ISSN: 1309 – 6575
Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi
Journal of Measurement and Evaluation in Education and Psychology
2019; 10(3); 266-301

Building On-Demand Test Forms in R

Halil İbrahim SARI *

Abstract
Automated Test Assembly (ATA) plays important role in test development, especially in large scale test administrations. However, there is a lack of tutorials showing how to solve ATA problems. This tutorial aims to show to how build on-demand test forms easily for researchers and practitioners, and share the R codes for their use. The study presents the annotated R codes for thirty-nine unique examples. The examples include building one form, multiple forms and more complex ones under different constraint conditions across equal or different form lengths. All examples were solved by using "xxIRT" R package. The graphical depictions of the form-level information functions for all examples were also provided. Some important notes about the codes were also provided at the end of the paper in case one did not find a solution. The thirty-six examples were provided in the main body of the paper, the other three complex examples were given in the Supplementary material.

Key Words: On-demand test forms, automated test assembly, xxIRT, R.

INTRODUCTION

The ultimate goal of any test in the educational and psychological measurement is to estimate student's cognitive ability more accurately or precisely. However, it is quite difficult to reach this goal without a good measurement tool. This implies that the instrument or test form has to carry some certain psychometric characteristics to cover the construct of interest (e.g., math ability). It highlights the importance of the building the test forms that meet the desired features.

The topic of constructing the desired test forms is one of the most popular topics of all time. This is because regardless of the test administration type (fixed linear test, linear on the fly test, computerized multistage test or shadow test); for test security purposes, any test developer wants to build test forms that meet some certain requirements purposes, especially in large scale tests. Depending on the test administration type (e.g., linear or adaptive testing), one may want to build a single test form, two or more parallel forms that are at the same difficulty levels or multiple forms that are at the different difficulty levels. However, it is not very easy to ensure that the forms meet with both statistical (e.g., difficulty level) and non-statistical (e.g., content balancing and word count) specifications, especially when one wants to create many forms. Thus, instead of manually assembling forms, it is always better to use software to satisfy all constraints (e.g., test length, content balancing, difficulty level, word count etc.). This will help one to keep test form quality at the desired level.

Automated Test Assembly (ATA) is an integer programming approach used to solve equations that have complex constraints. In psychometrics, ATA is used to build test forms, and constraints refer to the desired test specifications. For instance, content balancing or distribution, difficulty level of form, number of test items in the form and total word count of items in a form can be thought as the constraint.

There are several integer programming software that are used to build test forms automatically. The most widely used ones are ILOG CPLEX ((International Business Machines-IBM, 2006), LINGO 12.0 (LINDO), CASTISEL (Luecht, 1998), LPSolve IDE (Berkelaar, Eikland, & Notebaert, 2004), the Premium Solver Platform 7.0 add-in for Microsoft Excel, and R packages "IpSolve" (Berkelaar et al., 2015) and "lpSolveAPI" (Konis, 2016). One can refer to Donoghue (2015) for a long list and detailed description.

* Assist. Prof., Kilis 7 Aralık University, Muallim Rıfat Faculty of Education, Kilis-Turkey, hisari87@gmail.com, ORCID ID: 0000-0001-7506-9000

To cite this article:
Sarı, H. İ. (2019). Building on-demand test forms in R. *Journal of Measurement and Evaluation in Education and Psychology, 10*(3), 266-301. doi: 10.21031/epod.521330

Received: 02.02.2019
Accepted: 15.07.2019

Furthermore, there are some studies that illustrate building on-demand tests or solving ATA problems. The book written by Wim J. van der Linden (2006), "Linear Models for Optimum Test Design", details all aspects of constructing both criterion-referenced and norm-referenced test forms. It is probably the most comprehensive book written in this area. Cor, Alves and Gierl (2008, 2009) and Gierl, Daniels and Zhang (2017), in this journal, showed how to create parallel forms in Microsoft Excel. They vividly demonstrate all steps, and provided helpful screenshots. Han and Rudner (2014) showed how to build multiple parallel items with different techniques. Diao and van der Linden (2011) described how to solve complex ATA problems by using lpSolve R package in version 5.5. They presented three different ATA problems and showed how to solve them in R. Unfortunately, they provided the code for one of the problem cases only.

