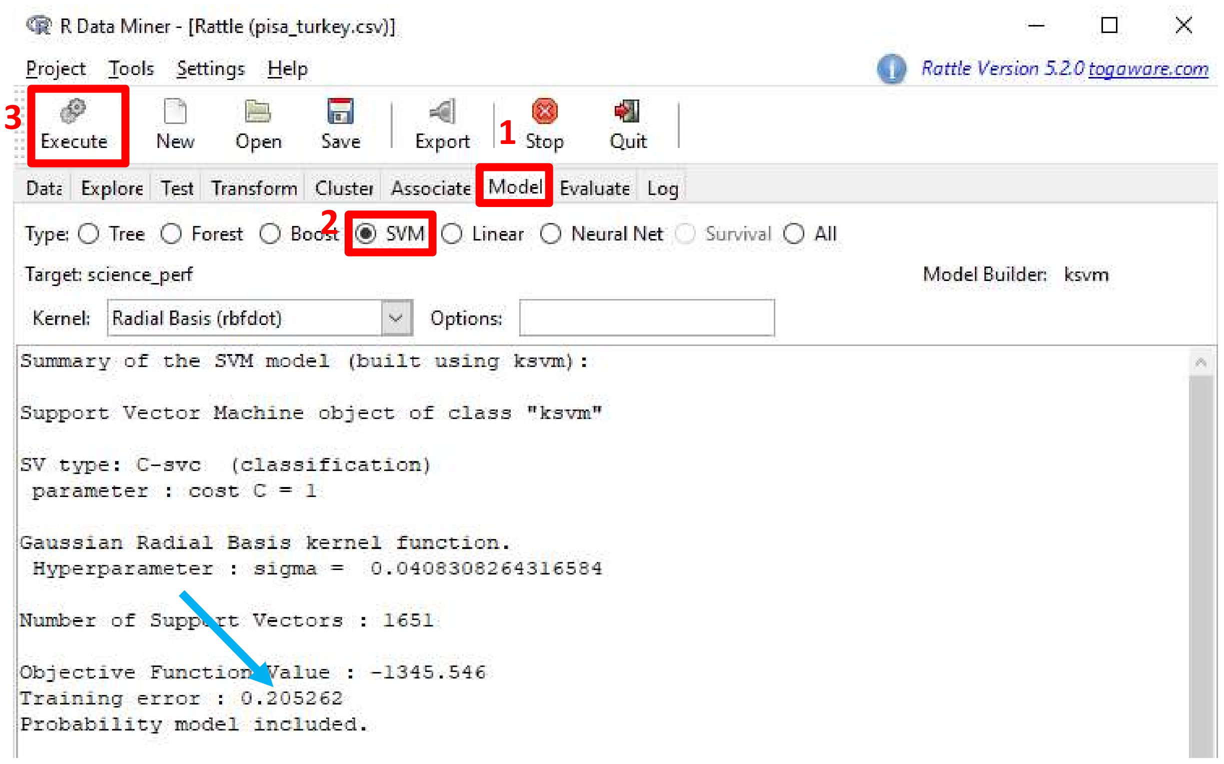


Figure 8. A plot of error rates for the RF algorithm

3.3. Support vector machine (SVM)

To implement the SVM algorithm for predicting science_perf, the "SVM" option must be selected under the "Model" tab. Unlike the CART and RF algorithms, there are not many model parameters to choose for the SVM algorithm. Instead, the most important decision that users must make is the selection of a kernel function. The default kernel function in the *rattle* is "rbfdot", which refers to the Gaussian radial basis function. The "rbfdot" function is a general-purpose kernel suitable for cases where there is no prior knowledge about the data. There are also other popular kernel functions available for the SVM algorithm, such as "polydot" for the polynomial kernel function and "vanilladot" for the linear kernel function. Non-linear kernels often provide a better model-data fit than linear kernels at the expense of high computational complexity and estimation time. Once a kernel is selected, the next step is to click on the "Execute" button to perform the analysis. Like the previous algorithms, the SVM algorithm also utilizes all of the selected input variables to predict the target variable (science_perf). Once the estimation is complete, the results are printed on the screen. Figure 9 shows the steps to be followed to implement the SVM algorithm in the *rattle*.


R Data Miner - [Rattle (pisa_turkey.csv)]
Project Tools Settings Help
Rattle Version 5.2.0 togaware.com
3
Execute
New
Open
Save
Export
1
Stop
Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
Type: Tree Forest Boost
2
[✓ SVM] Linear Neural Net Survival All
Target: science_perf
Model Builder: ksvm
Kernel: Radial Basis (rbfdot)
Options:

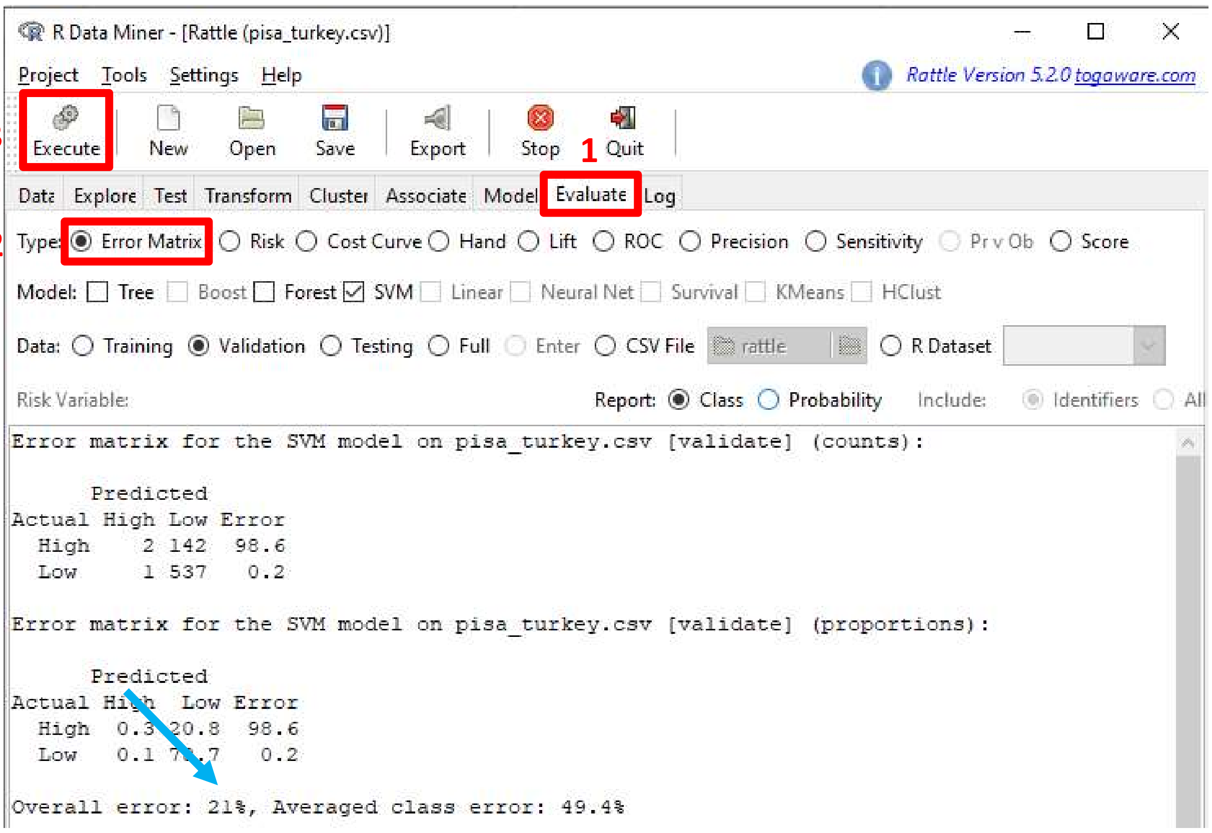
```
Summary of the SVM model (built using ksvm):

Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0408308264316584

Number of Support Vectors : 1651

Objective Function Value : -1345.546
Training error : 0.205262
Probability model included.
```

Figure 9. Building the support vector machine classification model

The output returned from the SVM algorithm shows the default settings used in the estimation process (the cost parameter of C as 1 and the hyperparameter of sigma as 0.0408). In addition, the output shows the number of support vectors created in the model (1651). An important section of the output is "Training error". The results show that the overall prediction error in the training dataset was around 20.53%. That is, roughly 80% of the predictions made for science_perf in the training dataset are accurate. It should be noted that although the output resulted from the SVM algorithm is quite concise in the *rattle* compared to those from the CART and RF algorithms, it is often much more difficult to interpret the content of this output given the complex hyperparameters used in the SVM algorithm.

3.4. Evaluating models

Unlike traditional statistical methods, the data mining methods require researchers to build several models, evaluate outcomes from each model, adjust the models accordingly, and continue to tune the models until an acceptable level of accuracy is reached. As demonstrated in this tutorial, several algorithms can also be used for the same classification or regression task. Therefore, researchers must not only tune their models but also select the most suitable algorithm based on the model evaluation measures. In the *rattle*, model evaluation can be performed using the options under the "Evaluate" tab. For model evaluation, either validation or test datasets should be used because these datasets consist of the observations that the algorithms have not seen when building the prediction model. Figure 10 shows the steps to view the error matrix from the SVM algorithm, although the same steps can be followed to see the same output for other algorithms as well. This tutorial focused on the prediction of a binary outcome variable (science_perf), and thus the error matrix returns a two-by-two matrix of predicted and actual values and proportions of the two categories (i.e., "High" and "Low" proficiency in scientific literacy). The overall prediction error for the SVM-based model is 21% and the average classes (i.e., category) error is 49.4%. The error matrix shows that the prediction accuracy of the "Low" category was precise, whereas the prediction accuracy of the "High" category was quite poor.


R Data Miner - [Rattle (pisa_turkey.csv)]
Project Tools Settings Help
Rattle Version 5.2.0 togaware.com
3
Execute
New
Open
Save
Export
Stop
1
Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
2
Type: Error Matrix Risk Cost Curve Hand Lift ROC Precision Sensitivity Pr v Ob Score
Model: Tree Boost Forest SVM Linear Neural Net Survival KMeans HClust
Data: Training Validation Testing Full Enter CSV File rattle R Dataset
Risk Variable:
Report: Class Probability
Include: Identifiers All

Type: [✓ Error Matrix] Model: [✓ SVM] Data: [✓ Validation] Report: [✓ Class]

```
Error matrix for the SVM model on pisa_turkey.csv [validate] (counts):

      Predicted
Actual High Low Error
  High    2 142  98.6
  Low     1 537   0.2

Error matrix for the SVM model on pisa_turkey.csv [validate] (proportions):

      Predicted
Actual High  Low Error
  High  0.3 20.8  98.6
  Low   0.1 78.7   0.2

Overall error: 21%, Averaged class error: 49.4%
```

Figure 10. The view of the "Evaluate" tab in *rattle*

The "Evaluate" tab offers many useful measures for model evaluation. For example, the sensitivity, specificity, precision, and recall plots can be created using the "Sensitivity" and "Precision" options under the "Evaluate" tab. Users must select one of these evaluation options and click "Execute" to draw the plots. Table 2 shows the calculation of the evaluation measures available in the *rattle*.

Table 2. Evaluation measures for the classification of "Low" and "High" groups in science_perf

Predicted Classification	Actual Classification	
	Low proficiency in science	**High** proficiency in science
Low proficiency in science	True Positive (TP)	False Positive (FP)
High proficiency in science	False Negative (FN)	True Negative (TN)

Note: Sensitivity = TP/TP+FN; Specificity = TN/TN+FP; Precision = TP/(TP+FP); Recall = TP/(TP+FN)

To compare the results from different data mining algorithms, the models must be estimated with each algorithm first so that the evaluation measures under the "Evaluation" tab can draw the plots by including the results from all algorithms. Figures 11 and 12 show the plots of sensitivity/specificity and precision/recall across the three data mining algorithms (i.e., CART, RF, and SVM). Figure 11 shows that there is a significant trade-off between sensitivity and specificity for all the algorithms. As the specificity level (i.e., detecting "High") increases, the sensitivity level (i.e., detecting "Low") decreases. Among the three algorithms, the performance of the RF algorithm appears to be the best in terms of balancing sensitivity and specificity. Figure 12 shows the precision and recall levels across the three algorithms. The results suggest that all the algorithms indicate high precision and recall values in predicting the "High" and "Low" values of science_perf. Given the similar precision and recall values across the three algorithms, sensitivity and specificity can be more decisive evaluation measures for the example presented in this study.

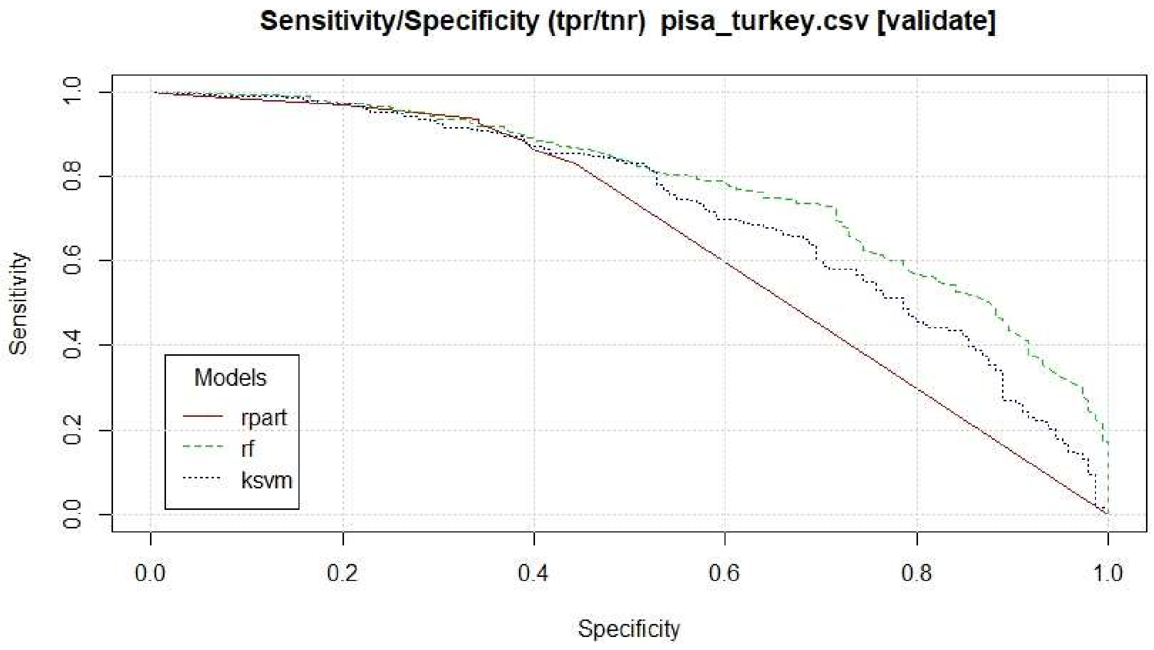


Figure 11. The sensitivity and specificity plots of the three data mining algorithms (**Note**: rpart refers to CART, rf refers to RF, and ksvm refers to SVM).