Purpose of the Study

The purpose of this tutorial is to show how to create on-demand test forms under different constraints for the researchers and practitioners so that they can use the codes according to their own cases. There is an abundant literature on the automated test assembly, however, a lack of R code tutorials available for the researchers. I provided the annotated R codes for thirty-nine unique examples. Due to space limit, I provided thirty-six of them in the manuscript. One can refer to the Appendix A for other three complex examples. The examples include building one form, multiple forms and more complex forms across the different conditions. I also provided test information functions for all examples.

METHOD

Item Response Theory (IRT) is integral part of ATA because IRT is used to determine the difficulty level of a form and to shape form level information function (e.g., test information function). Item parameter estimates such as difficulty, discrimination and pseudo-guessing are first computed or generated; then the best items that meet certain criteria are selected for a test form. When selecting the best items, item information function is used. This is vital because item information function allows us to see where on the theta scale an item provides the highest information or for whom an item is the best. The information function for item i (Ii(θ)) under the three parameter IRT model (Birnbaum, 1968) for an item is defined as

$$I_i(\theta)=a_i^2 \frac{(P_i(\theta)-c_i)^2}{1-c_i^2} \frac{Q_i(\theta)}{P_i(\theta)} \quad (1)$$

where *a*, *b* and *c* are the discrimination, difficulty and pseudo-guessing parameters for item *i*, *P(θ)* and *Q(θ)* are the probability of getting an item correct and incorrect for a person having θ as the ability score, respectively.

As shown in Luecht (1998), the test assembly finds a solution to maximize the Item Response Theory information function at a fixed theta point (i.e., Equation 1). Let denote θ_0 is the fixed theta point (or theta interval), and suppose we want a total of 20-item in the test. We first define a binary decision variable, x_i, (e.g., $x_i = 0$ means item *i* is not selected from the item bank, $x_i = 1$ means item *i* is selected from the item bank). The information function needs to be maximized;

$$I(\theta)=\sum_{i=1}^{N} I(\theta_0\ \xi i)x_i \quad (2)$$

where ξ_i represents the item parameters of item *i* (e.g., a, b, c parameters). Let's say one has two content areas (e.g., items measuring number properties and items measuring algebra denoted as C1 and C2, respectively), and wants to select ten items from each content area. The automated test assembly is modeled to maximize

$$I(\theta)=\sum_{i=1}^{N} I(\theta_0\ \xi i)x_i \quad (3)$$

subject to

$$\sum_{i\in C1}^{N} x_i \geq 10 \quad (4)$$

$$\sum_{i\in C2}^{N} x_i \geq 10 \quad (5)$$

$$\sum_{i=1}^{N} x_i \geq 20 \quad (6)$$

$$x_i \in (0,1),\ i=1,\ldots N \quad (7)$$

which put constraints on C1, C2, the total test length, and the range of decision variables, respectively. When the content balancing is not controlled, the constraints on the contents (Equations 4 and 5) can be removed from the model. When one wants to control other variables, he or she can add the additional constraints to the model.