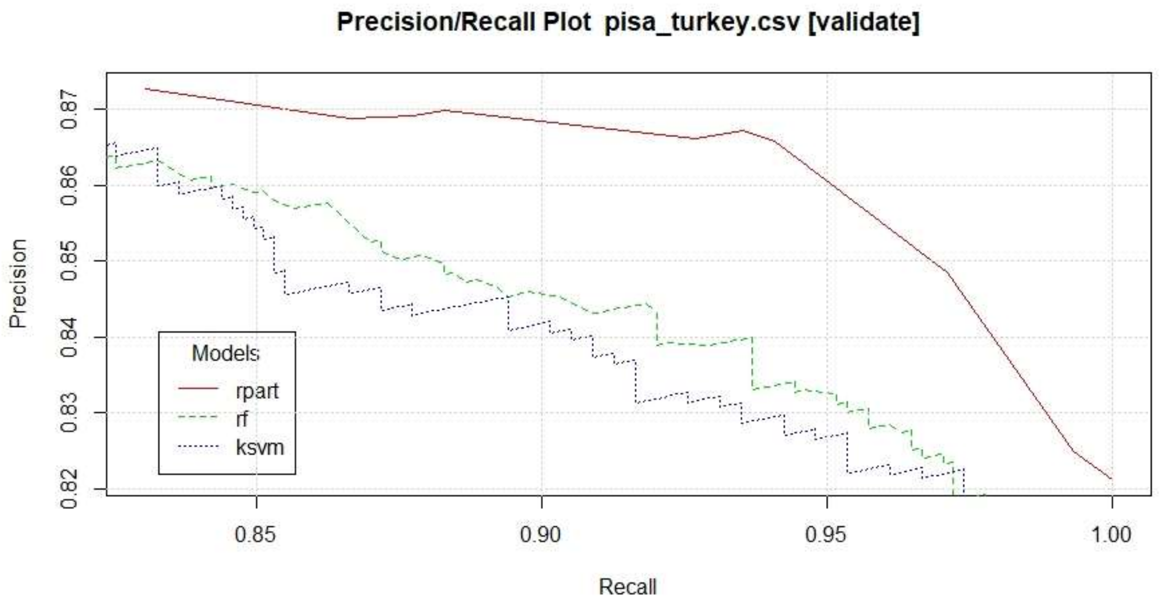


Figure 12. The precision and recall plots of the three data mining algorithms (**Note**: rpart refers to CART, rf refers to RF, and ksvm refers to SVM).

4. DISCUSSION and CONCLUSION

The purpose of this study was to demonstrate the implementation of the three data mining algorithms (i.e., CART, RF, and SVM) using the *rattle* package (Williams, 2011) in R (R Core Team, 2019). The selected algorithms are widely used methods in EDM research for both classification and regression tasks. The example used in this study demonstrated how to build classification models using the CART, RF, and SVM algorithms for predicting students' proficiency levels (low or high) in the scientific literacy test of PISA 2015. In addition, the model evaluation stages were also described.

Based on the results of this study, the RF algorithm appeared to be the best performing algorithm for predicting students' proficiency levels in the scientific literacy test of PISA 2015. This is not a surprising finding because the RF algorithm often provides accurate prediction results in datasets that contain both numerical and categorical predictors (i.e., features). These findings tie well with a previous study wherein Fernández-Delgado et al. (2014) compared the performances of 179 classification algorithms using 121 real datasets. The researchers found that the RF algorithm was the best algorithm for most real world classification problems, followed by the SVM algorithm. A similar pattern of results was obtained in the current study.

The results from the three algorithms were somewhat different in this study mainly because each algorithm handles different types of variables and their relationships in the pisa_turkey dataset. For example, the CART algorithm yielded sensitivity and specificity values similar to those from the other two algorithms, but it used fewer predictors in the estimation. Depending on what complexity parameter has been selected, the decision tree model can either retain or eliminate the subtrees created based on relatively less important predictors in the dataset. Furthermore, when some predictors are highly correlated, the CART algorithm may choose only one of those predictors and ignore the others. Therefore, researchers are recommended to choose an algorithm and tune its parameters after careful consideration and review of their data.

As a free software program, the *rattle* uses many powerful packages available within the R computing environment for conducting data mining analysis. Unlike the R software program that requires users to type and execute their codes, the *rattle* provides a user-friendly GUI that enables users to import their data files easily, select an algorithm from a variety of options, and evaluate the results using model evaluation measures. The output returned from the *rattle* involves both statistical and visual outcomes to facilitate users' evaluation and fine-tuning of their models. Although it is not demonstrated in this study, the *rattle* is also capable of providing users with the opportunity to explore their datasets descriptively and to transform variables (e.g., rescaling, recoding, and normalizing) before performing further analysis. In addition, the *rattle* is capable of performing unsupervised data mining, including clustering with *k*-means and hierarchical clustering methods and association rule analysis. For advanced R users who might prefer to keep the R codes for their analysis, the *rattle* provides a script that presents the underlying R codes for all analyses conducted in the program under the "Log" tab. For a comprehensive review of the *rattle*, readers are recommended to check out *Data Mining with Rattle and R* by Williams (2011) who is also the author of the *rattle*.

Acknowledgements

This study used some materials from the workshop of "Exploring, Visualizing, and Modeling Big Data with R" held at the 2019 annual meeting of the National Council on Measurement in Education, Toronto, ON, Canada.

ORCID

Okan Bulut https://orcid.org/0000-0001-5853-1267
Hatice Cigdem Yavuz https://orcid.org/0000-0003-2585-3686

5. REFERENCES

Agarwal, S., Pandey, G. N., & Tiwari, M. D. (2012). Data mining in education: Data classification and decision tree approach. *International Journal of e-Education, e-Business, e-Management and e-Learning, 2*(2), 140.

Aldowah, H., Al-Samarraie, H., & Fauzy, W. M. (2019). Educational Data Mining and Learning Analytics for 21st century higher education: A Review and Synthesis. *Telematics and Informatics, 37*, 13-49.

Aulck, L., Velagapudi, N., Blumenstock, J., & West, J. (2016). Predicting student dropout in higher education. *arXiv preprint arXiv:1606.06364*.

Baker, R. S., Martin, T., & Rossi, L. M. (2017). Educational data mining and learning analytics. In A. A. Rupp & J. P. Leighton (Eds.), *The handbook of cognition and assessment: Frameworks, methodologies, and applications* (pp. 379-396). Oxford, UK: John Wiley & Sons, Inc.

Berland, M., Baker, R. S., & Blikstein, P. (2014). Educational data mining and learning analytics: Applications to constructionist research. *Technology, Knowledge and Learning, 19*(1-2), 205-220.

Breiman, L. (2001). Random forest. *Machine Learning, 45*(1), 5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273-297.

Ducange, P., Pecori, R., Sarti, L., & Vecchio, M. (2016, October). Educational big data mining: how to enhance virtual learning environments. In *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16* (pp. 681-690). Springer, Cham.

Dutt, A., Ismail, M. A., & Herawan, T. (2017). A systematic review on educational data mining. *IEEE Access*, 5, 15991-16005.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, *15*(1), 3133-3181.

Guruler, H., Istanbullu, A., & Karahasan, M. (2010). A new student performance analysing system using knowledge discovery in higher educational databases. *Computers & Education, 55*(1), 247-254.

Hussain, M., Zhu, W., Zhang, W., Abidi, S. M. R., & Ali, S. (2019). Using machine learning to predict student difficulties from learning session data. *Artificial Intelligence Review*, *52*(1), 381-407.

Ivancevic, V., Celikovic, M., & Lukovic, I. (2011). Analyzing student spatial deployment in a computer laboratory. In *Proceedings of the 4th international conference on educational data mining* (pp. 265–270).

Koon, S., & Petscher, Y. (2015). *Comparing methodologies for developing an early warning system: Classification and regression tree model versus logistic regression. REL 2015-077*. Regional Educational Laboratory Southeast.

Koon, S., & Petscher, Y. (2016). *Can scores on an interim high school reading assessment accurately predict low performance on college readiness exams? REL 2016-124*. Regional Educational Laboratory Southeast.

Lawrence, M., & Lang, D. T. (2010). RGtk2: A ghraphical user interface toolkit for R. *Journal of Statistical Software, 37*(8), 1-52.

Mccuaig, J., & Baldwin, J. (2012). Identifying successful learners from interaction behaviour. In *Proceedings of the 5th international conference on educational data mining* (pp. 160–163).

Mostafa, T., Echazarra, A., & Guillou, H. (2018). *The science of teaching science: An exploration of science teaching practices in PISA 2015*. OECD Education Working Papers, No. 188. Paris, France: OECD Publishing.

OECD (2017). PISA 2015 *Assessment and Analytical Framework: Science, Reading, Mathematic, Financial Literacy and Collaborative Problem Solving*. PISA, OECD Publishing, Paris, https://doi.org/10.1787/9789264281820-en

OECD (2018). PISA 2015 results in focus. Retrieved from https://www.oecd.org/pisa/pisa-2015-results-in-focus.pdf

Pardos, Z. A., Wang, Q. Y., & Trivedi, S. (2012). The real world significance of performance prediction. In *Proceedings of the 5th international conference on educational data mining* (pp. 192–195).

Peña-Ayala, A. (2014). Educational data mining: A survey and a data mining-based analysis of recent works. *Expert System with Applications, 41*(4), 1432-1462. http://dx.doi.org/10.1016/j.eswa.2013.08.042

R Core Team (2019). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing.

Sinharay, S. (2016). An NCME instructional module on data mining methods for classification and regression. *Educational Measurement: Issues and Practice, 35*(3), 38–54. http://dx.doi.org/10.1111/emip.12088

Slater, S., Joksimović, S., Kovanovic, V., Baker, R. S., & Gasevic, D. (2017). Tools for Educational Data Mining: A Review. *Journal of Educational and Behavioral Statistics, 42*(1), 85–106. https://doi.org/10.3102/1076998616666808

Spikol, D., Ruffaldi, E., Dabisias, G., & Cukurova, M. (2018). Supervised machine learning in multimodal learning analytics for estimating success in project-based learning. *Journal of Computer Assisted Learning*, *34*(4), 366-377.

Strobl, C. (2013). Data mining. In T. Little (Ed.), *The Oxford handbook of quantitative methods in psychology* (Vol. 2, pp. 678–700). New York, NY: Oxford University Press.

Venables, W. N., Smith, D. N., & the R Core Team (2019). An introduction to R. Retrieved from https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

Williams, G. J. (2011). *Data mining with Rattle and R: The art of excavating data for knowledge discovery*. New York: Springer-Verlag.

International Journal of Assessment Tools in Education
2019, Vol. 6, No. 5-Special Issue, 37–43
https://dx.doi.org/10.21449/ijate.591669

Published at http://www.ijate.net http://dergipark.gov.tr *Research Article*

Determination of Sample Size and Observation Units

Tülin Acar [1,*]

[1] Parantez Education Research Consultancy Publishing, Ankara, Turkey

ARTICLE HISTORY

Received: 13 July 2019
Revised: 06 October 2019
Accepted: 05 November 2019

KEYWORDS

Simple Random Sampling,
Systematic Sampling,
Stratified Sampling,
Probability Sampling,

Abstract: The purpose of this study was to write programs to define sampling sizes and observation units by probability sampling methods and to provide an idea for software developers. The algorithms of the programs were written in Python 3. The programs may be run by double-clicking on the Windows operating system or by the command prompt of the DOS operating system. Each exe file has a memory space of 5 megabits on average. In this respect, the application files are very useful in terms of sharing by email and mobility by USB memory sticks.

1. INTRODUCTION

The purpose of science is to make explanations about "an object of research or its properties". The "method" is one of the principal issues in both creating scientific explanations that do not contract reason and logic and verifying or falsifying the existing explanations. Method is a kind of program for the things to do (Thomas, 2009). In this respect, acquisition of object(s), the number of objects observed, the representativeness of the objects selected, and whether objects are selected impartially are as critical as a scientific researcher's purpose, theory and hypotheses.

Since science has a language and a method, not all analyses, studies or observations are considered scientific, because the quality of being scientific implies a set of activities that do not rest on mere judgments but on proofs and methods. Both accuracy and ethics of the conclusions made in a study point to the soundness of the method of research. Therefore, a sound and strong method of scientific research rests on the knowledge of the researchers and their sensitivity of following the procedure and principles.

Testing/observing how many units/objects does it take to conduct a valid and sound research? Contrary to popular belief, the answer to this question is not related to a study being qualitative or quantitative. At this point, the object and subject matter of science or a scientific study takes priority. Science is what can be known by information. In other words, science is an effort to

CONTACT: Tülin ACAR ✉ totbicer@gmail.com Parantez Education Research Consultancy Publishing, Ankara, Turkey

explain/study a part of reality. Therefore, this effort to explain is possible by induction or deduction. Based on the principles of reasoning (deduction or induction) that must be employed in the effort to explain a part of reality, categorization of scientific studies as qualitative or quantitative research and association of certain sampling methods with the categorized types of research contradicts the nature of science. Just as the object of information in physics is not the same as that of anthropology, the nature of an object of information of a study cannot be associated with a researcher being qualitative or quantitative. The quality of being qualitative or quantitative is about the measurement and its result. Therefore, the answers to the question of "testing/observing how many units/objects does it take to conduct a valid and sound research?" are not about qualitative or quantitative research type.

One of the most essential elements to be distinguished in a scientific study that attempts to explain reality is to decide whether the sampling or a sample of the properties of the object of study should be analyzed. Sampling is a selection process. On the other hand, a sample is a specimen, a part of the whole. The question of "testing/observing how many units/objects does it take to conduct a valid and sound research?" will be inadequate when this difference is ignored. An initial principle will be necessary for a selection from defined units/objects/elements. The said initial principle may have several different sampling methods. In this case, sampling methods should inevitably have strengths and weaknesses compared to each other.

The method of sampling should not be considered merely a process of defining a number of units/objects/elements. The method of sampling is one of the premises of making a logical explanation about the subject matter of the object of research. For instance, it is not reasonable to take soil samples from every square meter of land using simple random sampling or purposeful sampling for the soil analysis of a field of 5 dunams since it is not compulsory to ensure that the sample is representative of the whole in such a study. On the other hand, for a research on reading comprehension of students in Turkey, it is essential for the students to be representative of the whole. Therefore, it is necessary to grasp the distinctive properties of all sampling methods. In the literature, sampling methods are principally classified in two categories, namely, probability sampling and non-probability sampling (Rubin & Babbie, 2010). Those sampling methods are summarized in Table 1.

In the probability sampling method, units/elements/items are selected from a defined universe based on a statistical probability. However, in the non-probability sampling method, the number of units/elements/items is not determined and units/elements/items are not selected based on a probability and sampling size. Defining a sampling size from the universe using non-probability sampling is accompanied by the selection of the observation units to be included in sampling. This selection process may be tedious for a researcher for studies with a broad universe in particular. Ways to save time and effort bring computer technologies to the mind. Computer technologies makes positive contributions to the research process particularly for the probability sampling methods. Software that is used to define the sample size and observation units is written by different programming languages including PHP, Java, C, C++, R, and Python.