It is important to note that the information function in Equation 1 can be maximized at a desired theta point (e.g., -1, 0 and 1 for easy, medium and hard test forms, respectively). Moreover, it can be maximized over a range of theta interval (e.g., -1 to 0 for an easy test form, and 0 to 1 for a hard test form). It is also possible to demand a user defined absolute amount of information either at a fixed theta point or over a range of theta interval (e.g., the amount of information is 8 at the theta point of 0 or at the interval of -1 to 1).

xxIRT PACKAGE

In this tutorial, I used "xxIRT" R package version 2.1.0 (Luo, 2018) to solve all given examples. The "xxIRT" R package is a recently released package, and uses "lpSolveAPI" package as the integer solver. The version of the package on CRAN has been recently updated. There are some core functions needed to be used when specifying and solving ATA problems. The most important function is *ata* which is used to create ATA problems. One needs to specify the information about item pool (either simulated or real), the number of test forms needs to be created, the length of form (e.g., 5 items), and maximum use of an item in the pool. The other two core functions are *ata_obj_relative* and *ata_obj_absolute*. The first is used when one wants to maximize the information at a fixed theta point or over a range of theta interval. When the theta interval is desired, the increments of theta points in the user defined theta interval can be specified. The latter is used when one wants to have an absolute amount of information for a test form. Similarly, when the target is to gather absolute information over a range of interval, the increment points can be specified by the user. The increment points are important because they may dramatically change the shape of test information function.

As discussed before, constraints are important elements of an ATA problem. The *ata_constraint* function adds the constraints to the ATA model. One can specify the number of items from each content area or total word count for a test form. Finally, *ata_solve* function solves the specified ATA problem. It is also possible to see the selected items and plot test information functions. One can refer to "xxIRT" package for more information about the codes and main functions. The annotated R codes different examples were given below.

The "xxIRT" package was primarily created to solve ATA problems for multistage testing (e.g., designing panels) so, there is no extended illustration of how to create simple or complex and single or multiple test forms. The current manual shows four simple examples but this provides shows many complex examples by using the same main functions.

Annonated Examples

I used a simulated item pool that consists of 1000 items and generated three hypothetical constraints as content area (e.g., algebra, numbers, equations), word count of each item (e.g., ranging from 30 to 150 for each item) and time required to solve the item (e.g., ranging from 100 seconds to 400 seconds for each item). I presented 12 ATA problems and for each problem, I showed how to solve the ATA by a) maximizing information function at a fixed theta point, b) maximizing information over a theta interval, and c) getting absolute amount of information for a form. For all 36 examples listed below, I always used the same simulated item pool. The number of items, distribution of item parameters, and hypothetical constraints are subject to change.

Preparing for the analysis

```
# Do not run!
#Install the "xxIRT" package first
install.packages("xxIRT",repos = "http://cran.us.r-project.org")
require("xxIRT")

# Let's generate an item pool
set.seed(10)
items=as.data.frame(cbind(
a=runif(1000, 0.5, 1.5), #a parameters from a uniform distribution. #Chan
ge accordingly!
b=runif(1000, -2, 2), #b parameters from a uniform distribution. Change #
accordingly!
c=runif(1000, 0, 0.20), #c parameters from a uniform distribution. Change
#accordingly!
content=sample(1:3,1000,replace = T), #3 content areas #(e.g., algebra, #
numbers, equations)
word_count=sample(30:150,1000,replace = T), #assigning random word counts
#for each item.
time=sample(100:400,1000,replace = T))) #assigning random time between #f
or each item.
#End
```

Problem 1: Building single forms without any constraint

Here, I created one single test form that does not have any constraints. There are three problems listed as Problem 1a, 1b and 1c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively.

```
#Problem1a: maximize the information at fixed theta point of -1
Problem1a <- ata(items, 1, #must be 1 when single form is built!
                 len=10, #test length. Change accordingly!
                 max_use=1) #Each item should be selected one time only!
Problem1a <- ata_obj_relative(Problem1a,
                              -1, # fixed theta point where we want to #
maximize the information.
                              "max")
Problem1a <- ata_solve(Problem1a, as.list=T) #Now we are ready to solve #
the ATA
plot(Problem1a) #plotting information function
#End

#Problem1b: Maximize the information at the theta interval of -1 to 1.
Problem1b <- ata(items, 1, len=10, max_use=1)
Problem1b <-  ata_obj_relative(Problem1b, seq(-1, 1, #change when a #diff
erent interval is desired
                                              0.10),  #increment of the #
theta points.
                              "max", flatten=0.10) #change accordingly!
Problem1b <- ata_solve(Problem1b, as.list=T) #Now we are ready to solve #
the ATA
plot(Problem1b) # plotting information function
```

```
# End

# Problem1c: absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0) # target theta points (from -1 #
to 1)
tif_target= 8 # desired amount of information at the test level. Change #
accordingly!
Problem1c <- ata(items, 1, len=10, max_use=1)
Problem1c <- ata_obj_absolute(Problem1c, theta_target, tif_target) #ATA #
problem
Problem1c <- ata_solve(Problem1c, as.list=T) #Now we are ready to solve #
the ATA.
plot(Problem1c)  # plotting information function
# End
```

Problem 2: Building single form with content constraint only

Here, I created one single test form that has content constraints. There are three problems listed as Problem 2a, 2b and 2c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 2 examples, I pulled 10 items as 2, 3, and 5 items from the content1, content2 and content3, respectively.

```
# Problem2a: maximize the information at fixed theta point of -1.
Problem2a <- ata(items, 1, #single test form. This must be 1 when single
#form is created!
                 len=10, #test length of 10. Change accordingly!
                 max_use=1) #Each item should be selected one time only!
Problem2a <-  ata_obj_relative(Problem2a, -1, "max") #specifying ATA #pro
blem
#Now let's add content distribution constraints before solving the ATA.
Problem2a <-  ata_constraint(Problem2a, "content", min=2, max=2, level=1)
# 2 items from Content 1
Problem2a <-  ata_constraint(Problem2a,"content", min=3, max=3, level=2)
# 3 items from Content 2
Problem2a <-  ata_constraint(Problem2a, "content", min=5, max=5, level=3)
# 5 items from Content 3
Problem2a <- ata_solve(Problem2a, as.list=T) #Now, ATA is ready to solve!
plot(Problem2a) # plotting information function
# End

# Problem2b: maximize the information at theta interval of -1 to 1.
Problem2b <- ata(items, 1, len=10, max_use=1) # Test length of 10. Change
#accordingly!
Problem2b <-  ata_obj_relative(Problem2b, seq(-1, 1, 0.10), "max", flatte
n=0.10)
#Now let's add content constraints before solve the ATA.
Problem2b <- ata_constraint(Problem2b, "content", min=2, max=2, level=1)
# 2 items from C 1
Problem2b <- ata_constraint(Problem2b,"content", min=3, max=3, level=2)
# 3 items from C 2
Problem2b <- ata_constraint(Problem2b, "content", min=5, max=5, level=3)
# 5 items from C 3
Problem2b <- ata_solve(Problem2b, as.list=T) #Now, ATA is ready to solve!
```

```
plot(Problem2b) # plotting information function
# End

#Problem2c: absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0) # target theta points where you
#want to maximize information.
tif_target= 8 # desired amount of information we want. Change #accordingl
y!
Problem2c <- ata(items, 1, len=10, max_use=1) # Test length of 10. Change
#accordingly!
Problem2c <- ata_obj_absolute(Problem2c, theta_target, tif_target) #speci
fying ATA problem
#Now let's add content constraints before solve the ATA.
Problem2c <-  ata_constraint(Problem2c, "content", min=2, max=2, level=1)
# 2 items from C 1
Problem2c <-  ata_constraint(Problem2c,"content", min=3, max=3, level=2)
# 3 items from C 2
Problem2c <-  ata_constraint(Problem2c, "content", min=5, max=5, level=3)
# 5 items from C 3
Problem2c <- ata_solve(Problem2c, as.list=T) #Now, ATA is ready to solve!
Problem2c$items #see selected items
plot(Problem2c) # plotting information function
# End
```