Table 1. Sampling methods and descriptive properties

		Descriptive properties
A Probability Sampling	A.1. Simple Random Sampling	The probability of all units/elements/items being selected is equal and independent of one another.
	A.2. Systematic Sampling	The initial unit/element/item is selected randomly. Nevertheless, selection of other units/elements/items depends on the selection coefficient.
	A.3. Stratified Sampling	The universe is made up of sub-strata (i.e. universes) that are heterogeneous among each other but each homogeneous in itself. Units/elements/items are selected randomly, taking into consideration the strata's rate of representation of the universe.
B. Non-probability Sampling	B.1. Convenience Sampling	Units/elements/items that are easier to reach based with the prevailing conditions are studied.
	B.2.Purposive Sampling	Units/elements/items are studied based on defined and restricted properties.
	B.3. Quota Sampling	The universe is made up of sub-strata (i.e. universes) that are heterogeneous among each other but each homogeneous in itself. Units/elements/items are selected based on a quota.
	B.4. Panel Sampling	Units/elements/items that make the whole are homogeneous in their properties, regardless of the size of the universe. Therefore, the sample is taken from the whole.

1.1. Aim of the Study

The purpose of this study was to write programs to define sampling sizes and observation units by probability sampling methods. Also it was to provide an idea for software developers.

2. METHOD

The algorithms of the programs were written in Python 3. Python is an object-oriented, interpretive, modular, interactive, high-level programming language that is fast and easy to learn (Sahoo& Sahoo, 2016). Codes of the programs written were shared as open source, and user's guides were prepared for the application files. The py code and MIT licenses of the programs can be accessed via Github.

2.1 Determination of the Sampling Size and Observation Units by Simple Random Sampling

Sampling size for simple random sampling is calculated at two stages using the equation below (Royse, Thyer&Padgett, 2010).

$$Sample\ size(n) = \frac{n_0}{1+\frac{n_0}{N}} \qquad \text{and} \qquad n_0 = \frac{Z^2 * [P*(1-P)]}{c^2}$$

Where:

n_0= Initial size

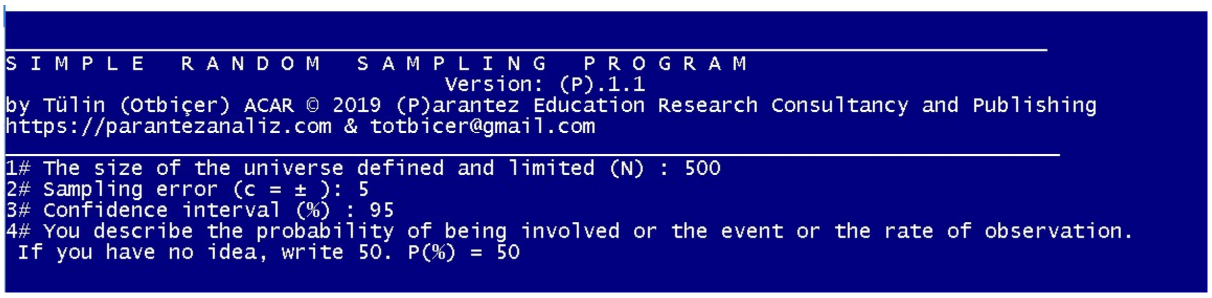
N= Universe size

Z = Standard Z value for the level of reliability (e.g. ±1.96)

c = Acceptable amount of error (e.g. ±5)

P= Estimated rate of the sampling of interest in the universe (e.g. 50%)

The algorithm written in Python based on this calculation formula is available at https://github.com/totbicer/Simple-Random-Sampling. The application Random_sampling.exe is available at https://parantezanaliz.com/programs/Random_sampling.zip. Once the **Random_sampling.exe** is run, data is input in four steps as seen in Figure 1.



```
S I M P L E   R A N D O M   S A M P L I N G   P R O G R A M
                          Version: (P).1.1
by Tülin (Otbiçer) ACAR © 2019 (P)arantez Education Research Consultancy and Publishing
https://parantezanaliz.com & totbicer@gmail.com

1# The size of the universe defined and limited (N) : 500
2# Sampling error (c = ± ): 5
3# Confidence interval (%) : 95
4# You describe the probability of being involved or the event or the rate of observation.
 If you have no idea, write 50. P(%) = 50
```

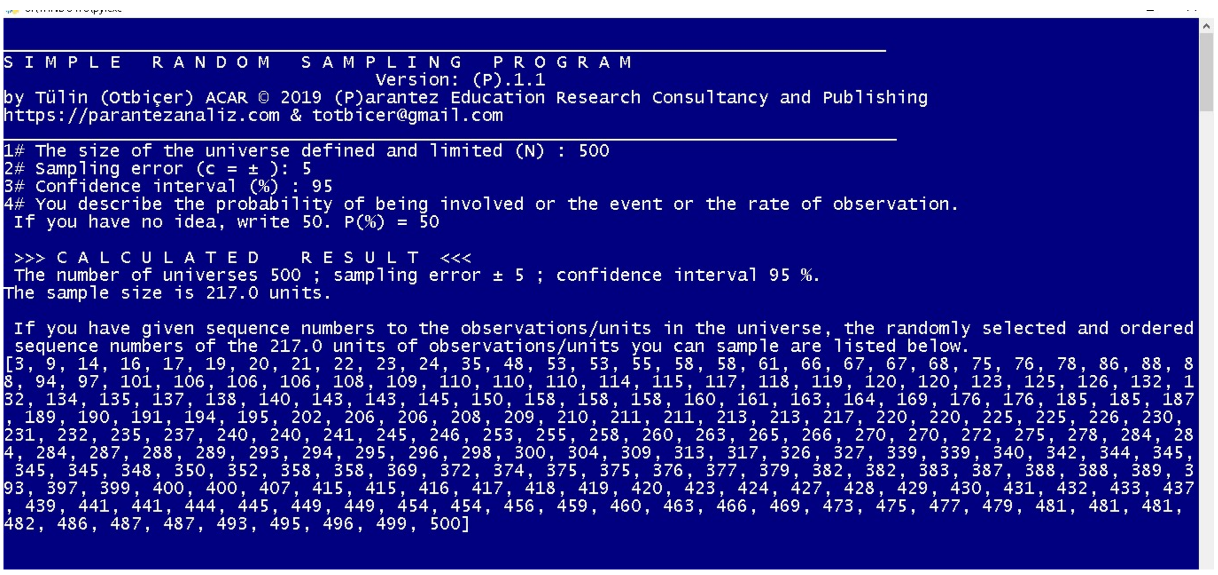
Figure 1. Screenshot of the simple random sampling program

Step 1: A numerical value is input with the defined and restricted size of the universe being a number. Assume that there are 500 units in the universe.

Step 2: A numerical value is input with the sampling error being an natural number between 1 and 100. Assume that the sampling error is 5.

Step 3: The confidence level is input. A numerical value of 95 or 99 is input. Assume that the confidence level is 95%.

Step 4: A numerical value is input with the probability or observation frequency of an event or fact of interest being a natural number. If no information is available as to the probability of happening or being observed, the value should be 50. Thus, the probability of happening or being for each of the observations is equal. For instance, 50 is input for a probability of 50%, 30 is input for a probability of 30%.



```
S I M P L E   R A N D O M   S A M P L I N G   P R O G R A M
                          Version: (P).1.1
by Tülin (Otbiçer) ACAR © 2019 (P)arantez Education Research Consultancy and Publishing
https://parantezanaliz.com & totbicer@gmail.com

1# The size of the universe defined and limited (N) : 500
2# Sampling error (c = ± ): 5
3# Confidence interval (%) : 95
4# You describe the probability of being involved or the event or the rate of observation.
 If you have no idea, write 50. P(%) = 50

 >>> C A L C U L A T E D   R E S U L T  <<<
 The number of universes 500 ; sampling error ± 5 ; confidence interval 95 %.
The sample size is 217.0 units.

 If you have given sequence numbers to the observations/units in the universe, the randomly selected and ordered
 sequence numbers of the 217.0 units of observations/units you can sample are listed below.
[3, 9, 14, 16, 17, 19, 20, 21, 22, 23, 24, 35, 48, 53, 53, 55, 58, 58, 61, 66, 67, 67, 68, 75, 76, 78, 86, 88, 8
8, 94, 97, 101, 106, 106, 106, 108, 109, 110, 110, 110, 114, 115, 117, 118, 119, 120, 120, 123, 125, 126, 132, 1
32, 134, 135, 137, 138, 140, 143, 143, 145, 150, 158, 158, 158, 160, 161, 163, 164, 169, 176, 176, 185, 185, 187
, 189, 190, 191, 194, 195, 202, 206, 206, 208, 209, 210, 211, 211, 213, 213, 217, 220, 220, 225, 225, 226, 230,
231, 232, 235, 237, 240, 240, 241, 245, 246, 253, 255, 258, 260, 263, 265, 266, 270, 270, 272, 275, 278, 284, 28
4, 284, 287, 288, 289, 293, 294, 295, 296, 298, 300, 304, 309, 313, 317, 326, 327, 339, 339, 340, 342, 344, 345,
 345, 345, 348, 350, 352, 358, 358, 369, 372, 374, 375, 375, 376, 377, 379, 382, 382, 383, 387, 388, 388, 389, 3
93, 397, 399, 400, 400, 407, 415, 415, 416, 417, 418, 419, 420, 423, 424, 427, 428, 429, 430, 431, 432, 433, 437
, 439, 441, 441, 444, 445, 449, 449, 454, 454, 456, 459, 460, 463, 466, 469, 473, 475, 477, 479, 481, 481, 481,
482, 486, 487, 487, 493, 495, 496, 499, 500]
```

Figure 2. Result of the simple random sampling program

When the operation is continued, a simple random sampling is made to show the computed sampling size as shown in Figure 2. Based on the assumption that units in the universe are represented by ordinal numbers, ordinal numbers of the units randomly selected are listed on the screen.

2.2 Determination of Observation Units by Systematic Sampling

Based on the simple random sampling method or a sampling size determined hypothetically, the unit selection process is performed by the following equation (Babbie, 2008).

$$k = N / n \quad \text{and} \quad Ssn = \text{It is determined randomly between 1 and k value.}$$

Where:

N= Universe size

n= Sampling size

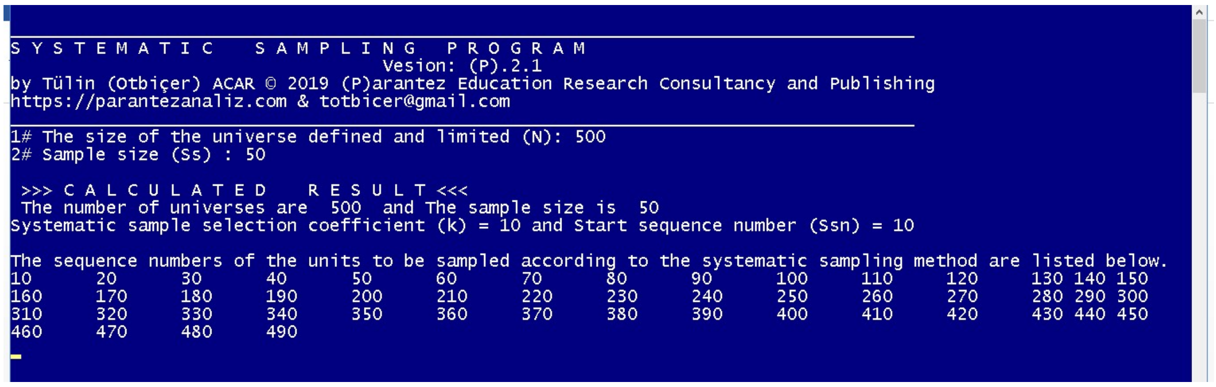
k= Selection coefficient

Ssn= Starting sequence number

The algorithm written in Python based on this calculation formula is available at https://github.com/totbicer/Systematic-Sampling. The application Systematic_sampling.exe is available at https://parantezanaliz.com/programs/Systematic_sampling.zip. Once the **Systematic_sampling.exe** is run, data is input in two steps as seen in Figure 3.

Step 1: A numerical value is input with the defined and restricted size of the universe being a number. Assume that there are 500 units in the universe.

Step 2: A numerical value is input with the sampling size being a natural number. Assume that sampling size is 50.



```
S Y S T E M A T I C    S A M P L I N G   P R O G R A M
                                 Vesion: (P).2.1
by Tülin (Otbiçer) ACAR © 2019 (P)arantez Education Research Consultancy and Publishing
https://parantezanaliz.com & totbicer@gmail.com

1# The size of the universe defined and limited (N): 500
2# Sample size (Ss) : 50

 >>> C A L C U L A T E D    R E S U L T <<<
 The number of universes are  500  and The sample size is  50
Systematic sample selection coefficient (k) = 10 and Start sequence number (Ssn) = 10

The sequence numbers of the units to be sampled according to the systematic sampling method are listed below.
10      20      30      40      50      60      70      80      90      100     110     120     130 140 150
160     170     180     190     200     210     220     230     240     250     260     270     280 290 300
310     320     330     340     350     360     370     380     390     400     410     420     430 440 450
460     470     480     490
```

Figure 3. Screenshot of the systematic sampling program

As can be seen in Figure 3, the selection coefficient and the randomly selected initial number are shown on the screen. Based on the assumption that units in the universe are represented by ordinal numbers, ordinal numbers of the units selected by the systematic sampling method are listed on the same screen.