Problem 3: Building single form with two constraints

Here, I created one single test form that has content and word count constraints. There are three problems listed as Problem 3a, 3b and 3c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 3 examples, I pulled 10 items as 2, 3, and 5 items from the content1, content2 and content3, respectively. The average word count of the items in the forms is between 60 and 70.

```
#Problem3a: maximize the information at the fixed theta point of -1.
Problem3a <- ata(items, 1, # Building single form. Change accordingly!
                 len=10, #Test length of 10. Change accordingly!
                 max_use=1) # Each item should be selected one time only!
Problem3a <- ata_obj_relative(Problem3a, -1, "max") #specifying ATA #prob
lem
#Now let's add content constraints before solve the ATA.  Change #accordi
ngly!
Problem3a <- ata_constraint(Problem3a, "content", min=2, max=2, level=1)
# 2 items from C 1
Problem3a <- ata_constraint(Problem3a,"content", min=3, max=3, level=2) #
3 items from C 2
Problem3a <- ata_constraint(Problem3a, "content", min=5, max=5, level=3)
# 5 items from C 3
#Now let's add word count constraint before solve the ATA. Change #accord
ingly!
Problem3a <-  ata_constraint(Problem3a, "word_count", min=60*10, max=70*1
0)
Problem3a <- ata_solve(Problem3a, as.list=T) #Now, ATA is ready to solve!
Problem3a$items #see selected items
plot(Problem3a) # plotting information function
```

```
# End

# Problem3b:  maximize the information at theta interval of -1 to 1.
Problem3b <- ata(items, 1, len=10, max_use=1)
Problem3b <-  ata_obj_relative(Problem3b, seq(-1, 1, 0.10),  "max", flatt
en=0.10)
#Now let's add content constraints before solve the ATA.  Change #accordi
ngly!
Problem3b <-  ata_constraint(Problem3b, "content", min=2, max=2, level=1)
# 2 items from Content 1
Problem3b <-  ata_constraint(Problem3b,"content", min=3, max=3, level=2)
# 3 items from Content 2
Problem3b <-  ata_constraint(Problem3b, "content", min=5, max=5, level=3)
# 5 items from Content 3
#Now let's add word count constraint before solve the ATA.  Change #accor
dingly!
Problem3b <-  ata_constraint(Problem3b, "word_count", min=60*10, max=70*1
0)
Problem3b <- ata_solve(Problem3b, as.list=T) #Now, ATA is ready to solve!
Problem3b$items #see selected items
plot(Problem3b) # plotting information function
# End

# Problem3c: absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0) # target theta points where you
#want to maximize
#information. One can also specify fixed theta point! Change accordingly!
tif_target= 8 # desired amount of information. Change accordingly!
Problem3c <- ata(items, 1, len=10, max_use=1)
Problem3c <- ata_obj_absolute(Problem3c, theta_target, tif_target) #speci
fying ATA problem
#Now let's add content constraints before solve the ATA.  Change #accordi
ngly!
Problem3c <-  ata_constraint(Problem3c, "content", min=2, max=2, level=1)
# 2 items from C 1
Problem3c <-  ata_constraint(Problem3c,"content", min=3, max=3, level=2)
# 3 items from C 2
Problem3c <-  ata_constraint(Problem3c, "content", min=5, max=5, level=3)
# 5 items from C 3
#Now let's add word count constraint before solve the ATA.  Change #accor
dingly!
Problem3c <-  ata_constraint(Problem3c, "word_count", min=60*10, max=70*1
0)
Problem3c <- ata_solve(Problem3c, as.list=T) #Now, ATA is ready to solve!
plot(Problem3c) # plotting information function
# End
```

Problem 4: Building single form with three constraints

Here, I created one single test form that has content, word count and time constraints. There are three problems listed as Problem 4a, 4b and 4c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 4 examples,