2.3 Determination of the Sizes of the Strata with Stratified Sampling

In the stratified sampling method, if a sampling size is not available, the sampling size is determined firstly by the simple random sampling method. Based on a known or determined sampling size and the number of strata, observation units are selected by the equations below (Kalton, 1983).

$$W_h = N_h / N \qquad \text{and} \qquad F_h = W_h * Ss$$

Where:

h= Number of stratum

N_h= h. sampling size of stratum

N= Universe size, $\sum N_h$

W_h = Representation rate of the stratum of the universe, $\sum W_h = 1$

Ss= Sampling size

F_h= h. sampling size for the stratum

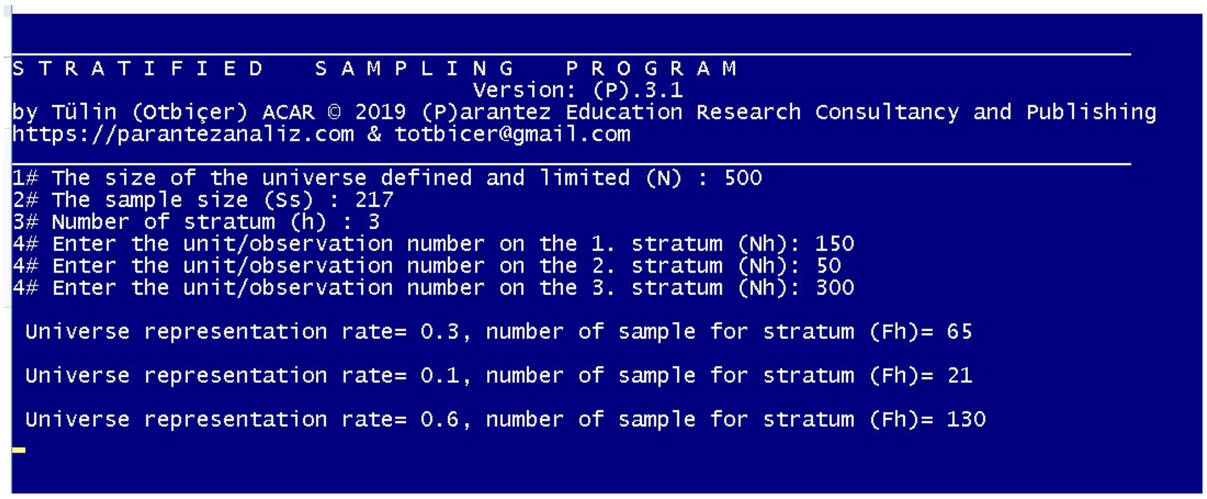
The algorithm written in Python based on this calculation formula is available at https://github.com/totbicer/Stratified-Sampling. The application Stratified_sampling.exe is available at https://parantezanaliz.com/programs/Stratified_sampling.zip. Once the **Stratified_sampling.exe** is run, data is input in four steps as seen in Figure 4.

Step 1: A numerical value is input with the defined and restricted size of the universe being a number. Assume that there are 500 units in the universe.

Step 2: A numerical value is input with the sampling size being a natural number. Assume that sampling size is 217.

Step 3: A numerical value is input with the number of strata being a natural number. For instance, assume that there are 3 strata in the universe.

Step 4: A numerical value is input with the number of units/observations for each stratum being a natural number. For instance, assume that there are 150, 50 and 300 observations for each stratum.



```
______________________________________________________________________________________
S T R A T I F I E D   S A M P L I N G   P R O G R A M
                                   Version: (P).3.1
by Tülin (Otbiçer) ACAR © 2019 (P)arantez Education Research Consultancy and Publishing
https://parantezanaliz.com & totbicer@gmail.com
______________________________________________________________________________________
1# The size of the universe defined and limited (N) : 500
2# The sample size (Ss) : 217
3# Number of stratum (h) : 3
4# Enter the unit/observation number on the 1. stratum (Nh): 150
4# Enter the unit/observation number on the 2. stratum (Nh): 50
4# Enter the unit/observation number on the 3. stratum (Nh): 300

 Universe representation rate= 0.3, number of sample for stratum (Fh)= 65

 Universe representation rate= 0.1, number of sample for stratum (Fh)= 21

 Universe representation rate= 0.6, number of sample for stratum (Fh)= 130
```

Figure 4. Screenshot of the stratified sampling program.

The rate of representation of each stratum of the universe and the number of units to be sampled for the strata are output on the screen. The process of selecting observation units of a specified number for the strata may also be performed randomly or systematically.

3. CONCLUSION

Execution files in exe format written in Python 3 were prepared to determine the sampling size and observation units for the probability sampling methods. Each program was shared as as open source in trial version for a year. Updates were made during the trial period to ensure that the language and the expressions were clear and comprehensible. No software error was reported during the trial period.

The programs may be run by double-clicking on the Windows operating system or by the command prompt of the DOS operating system. Each exe file has a memory space of 5 megabits on average. In this respect, the application files are very useful in terms of sharing by email and mobility by USB memory sticks.

ORCID

Tülin ACAR https://orcid.org/0000-0001-7976-5521

4. REFERENCES

Babbie, E. (2008). *The basics of social research.* USA: Thomson Coperaion

Kalton, G. (1983). *Introduction to survey sampling.* , USA: Sage Publications Ltd

Royse, D., Thyer, B. A. & Padgett, D. (2010). *Program evaluation: An ıntroduction.* USA: Cengage Learning

Rubin, A. & Babbie, E.R. (2010). *Essential research methods for social work.* USA: Cengage Learning

Sahoo, R. & Sahoo, G. (2016). *Computer science with python.* India: Newsaraswati House Pvt. Ltd.

Thomas, G. (2009). *How to do your research project*: A *guide for students in education and applied social sciences*. Londan: Sage Publications Ltd

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.

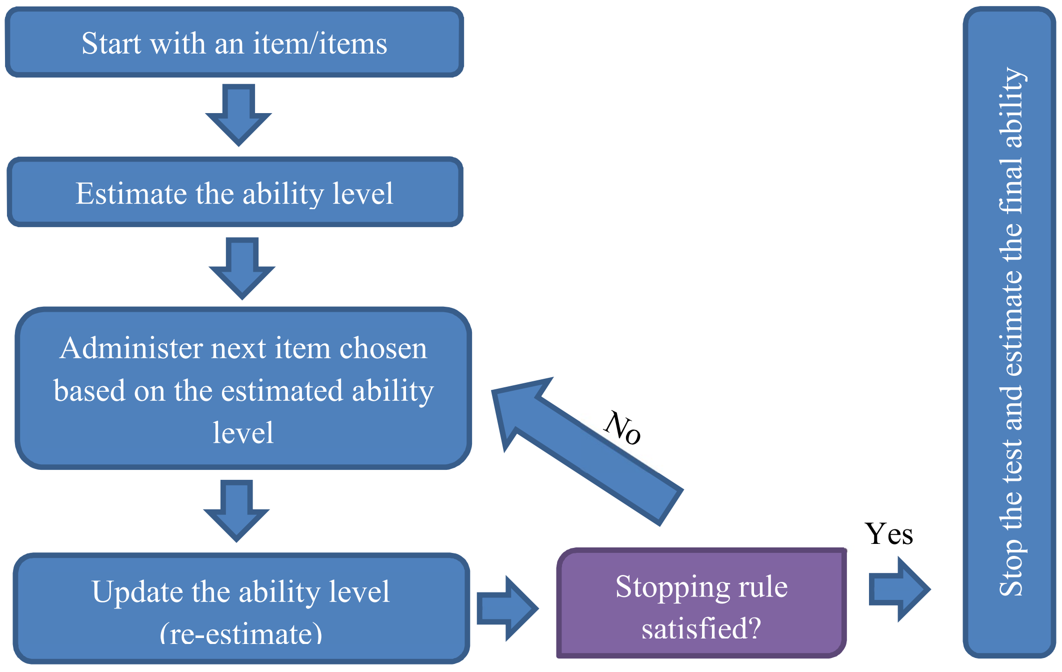

Start with an item/items
Estimate the ability level
Administer next item chosen based on the estimated ability level
No
Update the ability level (re-estimate)
Stopping rule satisfied?
Yes
Stop the test and estimate the final ability

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.

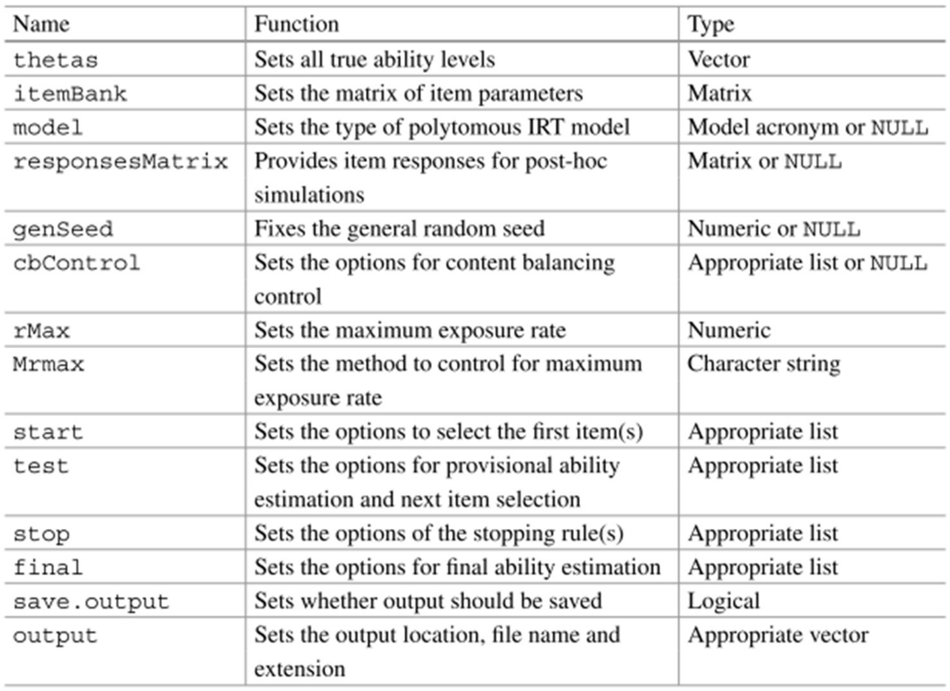
Input arguments in *simulateRespondents()* function are summarized by Magis, Yan and von Davier (2017) and given in Figure 2. Detailed information is provided for the related arguments inside the function.

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.

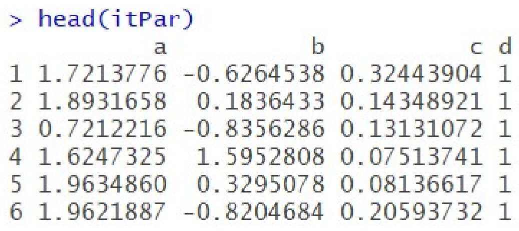
An item pool of 300 items was generated according to three-parameter logistic model. R codes are presented in Table 1.