I pulled 10 items as 2, 3, and 5 items from the content1, content2 and content3, respectively. The average word count of the items in the forms is between 60 and 70. The average time to solve an item is between 200 and 300 seconds.

```
#Problem4a: maximize the information at the fixed theta point of -1.
Problem4a <- ata(items, 1, # Building single form. Change accordingly!
                 len=10, #Test length of 10. Change accordingly!
                 max_use=1) # Each item should be selected one time only!
Problem4a <-  ata_obj_relative(Problem4a, -1, "max")
#Now let's add content constraints before solve the ATA.  Change #accordi
ngly!
Problem4a <-  ata_constraint(Problem4a, "content", min=2, max=2, level=1)
# 2 items from C 1
Problem4a <-  ata_constraint(Problem4a,"content", min=3, max=3, level=2)
# 3 items from C 2
Problem4a <-  ata_constraint(Problem4a, "content", min=5, max=5, level=3)
# 5 items from C 3
#Now let's add word count constraint before solve the ATA.  Change #accor
dingly!
Problem4a <-  ata_constraint(Problem4a, "word_count", min=60*10, max=70*1
0)
#Now let's add time constraint before solve the ATA.  Change accordingly!
Problem4a <-  ata_constraint(Problem4a, "time", min=200*10, max=300*10)
Problem4a <- ata_solve(Problem4a, as.list=T) #Now, ATA is ready to solve!
Problem4a$items #see selected items
plot(Problem4a) # plotting information function
# End

# Problem4b: maximize the information at theta interval of -1 to 1.
Problem4b <- ata(items, 1, len=10, max_use=1) #A total of 10 items. #Chan
ge accordingly!
Problem4b <-  ata_obj_relative(Problem4b, seq(-1, 1, 0.10), "max", flatte
n=0.10)
#Now let's add content constraints before solve the ATA.  Change #accordi
ngly!
Problem4b <-  ata_constraint(Problem4b, "content", min=2, max=2, level=1)
# 2 items from C 1
Problem4b <-  ata_constraint(Problem4b,"content", min=3, max=3, level=2)
# 3 items from C 2
Problem4b <-  ata_constraint(Problem4b, "content", min=5, max=5, level=3)
# 5 items from C 3
#Now let's add word count constraints before solve the ATA.  Change #acco
rdingly!
Problem4b <-  ata_constraint(Problem4b, "word_count", min=60*10, max=70*1
0)
#Now let's add time constraint before solve the ATA.  Change accordingly!
Problem4b <-  ata_constraint(Problem4b, "time", min=200*10, max=300*10)
Problem4b <- ata_solve(Problem4b, as.list=T) #Now, ATA is ready to solve!
Problem4b$items #see selected items
plot(Problem4b) # plotting information function
# End

# Problem4c:  absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0) # target theta points where you
```

```
#to maximize #information. One can also specify fixed theta point! Change
#accordingly!
tif_target= 8 # desired amount of information. Change accordingly!
Problem4c <- ata(items, 1, len=10, max_use=1) #A total of 10 items. #Chan
ge accordingly!
Problem4c <- ata_obj_absolute(Problem4c, theta_target, tif_target) #speci
fying ATA problem
#Now let's add content constraints before solve the ATA.  Change #accordi
ngly!
Problem4c <-  ata_constraint(Problem4c, "content", min=2, max=2, level=1)
#2 items from C 1
Problem4c <-  ata_constraint(Problem4c,"content", min=3, max=3, level=2)
#3 items from C 2
Problem4c <-  ata_constraint(Problem4c, "content", min=5, max=5, level=3)
#5 items from C 3
#Now let's add word count constraints before solve the ATA.  Change #acco
rdingly!
Problem4c <-  ata_constraint(Problem4c, "word_count", min=60*10, max=70*1
0)
#Now let's add time constraints before solve the ATA.  Change #accordingl
y!
Problem4c <-  ata_constraint(Problem4c, "time", min=200*10, max=300*10)
Problem4c <- ata_solve(Problem4c, as.list=T) #Now, ATA is ready to solve!
Problem4c$items #see selected items
plot(Problem4c) # plotting information function
#End
```