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
```

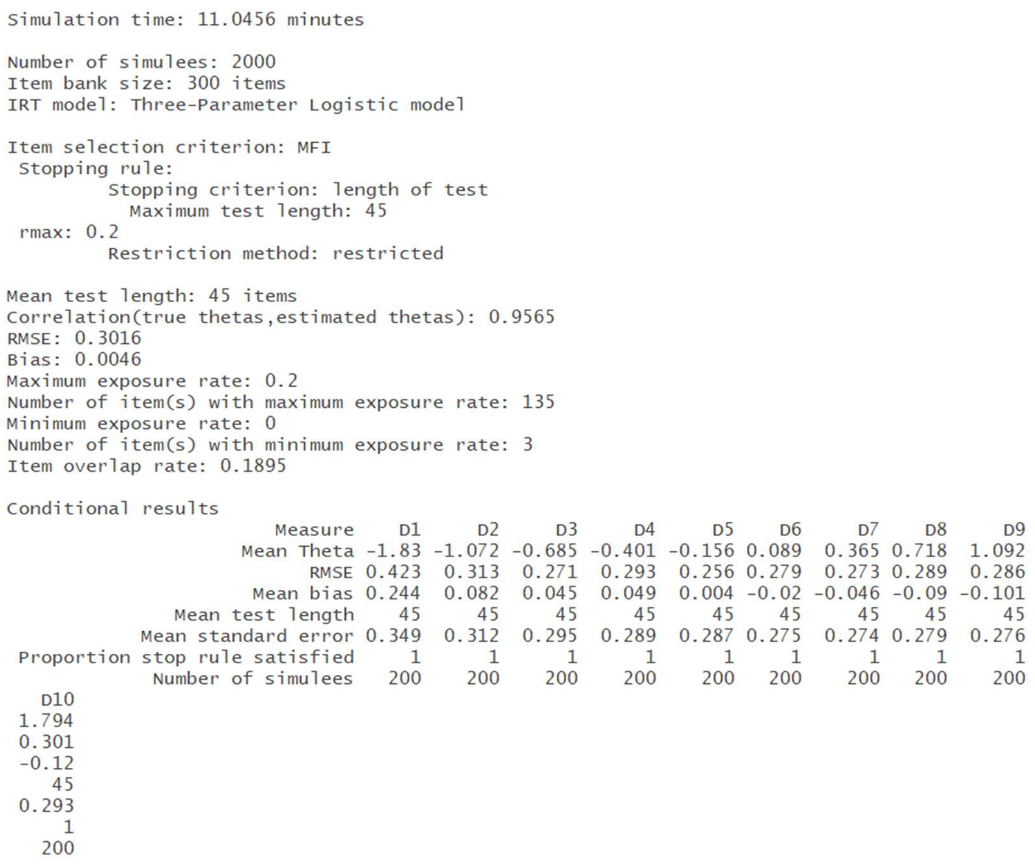
Figure 3. Generated item parameters of first six items

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.

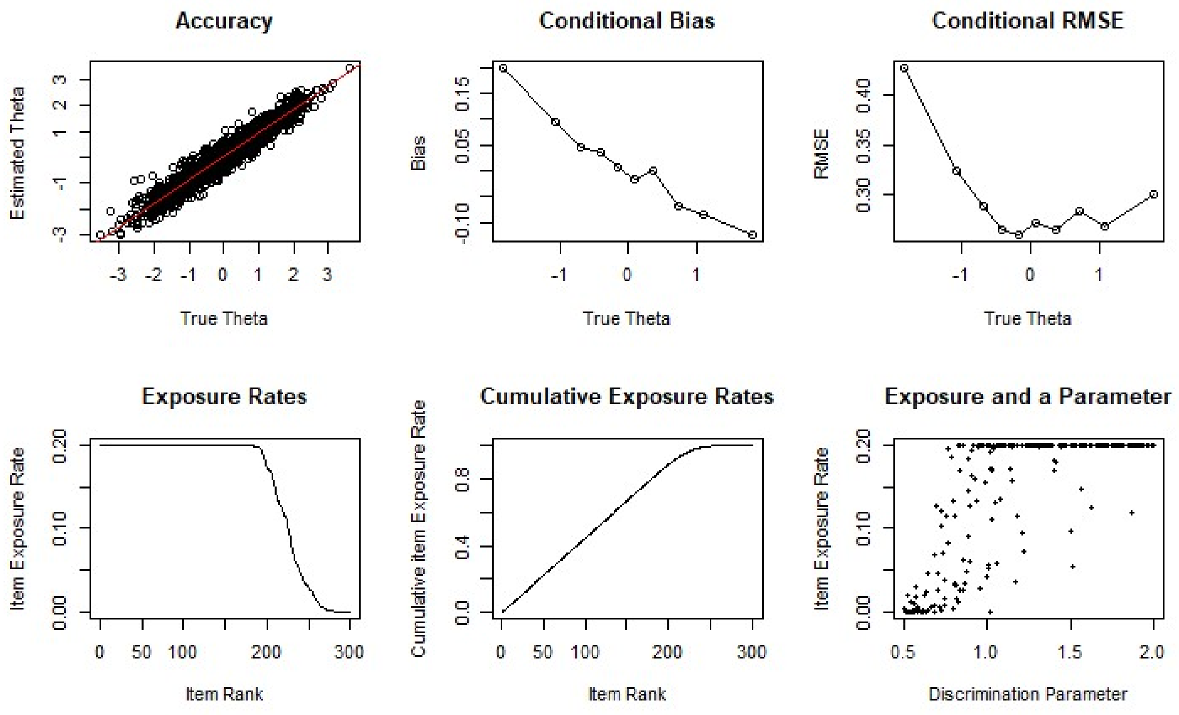


Figure 5. The graphical representation of simulation

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.

- <u>Accuracy:</u> 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.
- <u>Conditional bias</u>: 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.
- <u>Conditional RMSE</u>: 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.
- <u>Exposure rates</u>: 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.
- <u>Cumulative exposure rates</u>: 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.
- <u>Exposure and a parameter</u>: 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*.

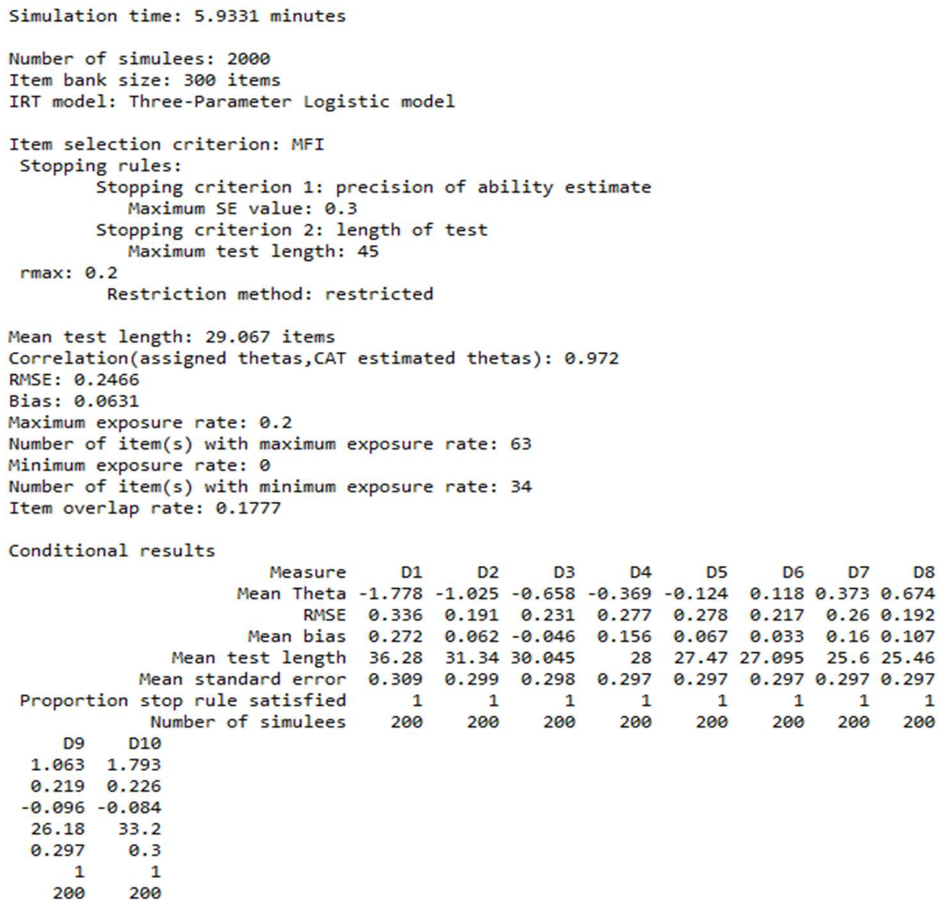
In order to see a different aspect, stopping rule was updated and rather than that everything was same such as ability estimation rule, item selection rule, item exposure issues etc. The code for the simulation process is given below in Table 6. The results were given in the output in Figure 6 below.

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

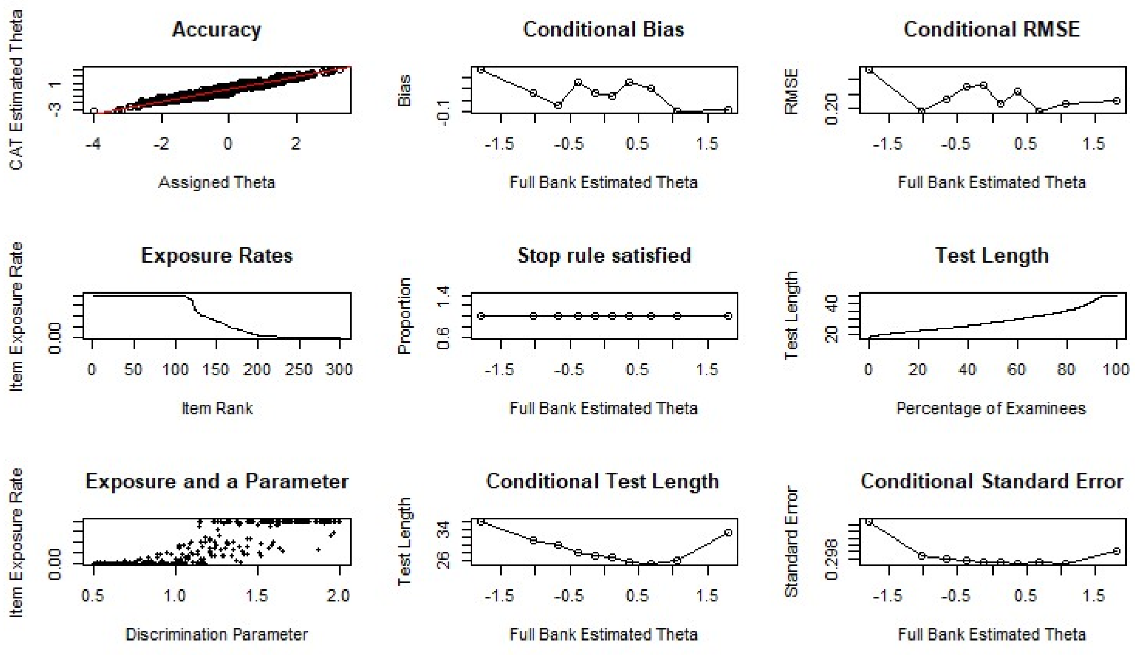


Figure 7. The graphical representation of simulation results

- 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.

International Journal of Assessment Tools in Education
2019, Vol. 6, No. 5-Special Issue, 57–69
https://dx.doi.org/10.21449/ijate.658747

Published at http://www.ijate.net http://dergipark.gov.tr ***Research Article***

Coping with Unbalanced Designs of Generalizability Theory: G String V

Gülşen Taşdelen Teker [1,*]

[1]Hacettepe University, Faculty of Medicine, Department of Medical Education and Informatics, Ankara, Turkey

ARTICLE HISTORY

Received: 28 October 2019

Accepted: 15 December 2019

KEYWORDS

Generalizability theory,
G String V,
Unbalanced design

Abstract: The aim of this paper is to introduce a software that is appropriate for the generalizability theory for not only balanced but also unbalanced data sets. Because it is possible to have unbalanced data sets while conducting a study, the researchers have devised an easy solution, other than deleting data, to balance the design to cope with this situation. Thus, the software G String V will be introduced. First, the generalizability theory will be reviewed, followed by a description of the unbalanced synthetic data that was used to conduct the analysis using the software. Explanations are provided for installing the software, preparation of the data, and the step-by-step data analysis. Moreover, the interpretation of the data is also explained. Finally, the limitations of the software are shared.

1. INTRODUCTION

Generalizability (G) theory, which was developed by Cronbach, Gleser, Nanda and Rajaratnam (1972) as an alternative to the classical test theory (CTT), is a statistical theory for evaluating the dependability or reliability of behavioral measurements (Brennan, 2001a; Shavelson and Webb, 1991). Conceptually, the G theory can be regarded as a multifaceted extension of the CTT and can be seen as a combination of the CTT and variance analysis (Brennan, 2000; Suen & Lei, 2007). Reliability, which is defined in the CTT as the consistency of the scores obtained through measurements, can vary according to the source to which the error is connected. For example, (i) when designing a reliability study to produce two sets of observations, one might give the same test two times, separated by two weeks: test-retest reliability; (ii) designing a reliability study to create two parallel forms of the test, as Form 1 and Form 2, and give the two forms of the test on the same day: parallel forms reliability; or (iii) calculating the reliability of a single form of a test on a single occasion: split-half reliability. Although there are multiple sources of error for these three examples, the CTT takes only one error source as time, forms, and items, respectively. In other words, the errors in the measurement results are considered as the errors coming from only one source of variability, and this emerges as a restriction of the CTT. Because the G theory can consider several sources of error simultaneously, estimations can be made more accurately than the ones in the CTT.

CONTACT: Gülşen TAŞDELEN TEKER ✉ gulsentasdelen@gmail.com Hacettepe University, Faculty of Medicine, Department of Medical Education and Informatics, Ankara, Turkey

ISSN-e: 2148-7456 /© IJATE 2019

G theory is structured around different sources of variation, called facets. There are two main types of facets, which are *facet of differentiation* and *facet of instrumentation*, according to Brennan's (2001) classification. In G theory, a behavioral measurement is conceived of as a sample from a u*niverse of admissible observations,* which consists of all possible observations on an *object of measurement* that a decision maker considers to be acceptable substitutes for the observation in hand. The object of measurement is also referred to as the facet of differentiation. You can easily determine the object of measurement of your research by answering the question: "What are you trying to attach the measurement to?" The answer can be students, items, etc. For instance, if you want your friends (n=10) to rate a number of dark chocolates (n=3) on five three-point scales, the object of measurement is dark chocolate. If you want to get patients' satisfaction with their experiences on a hospital's inpatient ward, the object of measurement is ward. Alternatively, as a more familiar example, if you want to evaluate the students' performances in a classroom activity, then the object of measurement is student. Whatever the object of measurement is, there is only one per analysis.

Although there are some researchers who use Brennan's classification of facets (Cardinet, Johnson & Pini, 2010: Cardinet, Tourneur & Allal, 1981), the facet of instrumentation is also referred to as the facet of generalization in some sources (Bloch & Norman, 2015; Bloch & Norman, 2012; Cardinet, Tourneur & Allal, 1976). This is acceptable and is also used in G String terminology. Every observation of object of measurement is subject to error, derived from various sources (Bloch & Norman, 2015). These sources are also called *facets of instrumentation* and address the following question: "To what extent can I generalize a measurement from one situation to another with a different level of the facet of instrumentation?" There are two types of instrumentation facets, called fixed and random. Typically, a random facet is created by randomly sampling levels of a facet. Meanwhile, when the levels of a facet have not been sampled randomly from the universe of admissible observations, and the intended universe of generalization is infinitely large, the concept of exchangeability may be invoked to consider the facet as random (Shavelson & Webb, 1981). A fixed facet arises in three conditions: (a) purposely selecting certain conditions and not interested in generalizing beyond them, (b) finding it unreasonable to generalize beyond the levels observed, or (c) when the entire universe of levels is small and all levels are included in the measurement design (Shavelson & Webb, 2006). G theory is essentially a random effects theory. Therefore, there should be at least one random facet in the data set.

According to Brennan (2001, p.108) the rules and equations of G theory assume that the objects of measurement are not nested within some other facet. However, G theory can treat such nested, or stratified, objects of measurement, but requires special consideration to do so. Objects of measurement are stratified with respect to some other variable, and an investigator may be interested in variability within levels of the stratification variable, as well as the variability across levels (Brennan, 2001, p.153). For instance, assume that there are 100 people in each of four regions (east, west, south, and north). Here, people are the object of measurement, and they are nested in regions. According to Brennan (2001) it is quite complex to cope such designs. Moreover, it is stated that if the design is unbalanced the procedures discussed do not apply, and appropriate procedures are much more complicated. Bloch and Norman (2018) defined this situation by using another term. According to them, when the facet of differentiation (object of measurement) is nested within another facet, this facet is referred to as a *facet of stratification*. For instance, there are students who are at different educational levels (senior vs. junior students). Commonly the difference between the two groups is viewed as a test of construct validity. However, in terms of reliability, the person variance should be computed within educational levels (we want to see if we can differentiate among individuals at the same level). Here, the educational level is a stratification facet (Bloch and Norman (2018). Although the term "stratification facet" in not in Brennan's (2001a) Generalizability Theory book, which can be defined as the bible of the G theory, it is important to explain the stratification facet because G String V also uses this terminology.

There is also specific terminology associated with the design: crossed design and nested design. Assume all students (s) respond to all the items (i) in a test so that students are crossed with items. We denote this design as *s x i*. However, if each student responds to a different set of items then it is expressed as items nested within students. We denote this design as *i : s*. A crossed design is usually preferred in studies conducted using G Theory. The reason for this is that all sources of error, associated with all probable facets and the interactions between those facets, can be estimated in crossed-designed studies.

The importance of G theory lies in its applications to educational measurement. There are two major functions of G theory. One of them is to evaluate the quality of measurement procedures and the other is to make projections about how one can improve the quality of measurement procedures (Chiu, 2001). The former function can be done through the generalizability (G) study and it is possible to attain the second function via a decision (D) study. In other words, to evaluate the dependability of behavioral measurements, a G study is designed to isolate and estimate variation due to the object of measurement while examining as many facets of measurement error as possible. A D study uses the information provided by the G study to design the best possible application of the measurement for a particular purpose (Webb, Shavelson & Haertel, 2006) and answers the question "What if…?" by designing variations in measurement via optimization (Brennan, 2001). While planning the D study, the researcher defines a *universe of generalization*, the set of facets and their levels to which he or she wants to generalize, and specifies the proposed interpretation of the measurement. The decision maker uses the information from the G study to evaluate the effectiveness of alternative designs for minimum error and maximum reliability (Webb, Shavelson & Haertel, 2006).

Although there are wide applications of the theory, it has limitations in its capability of handling unbalanced designs of the data. The number of observations in balanced designs is equal at each level for the source of variability (Brennan, 2001a). By contrast, an unbalanced design has unequal numbers of observations in its sub-classifications. For instance, there can be differing numbers of items nested within testlets, pupils nested within differently sized classrooms, or observers nested within occasions with an unequal number of observers present at each occasion. These three examples are defined as unbalanced because the nested designs may be purposely unbalanced, dictated by the context itself or created by unforeseen circumstances, respectively. One other reason for unbalanced situations can be missing observations from crossed and nested designs (Webb, Shavelson & Haertel, 2006). As sample sizes tend to be small in the G theory analyses (Rios, Li & Faulkner-Bond, 2012; Taşdelen Teker & Güler, 2019), missing data becomes an important topic. Researchers normally prefer listwise deleting, inputing missing observations, or employing unbalanced designs to deal with missing data. Shavelson and Webb (1991) encouraged deleting data to create a balanced design to circumvent estimation challenges. Shavelson, Webb and Rowley (1989) found little effect in the estimated variance components when data was deleted to create balance. However, it can be problematic for very small sample sizes, such as less than 20. Rios, Li and Faulkner-Bond (2012) conducted a systematic review of the most recently published literature to understand the current methodological trends in the G theory better. Unbalanced design was used in 19 of 58 studies reviewed. Taşdelen Teker and Güler (2019) conducted a thematic content analysis of studies using the G theory in the field of education in Turkey and found that 6 of 60 studies were conducted with unbalanced design. According to the results of the above-mentioned review studies of Rios, Li and Faulkner-Bond (2012), and Taşdelen Teker and Güler (2019), the ratio of unbalanced designs is high enough to make coping with them indispensable.

Estimating variance components in unbalanced designs was challenging. Some or all methods had problems of computational complexity, distributional assumptions, and biased estimation, requiring decisions that could not be justified in the context of the G theory, or produced results that were inconclusive (Brennan, 2001). However, now it is possible to run G and D studies to estimate variance components and reliability coefficients by using the software G String V. The main purpose of this study was to introduce a computer program that was appropriate for G theory for balanced, and even more important, unbalanced data sets. According to the manual of G String V (Bloch & Norman, 2018), because the software is based on variance component estimates from urGENOVA, which was written by R. L. Brennan (2001a), the mathematical formulation declared by Brennan (2001a) will be given before introducing the software. Moreover, synthetic data taken from Brennan (2001a, p.224) will be provided to clarify the notations. Lastly, the software will be introduced step by step by using the same synthetic data shown in Table 1.

1.1. Synthetic Data and Mathematical Computations

Assume that eight students (s) take an 8-item test that is composed of three testlets (h) containing 2, 4, and 2 items (i), respectively. Because the number of items per testlet is not equal, the design is defined as unbalanced. It is a random facet nested design, symbolized as *sx(i:h)*. The data entry is shown in Table 1.

Table 1. *sx(i:h)* Unbalanced Design

Student	Testlet 1		Testlet 2				Testlet 3	
	Item 1	Item 2	Item 1	Item 2	Item 3	Item 4	Item 1	Item 2
1	4	5	3	3	5	4	5	7
2	2	1	2	3	1	4	4	6
3	2	4	4	7	6	5	8	7
4	1	3	5	4	5	5	4	5
5	3	3	6	7	5	7	8	9
6	1	2	5	6	4	4	5	6
7	3	5	6	8	6	7	7	8
8	0	1	1	2	0	4	7	8

As seen in Table 1, there is no missing data. It is strongly advised to cope with the missing data by using standard statistical approaches before using G String V. However, if the researcher forgets to deal with the missing data, then G String V will replace the missing values by the grand mean and warn the user when this occurs.

The estimation of variance components in terms of mean squares are given in Table 2. The n_{i+} notation, given under the degrees of freedom (*df*) column of Table 2, is the total number of levels of i over all levels of h; that is, $n_{i+} = \sum_h n_{i:h}$. Moreover, r_i and t_i, which are used for the estimation of variance components in terms of mean squares, are computed by using the following equations: $r_i = \sum_h \frac{n_{i:h}^2}{n_{i+}}$ and $t_i = \frac{n_{i+}-r_i}{n_h-1}$.

Table 2. *sx(i:h)* Unbalanced Design

Source of Variance	df	Mean Squares	Estimators of the variance components in terms of mean squares
s	n_s-1	MS_s	$\sigma^2_s= [MS_{(s)}-r_iMS_{(sh)}/t_i+(r_i-t_i)MS_{(si:h)}/t_i]/n_{i+}$
h	n_h-1	MS_h	$\sigma^2_h= [MS_{(h)}-MS_{(i:h)}-MS_{(sh)}+MS_{(si:h)}]/n_st_i$
i:h	n_{i+} - n_h	$MS_{i:h}$	$\sigma^2_{i:h}= [MS_{(i:h)}-MS_{(si:h)}]/n_s$
sh	$(n_s-1)(n_h-1)$	MS_{sh}	$\sigma^2_{sh}= [MS_{(sh)}-MS_{(si:h)}]/t_i$
si:h	$(n_s-1)(n_{i+} -n_h)$	$MS_{si:h,e}$	$\sigma^2_{si:h,e}= MS_{(si:h,e)}$

By using the variance components obtained from the G study, it is possible to estimate relative and absolute error variances. After that, the error variances are used to estimate the generalizability and dependability coefficients. The equations used for the estimations of relative/absolute error variances and generalizability/dependability coefficients are given below. Equation 1 is used to estimate the relative error variance ($\sigma^2(\delta)$) and Equations 2 and 3 are used to compute generalizability coefficients ($E\rho^2$) for unbalanced sx(i:h) design. Equation 4 is used for the estimation of absolute error variance. Then Equations 2 and 5 are used for the estimation of index of dependability (ϕ). The $\breve{n}_h$ term used for the estimation of relative and absolute error variances is equal to $\breve{n}_h = \frac{n_{i+}^2}{\sum_h n_{i:h}^2}$.

$$\sigma^2(\delta) = \frac{\sigma_{sh}^2}{\breve{n}_h} + \frac{\sigma_{si:h}^2}{n_{i+}} \quad [1]$$

$$\sigma^2(\tau) = \sigma^2(s) \quad [2]$$

$$E\rho^2 = \frac{\sigma^2(\tau)}{\sigma^2(\tau) + \sigma^2(\delta)} \quad [3]$$

$$\sigma^2(\Delta) = \frac{\sigma_h^2}{\breve{n}_h} + \frac{\sigma_{i:h}^2}{n_{i+}} + \frac{\sigma_{sh}^2}{\breve{n}_h} + \frac{\sigma_{si:h}^2}{n_{i+}} \quad [4]$$

$$\phi = \frac{\sigma^2(\tau)}{\sigma^2(\tau) + \sigma^2(\Delta)} \quad [5]$$

2. THE SOFTWARE: G STRING V

G String V (Bloch & Norman, 2018) is a software that functions on the basis of urGENOVA (Brennan, 2001b) and is used in G theory analyses. Because urGENOVA is a traditional command line program that does not have a graphical user interface, users must specify their parameters, which makes it difficult to work with. Moreover, although urGENOVA provides the variance components for the individual effects, it does not calculate variance coefficients under different conditions. However, G String V does this as well (Bloch & Norman, 2018). G String V was designed and coded by Ralph Bloch as part of a project commissioned by The Medical Council of Canada and was subsequently further developed. It is written in Java on the Linux platform. The most recent version of the program runs under the Windows operating system (Bloch & Norman, 2015) and Macintosh and Linux operating systems (Bloch & Norman, 2018). The G-String V has a more user-friendly interface and therefore, is much easier to use compared to urGENOVA.

2.1. Installing the Software

G String V can be downloaded for free from the Web. Researchers may install the latest version of the G String V software, released in July 2018 from https://healthsci.mcmaster.ca/merit/research/g_string_v. The program is contained in a software package called "G_String_V.jar".

Before downloading the software, install Java Runtime JRE 8 on your computer if it is not already installed. Then create a new folder called "G_String_V" in a suitable location of your file system. After selecting your computer's operating system (Windows, Mac-OS or Linux), download the software package and copy it from the Downloads folder into the G_String_V folder. Next, create a new sub-folder within the G_String_V folder called "work." Then double click on G_String_V. jar. As shown in Figure 1, set the "work" sub-folder as your working directory by clicking the Setup and Set Working Directory buttons.

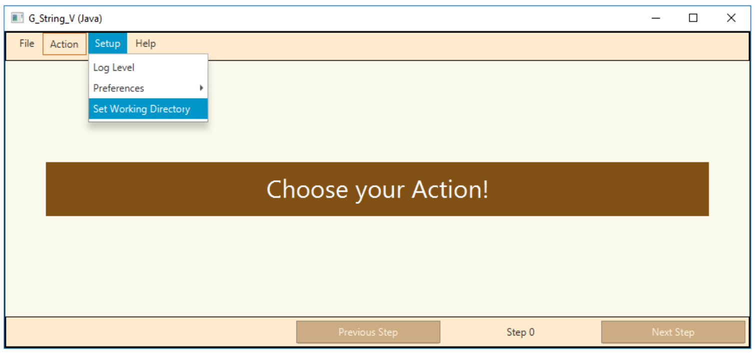


Figure 1. G String V Software Working Directory Setup Screen

2.2. Software Interface

The main interface of the G String V software is shown in Figure 2. This interface consists of the main menu that includes File, Action, Setup, and Help. To start the analysis, click on Action. There are three sub-menus under Action as seen in Figure 2. To start a new analysis click on Action→ Start New and create a new G String V run. If you want to do multiple runs on a pre-used data base then click on Use Existing. For the G String V to automatically count the number of levels of each facet, which can be helpful for unbalanced nested designs, select Auto Index.

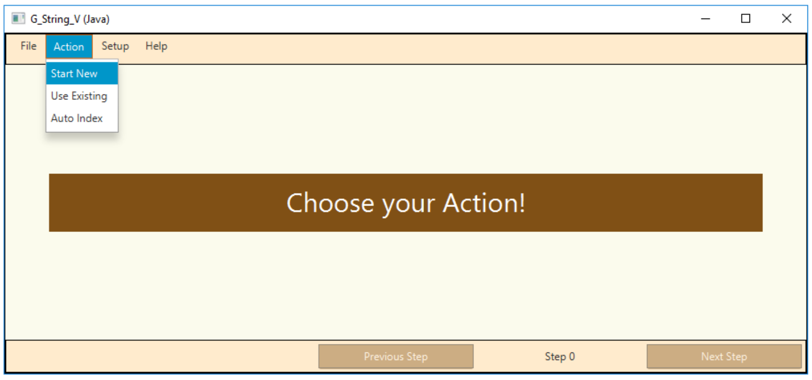


Figure 2. G String V Software Main Screen

2.3. Data Analysis using G String V

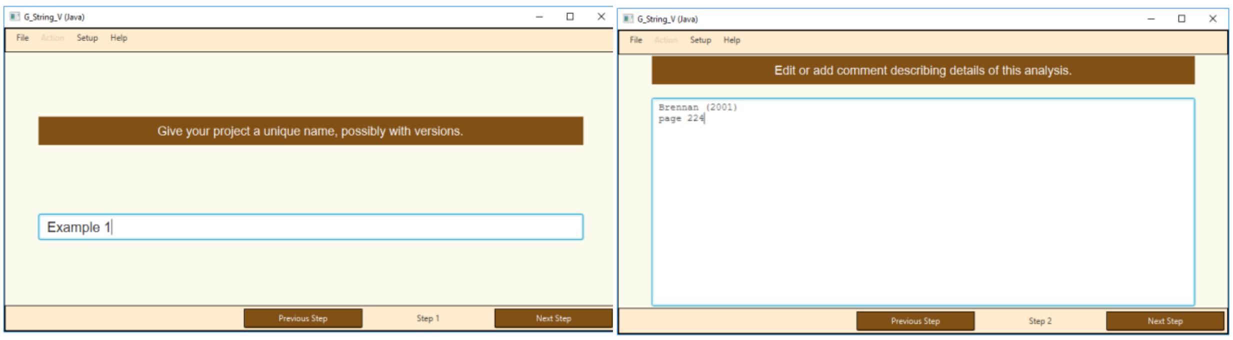
If starting with a new data set click on Start New as seen in Figure 2. From this point on, analyzing with the G String V will be explained step by step. As seen in Table 1, Brennan's example (2001, p. 224) will be used to better illustrate the steps. There are three testlets (h) containing 2, 4, and 2 items (i), respectively; and answers to all of the questions for the eight students (s). Because the items are nested within testlets and students answer all of the items, the design can be symbolized as *sx(i:h)*. Moreover, because the number of items per testlet is not the same, the design is unbalanced, as previously stated.

2.3.1. *Preparing Data for Analysis*

While urGENOVA requires the data to be in ASCII text files (.dat or .txt), G String V is set up to handle tab-delimited or fixed format text files. ASCII files can be easily generated from a spreadsheet, such as Excel, by simply saving as a “Text - tab delimited (*.txt)” file (Bloch & Norman, 2018). Like all previous versions of G String V and other software used for G theory analysis (SPSS and EduG), G String V requires that the data be ordered, so that all records related to a particular level of a facet are together. After entering the data in an Excel spreadsheet, and saving as Text - tab delimited (*.txt) it can be used to run G theory analysis via G String V.

Steps 1 and 2: Title and details of the conducted research

As seen in Figure 3, you can provide a unique name for your research in Step 1 and add more comments to describe the details of the analysis in Step 2. Comments provide an explanation of the study to the reader of the output. The information written here does not affect the computations of the software, so if you do not want to enter any information omit this step by clicking the Next Step button. To exemplify how it would appear in the output file, a title and brief description of the data was entered.



Give your project a unique name, possibly with versions.

Example 1

Edit or add comment describing details of this analysis.

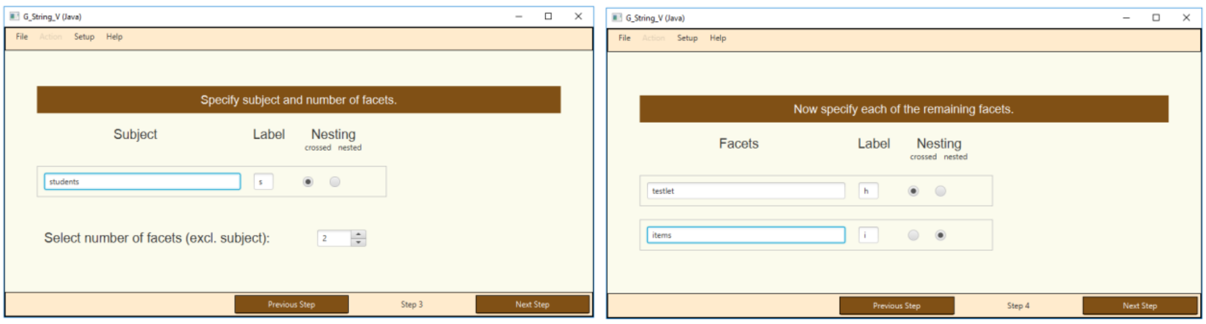
```
Brennan (2001)
page 224
```

Figure 3. G String V Software: (a) Step 1 and (b) Step 2

Steps 3 and 4: Defining object of measurement and facets

As seen in Figure 4(a), the object of measurement is specified in Step 3 by providing a descriptive name and a corresponding one-character lowercase abbreviation for the object of measurement. Although the object of measurement is usually crossed with other facets, it may also be nested within another facet. For instance, as given in Brennan's example (2001, p.154), people can be nested within regions. Because of this, the nesting situation of the object of measurement also should be specified. For the example discussed here, "crossed" should be selected. Then specify the number of facets. For the example used here, there are two, testlet and item. Each facet should be given a descriptive name and a one-character abbreviation in Step 4, as seen in Figure 4(b). Moreover, the nested facets are specified by changing the default "crossed" to "nested." In our example, because the items are nested within testlets, the nesting conditions of items have "nested" selected. The nesting condition of testlets remains "crossed" by default.

The order of the facets is also important. They must be listed in the order they are encountered in the data file, from slowest-moving to fastest-moving (Bloch & Norman, 2018). In other words, the first facet to be declared is the one whose levels change least rapidly and the last facet to be declared would be the one whose levels change the fastest (Cardinet, Johnson & Pini, 2010). In our example, the first facet is testlet as it changes more slowly than items. More clearly, if the data have one record per student, with all data for each testlet, then the responses on each item of testlet, the order of facets would be: Testlet, Item.



Specify subject and number of facets.

Subject	Label	Nesting (crossed / nested)
students	s	● crossed

Select number of facets (excl. subject): 2

Now specify each of the remaining facets.

Facets	Label	Nesting (crossed / nested)
testlet	h	● crossed
items	i	● nested

Figure 4. G String V Software: (a) Step 3 and (b) Step 4

Steps 5 and 6: Setting order of facets and arrange nesting of the facets

In Step 5, the order of facets should be specified as they appear in the data set. For instance, because each student's answers to eight items are listed in one row, the asterisk is beside student as seen in Figure 5(a). In Step 6, drag and drop the nested facets from the left side to the right side. By doing so, the nested facets are located under the facet in which they are nested nested as seen in Figure 5(b). Because items are nested within testlets in our example, the item facet is dragged and dropped under the testlet facet and appears as i:h.

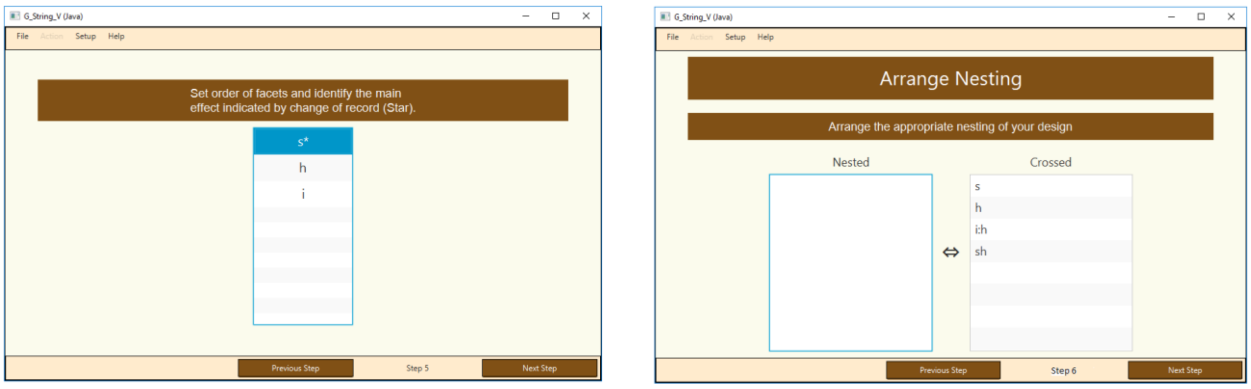
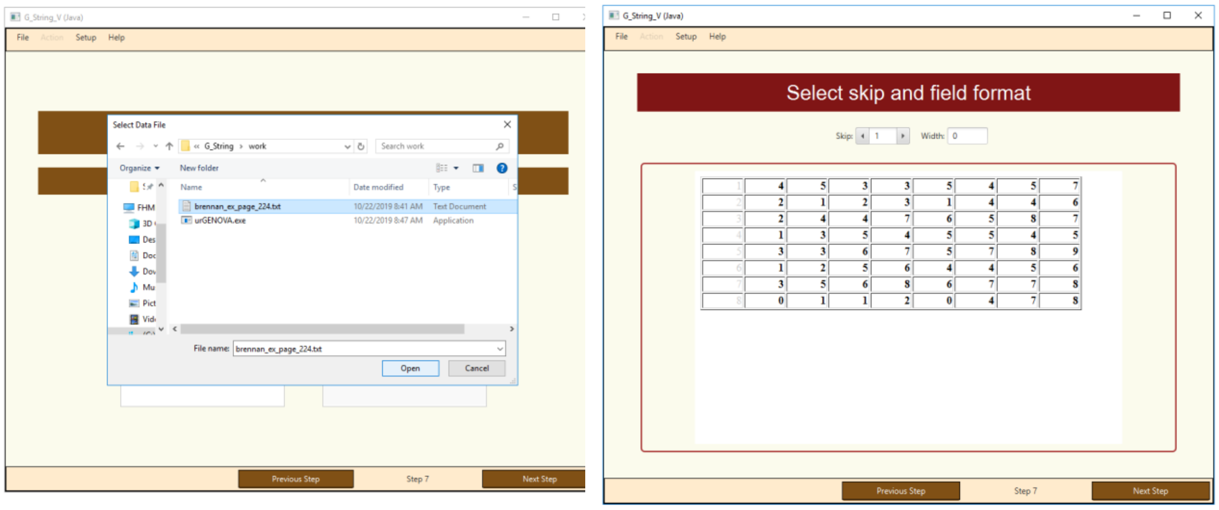

G_String_V (Java)
File
Action
Setup
Help
Set order of facets and identify the main effect indicated by change of record (Star).
s*
h
i
Previous Step
Step 5
Next Step
Arrange Nesting
Arrange the appropriate nesting of your design
Nested
Crossed
s
h
i:h
sh
Step 6

Figure 5. G String V Software: (a) Step 5 and (b) Step 6

Step 7: Locating data file

As seen in Figure 6(a), the exact location of the data file is selected. The data must be in an ASCII text file. To do so, you can enter your data in Excel and save it as "Text (Tab delimited) (*.txt)." After selecting the location of the data, you will see it on the screen. As seen in Figure 6(b), there are nine columns in the data file. The first column contains the student ID, which means the actual data begins in the second column. To indicate how many columns are to be skipped, enter the information in the "Skip" field. As you can see from the second screenshot of Figure 6, "1" is in the "Skip" area, so the first column of the data becomes colorless, which means it will not be analyzed. You can only skip fields at the beginning of the data, never in the middle.



1	4	5	3	3	5	4	5	7
2	2	1	2	3	1	4	4	6
3	2	4	4	7	6	5	8	7
4	1	3	5	4	5	5	4	5
5	3	3	6	7	5	7	8	9
6	1	2	5	6	4	4	5	6
7	3	5	6	8	6	7	7	8
8	0	1	1	2	0	4	7	8

Figure 6. G String V Software Step 7: (a) Data location and (b) Data view

Step 8: Specifying the sample sizes of object of measurement and facets

In Step 8, G String V asks for the sample size of the object of measurement and then the facets' sample sizes. For nested variables, specify the number of levels at each level of the nesting variable. For the object of measurement, this will be 8; for the testlet facet, this will be 3; and for the item facet, this will be the number of items per testlet; 2, 4, and 2, as seen in Figure 7.

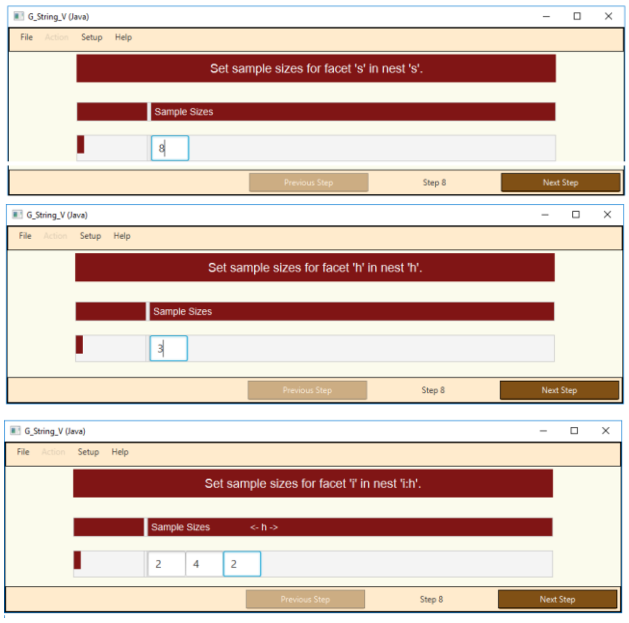
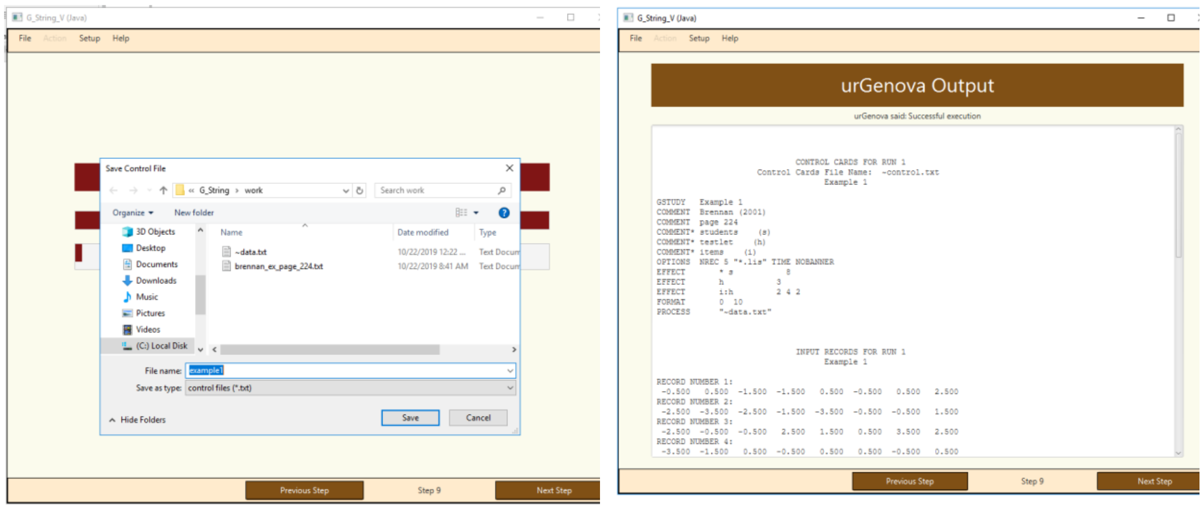


Figure 7. G String V Software Step 8

Step 9: Saving the Control File and obtaining variance components

After completing the specification by running the previous eight steps, a control file called "gControl.txt" was also generated. It was stored in the working directory by default. In Step 9, you can change both its name and folder as seen in Figure 8(a). After saving the proper control file path, urGENOVA is executed automatically to calculate the variance components and the coefficients (Bloch & Norman, 2018) as seen in Figure 8(b). Step 9 shows the variance components as part of a G study of G theory.



~data.txt 10/22/2019 12:22 ... Text Docum
brennan_ex_page_224.txt 10/22/2019 8:41 AM Text Docum

File name: example1
Save as type: control files (*.txt)

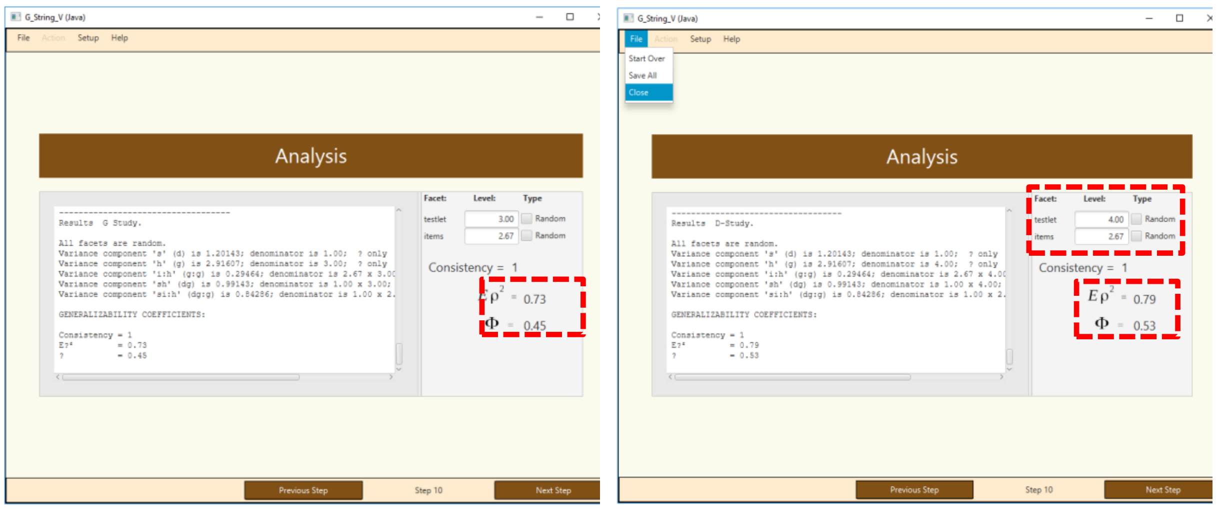
urGenova said: Successful execution

```
                    CONTROL CARDS FOR RUN 1
              Control Cards File Name:  ~control.txt
                          Example 1

GSTUDY   Example 1
COMMENT  Brennan (2001)
COMMENT  page 224
COMMENT* students    (s)
COMMENT* testlet    (h)
COMMENT* items    (i)
OPTIONS  NREC 5 "*.lis" TIME NOBANNER
EFFECT       * s          8
EFFECT       h           3
EFFECT       i:h         2 4 2
FORMAT       0  10
PROCESS      "~data.txt"



                    INPUT RECORDS FOR RUN 1
                          Example 1

RECORD NUMBER 1:
 -0.500   0.500  -1.500  -1.500   0.500  -0.500   0.500   2.500
RECORD NUMBER 2:
 -2.500  -3.500  -2.500  -1.500  -3.500  -0.500  -0.500   1.500
RECORD NUMBER 3:
 -2.500  -0.500  -0.500   2.500   1.500   0.500   3.500   2.500
RECORD NUMBER 4:
 -3.500  -1.500   0.500  -0.500   0.500   0.500  -0.500   0.500
```

Figure 8. G String V Software Step 9: (a) Saving control card and (b) urGENOVA output

Step 10: Calculating G Coefficients and Running D Studies

In Step 10, G String V calculates the G coefficients as seen in Figure 9(a). The first coefficient is called the G coefficient and is symbolized as Eρ^2, which is used for relative decisions. The second coefficient is called the index of dependability (Brennan & Kane, 1977) and symbolized as ϕ, which is used for absolute decisions. Furthermore, by changing levels and types of facets, you can calculate different coefficients to answer the question "What if…?" as part of a D study of the G theory. As seen in Figure 9(b), the level of testlet changed from 3 to 4, but the type of facets was left as random. After this change, the Eρ^2 and ϕ coefficients were also changed from .73 to .79 and .45 to .53, respectively. After completing all the intended D studies, by changing the levels of facets and clicking Next Step to obtain the results, close the software by clicking File→Close from the menu bar.



```
-----------------------------------
Results  G Study.

All facets are random.
Variance component 's' (d) is 1.20143; denominator is 1.00;  ? only
Variance component 'h' (g) is 2.91607; denominator is 3.00;  ? only
Variance component 'i:h' (g:g) is 0.29464; denominator is 2.67 x 3.00
Variance component 'sh' (dg) is 0.99143; denominator is 1.00 x 3.00;
Variance component 'si:h' (dg:g) is 0.84286; denominator is 1.00 x 2.

GENERALIZABILITY COEFFICIENTS:

Consistency = 1
E?²         = 0.73
?           = 0.45
```

testlet 3.00 Random; items 2.67 Random

Consistency = 1

$E\rho^2$ = 0.73

Φ = 0.45

```
-----------------------------------
Results  D-Study.

All facets are random.
Variance component 's' (d) is 1.20143; denominator is 1.00;  ? only
Variance component 'h' (g) is 2.91607; denominator is 4.00;  ? only
Variance component 'i:h' (g:g) is 0.29464; denominator is 2.67 x 4.00
Variance component 'sh' (dg) is 0.99143; denominator is 1.00 x 4.00;
Variance component 'si:h' (dg:g) is 0.84286; denominator is 1.00 x 2.

GENERALIZABILITY COEFFICIENTS:

Consistency = 1
E?²         = 0.79
?           = 0.53
```

testlet 4.00 Random; items 2.67 Random

Consistency = 1

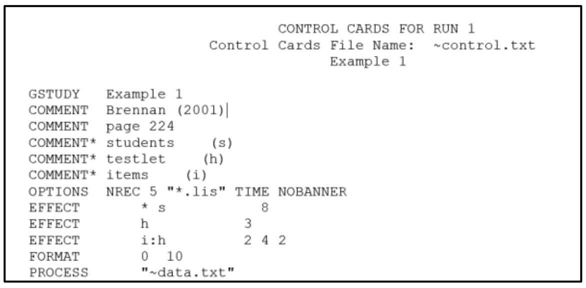
$E\rho^2$ = 0.79

Φ = 0.53

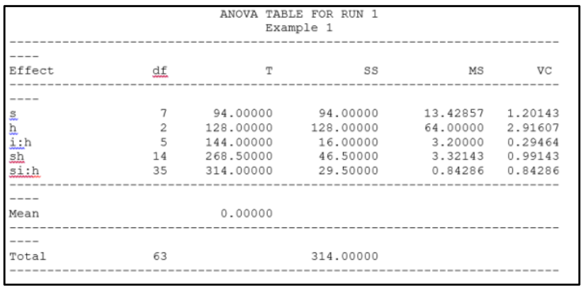
Figure 9. G String V Software Step 10: (a) G study results and (b) D study results

2.4. Evaluation of the Results

After closing the software, the output file will be in your working directory, named as "example1.txt.lis". This file can be opened by Word. As seen in Figure 10(a), there is a control card at the beginning of the output file. It contains the information entered in Step 1 (Example 1) and Step 2 (Brennan, 2001a, p.224). The names and levels of facets and the design of the study are also on the control card. Figure 10(b) shows the ANOVA table created by urGENOVA. The variance components used in the calculations of the $E\rho^2$ and ϕ coefficients are shown to the right of the table. It is possible to estimate variance components as negative because of erroneous measurement models or sampling errors (Güler, Kaya Uyanık & Taşdelen Teker, 2012). There are two different approaches to handle this situation. Cronbach et al., (1972) initially said that the negative variance should be replaced with zero, and that zero should be used to calculate other variance components. Brennan (2001a), however, argued that this suggestion could cause biased calculations of variance components. Cronbach responded by saying that although the negative variance should be replaced by zero, the negative value itself should be used to calculate other variance components (Atılgan, 2004). Negative variances are set to zero when computing coefficients by using G String V.



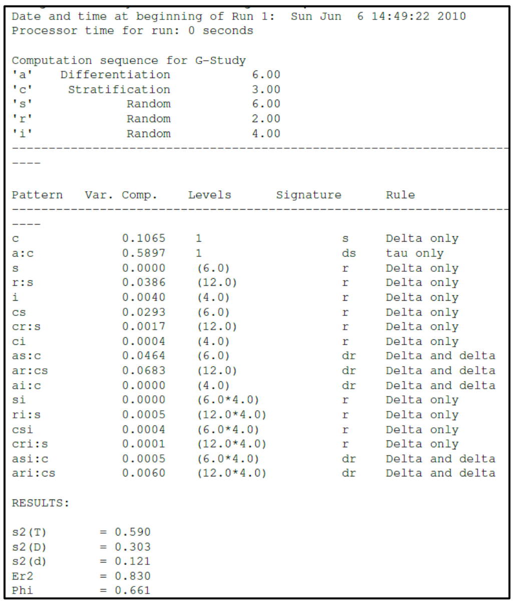
```
                        CONTROL CARDS FOR RUN 1
               Control Cards File Name:  ~control.txt
                             Example 1

GSTUDY   Example 1
COMMENT  Brennan (2001)|
COMMENT  page 224
COMMENT* students    (s)
COMMENT* testlet    (h)
COMMENT* items    (i)
OPTIONS  NREC 5 "*.lis" TIME NOBANNER
EFFECT       * s           8
EFFECT       h           3
EFFECT       i:h         2 4 2
FORMAT       0  10
PROCESS      "~data.txt"
```



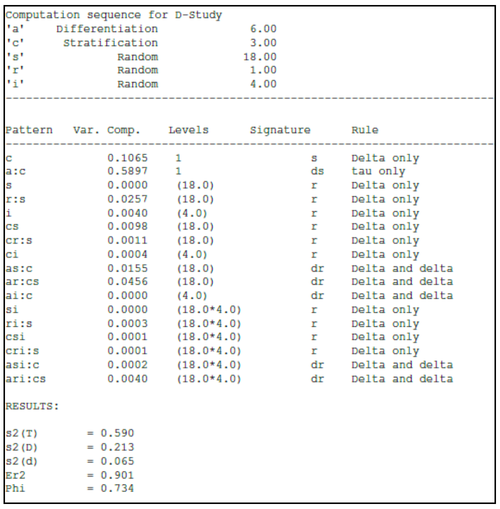
```
                        ANOVA TABLE FOR RUN 1
                             Example 1
--------------------------------------------------------------------------
----
Effect              df          T             SS              MS         VC
--------------------------------------------------------------------------
----
s                    7     94.00000       94.00000       13.42857   1.20143
h                    2    128.00000      128.00000       64.00000   2.91607
i:h                  5    144.00000       16.00000        3.20000   0.29464
sh                  14    268.50000       46.50000        3.32143   0.99143
si:h                35    314.00000       29.50000        0.84286   0.84286
--------------------------------------------------------------------------
----
Mean                        0.00000
--------------------------------------------------------------------------
----
Total               63                   314.00000
--------------------------------------------------------------------------
```

Figure 10. Output file: (a) Control card and (b) ANOVA Table

Calculations of $E\rho^2$and ϕ coefficients are shown in the output below the ANOVA table. In Figure 11(a), the estimated $E\rho^2$and ϕ coefficients are seen at the bottom of the figure and the results of the D studies are shown in Figure 11(b). When the level of testlet changed from 3 to 4, there is an increase in Eρ^2 and ϕ coefficients. More clearly, if the researcher increases the number of testlets, for instance to cover content area better, the results will be more reliable. Moreover, only the level of facets was changed and there was no change on the type of the facet from random to fixed. The reason of remaining the testlet facet as random was that since the entire universe of testlet levels was quite large and all levels were impossible to be included, the researcher was interested in generalizing beyond 3 or 4 testlets.



```
Date and time at beginning of Run 1:  Sun Jun  6 14:49:22 2010
Processor time for run: 0 seconds

Computation sequence for G-Study
'a'    Differentiation           6.00
'c'     Stratification           3.00
's'            Random            6.00
'r'            Random            2.00
'i'            Random            4.00
--------------------------------------------------------------------------
----

Pattern   Var. Comp.    Levels      Signature      Rule
--------------------------------------------------------------------------
----
c              0.1065    1                   s     Delta only
a:c            0.5897    1                   ds    tau only
s              0.0000    (6.0)               r     Delta only
r:s            0.0386    (12.0)              r     Delta only
i              0.0040    (4.0)               r     Delta only
cs             0.0293    (6.0)               r     Delta only
cr:s           0.0017    (12.0)              r     Delta only
ci             0.0004    (4.0)               r     Delta only
as:c           0.0464    (6.0)               dr    Delta and delta
ar:cs          0.0683    (12.0)              dr    Delta and delta
ai:c           0.0000    (4.0)               dr    Delta and delta
si             0.0000    (6.0*4.0)           r     Delta only
ri:s           0.0005    (12.0*4.0)          r     Delta only
csi            0.0004    (6.0*4.0)           r     Delta only
cri:s          0.0001    (12.0*4.0)          r     Delta only
asi:c          0.0005    (6.0*4.0)           dr    Delta and delta
ari:cs         0.0060    (12.0*4.0)          dr    Delta and delta

RESULTS:

s2(T)        = 0.590
s2(D)        = 0.303
s2(d)        = 0.121
Er2          = 0.830
Phi          = 0.661
```



```
Computation sequence for D-Study
'a'    Differentiation           6.00
'c'     Stratification           3.00
's'            Random           18.00
'r'            Random            1.00
'i'            Random            4.00
--------------------------------------------------------------------------

Pattern   Var. Comp.    Levels      Signature      Rule
--------------------------------------------------------------------------
c              0.1065    1                   s     Delta only
a:c            0.5897    1                   ds    tau only
s              0.0000    (18.0)              r     Delta only
r:s            0.0257    (18.0)              r     Delta only
i              0.0040    (4.0)               r     Delta only
cs             0.0098    (18.0)              r     Delta only
cr:s           0.0011    (18.0)              r     Delta only
ci             0.0004    (4.0)               r     Delta only
as:c           0.0155    (18.0)              dr    Delta and delta
ar:cs          0.0456    (18.0)              dr    Delta and delta
ai:c           0.0000    (4.0)               dr    Delta and delta
si             0.0000    (18.0*4.0)          r     Delta only
ri:s           0.0003    (18.0*4.0)          r     Delta only
csi            0.0001    (18.0*4.0)          r     Delta only
cri:s          0.0001    (18.0*4.0)          r     Delta only
asi:c          0.0002    (18.0*4.0)          dr    Delta and delta
ari:cs         0.0040    (18.0*4.0)          dr    Delta and delta

RESULTS:

s2(T)        = 0.590
s2(D)        = 0.213
s2(d)        = 0.065
Er2          = 0.901
Phi          = 0.734
```

Figure 11. Coefficient estimation based on (a) G study and (b) D Studies

3. LIMITATIONS OF G STRING V

The first limitation of G String V is related to the sample size of the data. The maximum number of the facet of differentiation, which is the object of measurement, is 1500. If your sample size is above this limit, it is stated on the G String V manual that you can write to the developers of the software and they can furnish a modified version of it. The other limitation of the software is related to the number of stratification facets of the study. For practical reasons, it cannot handle more than four stratification facets. According to the results of two review studies conducted by Rios, Li and Faulkner-Bond (2012) and Taşdelen Teker and Güler (2019), there is no study that has more than one stratification facet. Meanwhile, when a researcher has more than four stratification facets it has been suggested to collapse the facets that are unlikely to contribute to error variance.

Acknowledgements

The author wish to thank the Hacettepe Üniversitesi Teknokent Teknoloji Transfer Merkezi for proofreading process.

ORCID

Gülşen TAŞDELEN TEKER https://orcid.org/0000-0003-3434-4373

4. REFERENCES

Atılgan, H. (2004). *Genellenebilirlik kuramı ve cok değişkenlik kaynaklı Rasch modelinin karşılaştırılmasına ilişkin bir araştırma [A research on the comparison of the generalizability theory and many facet Rasch model]* (Doctoral Dissertation). Hacettepe University, Ankara.

Bloch, R. & Norman, G. (2018). *G String V User Manual*. Hamilton, Ontario, Canada.

Bloch, R. & Norman, G. (2015). *G String IV* (Version 6.1.1) User Manual. Hamilton, Ontario, Canada.

Bloch, R. & Norman, G. (2012). Generalizability theory for the perplexed: A practical introduction and guide: AMEE Guide No. 68. *Medical Teacher*, *34* (11), 960-992. DOI: 10.3109/0142159X.2012.703791

Brennan, R. L. (2001a). *Generalizability Theory*. New York: Springer.

Brennan, R. L. (2001b). *Manual for urGENOVA* (Version 2.1) (Iowa Testing Programs Occasional Paper Number 49). Iowa City, IA: Iowa Testing Programs, University of Iowa.

Brennan, R. L. (2000). Performance Assessments from the Perspective of Generalizability Theory. *Applied Psychological Measurement*, *24*(4), 339-353.

Brennan. R. L., & Kane, M. T. (1977). An index of dependability for mastery tests. *Journal of Educational Measurement*, *14*, 277-289.

Cardinet, J., Johnson, S. & Pini, G. (2010). *Applying Generalizability Theory using EduG*. New York, NY: Routledge – Taylor & Francis Group.

Cardinet, J., Tourneur, Y. & Allal, L. (1981). Extension of Generalizability Theory and Its Applications in Educational Measurement. *Journal of Educational Measurement*, *18* (4), 183-204.

Cardinet, J., Tourneur, Y. & Allal, L. (1976). The Symmetry of Generalizability Theory: Applications to Educational Measurement. *Journal of Educational Measurement*, *13* (2), 119-135.

Chiu, C. W. T. (2001). *Scoring performance assessments based on judgments: Generalizability theory*. Boston, MA: Kluwer Academic.

Cronbach, L. J., Gleser, G. C., Nanda, H. & Rajaratnam, N. (1972). *The Dependability of Behavioral Measurements: Theory of Generalizability for Scores and Profiles*. New York: Wiley.

Furr, R. M. (2011). *Scale construction and psychometrics for social and personality psychology*. Thousand Oaks, CA: Sage Publications Ltd.

Güler, N., Kaya Uyanık, G. & Taşdelen Teker, G. (2012). *Genellenebilirlik Kuramı [Generalizability Theory]*. Ankara: PegemA Yayıncılık.

Rios, J.A., Li, X., & Faulkner-Bond, M. (2012, October). *A review of methodological trends in generalizability theory*. Paper presented at the annual conference of the Northeastern Educational Research Association, Rocky Hill, CT.

Shavelson, J. R. & Webb, N. M. (2006). Generalizability theory. In: Green, J.L., Camill, G., Elmore, P.B., editors. *Handbook of complementary methods in education research*. Mahwah: Lawrence Erlbaum Associates Publishers, p. 309–322.

Shavelson, J. R. & Webb, N. M. (1991). *Generalizability Theory: A Primer*. Newbury Park. CA: Sage Publications.

Shavelson, R.J., Webb, N.M., & Rowley, G.L. (1989). Generalizability theory. *American Psychologist*, *44*(6), 922-932.

Shavelson, R. J., & Webb, N. M. (1981). Generalizability theory: 1973–1980. *British Journal of Mathematical and Statistical Psychology*, *34*, 133–166.

Suen, H. K. & Lei, P.W. (2007). Classical Versus Generalizability Theory of Measurement. *Educational Measurement*, 4, 1-13.

Taşdelen Teker, G. & Güler, N. (2019). Thematic Content Analysis of Studies Using Generalizability Theory. *International Journal of Assessment Tools in Education*, 6(2), 279–299. https://dx.doi.org/10.21449/ijate.569996

Webb, N. M., Shavelson, R. J. & Haertel, E. H. (2006). Reliability Coefficients and Generalizability Theory. *Handbook of Statistics*, *26*, 81-124. DOI: 10.1016/S0169-7161(06)26004