Problem 5: Building two equal-length forms with no constraints

Here, I created two test forms that have any constraints. There are three problems listed as Problem 5a, 5b and 5c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 5 examples, I pulled equal test lengths (10 items) and content was not controlled.

```
# Problem5a: maximize the information at the fixed theta point of 0 for #
all forms
Problem5a <- ata(items, 2, #Building 2 forms. Change accordingly!
                 len=10, # Test length of 10. Change accordingly!
                 max_use=1) #We don't want item overlapping. Change #acco
rdingly!
Problem5a <-  ata_obj_relative(Problem5a, 0, # change when a different #f
ixed theta point is desired
                               "max")
Problem5a <- ata_solve(Problem5a, as.list=T) # "as.list=F" gives all #sel
ected items together
Problem5a$items #see selected items
plot(Problem5a) # plotting information function
#End

# Problem5b: maximize the information at theta interval of -1 to 1.
Problem5b <- ata(items, 2, len=10, max_use=1)
Problem5b <-  ata_obj_relative(Problem5b, seq(-1, 1, 0.50), # change #acc
ordingly!
```

```
                                "max", flatten=0.10) # change accordingly!
Problem5b <- ata_solve(Problem5b,as.list=T) #Now let's solve the ATA!
Problem5b$items #see selected items
plot(Problem5b) # plotting information function
#End

# Problem5c: absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0) # target theta points. Change #a
ccordingly!
tif_target= 8 # desired amount of information. Change accordingly!
Problem5c <- ata(items, 2, len=10, max_use=1)
Problem5c <- ata_obj_absolute(Problem5c, theta_target, tif_target) # #Spe
cifying the ATA.
Problem5c <- ata_solve(Problem5c, as.list=T) # Let's solve the ATA #probl
em.
Problem5c$items #see selected items
plot(Problem5c) # plotting information function
# End
```

Problem 6: Building unequal-length two forms with no constraints

Here, I created two test forms that have any constraints same in problem 5a, 5b and 5c. However, in problem 6 examples, the test lengths are not equal. There are three problems listed as Problem 6a, 6b and 6c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 6 examples, I pulled 5 items for form 1 and 8 items for form 2 but content distribution was not controlled for both forms.

```
# Problem6a:  maximize the information at the fixed theta point of 0 for
#all forms
Problem6a <- ata(items, 2, # don't specify form length "len=10" because #
of unequal test lengths
                 max_use=1) #we do not want overlapping items.
Problem6a <-  ata_obj_relative(Problem6a, 0, #fixed theta point for both
#forms. Change accordingly!
                                "max")
Problem6a <- ata_constraint(Problem6a,1, min=5, max=5, forms=1) #5 items
#in form 1
Problem6a <- ata_constraint(Problem6a,1,  min=8, max=8, forms=2) #8 items
#in form 2
Problem6a <- ata_solve(Problem6a, as.list=T) # Let's solve the ATA #probl
em.
Problem6a$items #see selected items
plot(Problem6a) # plotting information function
#End

# Problem6b: maximize the information at theta interval of -1 to 1.
Problem6b <- ata(items, 2,  # two forms
                 max_use=1) #we do not want overlapping items. Change #ac
cordingly!
Problem6b <-  ata_obj_relative(Problem6b, seq(-1, 1, 0.50),  #Change the
interval accordingly!
                                "max", flatten=0.10)
Problem6b <- ata_constraint(Problem6b,1, min=5, max=5, forms=1) #5 items
```

```
#in form 1
Problem6b <- ata_constraint(Problem6b,1,  min=8, max=8, forms=2) #8 items
#in form 2
Problem6b <- ata_solve(Problem6b,as.list=T) #Now Let's solve the ATA
Problem6b$items #see selected items
plot(Problem6b) # plotting information function
#End

# Problem6c: absolute information target
theta_target=0 #the theta point where we want to maximize the #informatio
n. Change accordingly!
tif_target= 5 # desired amount of information. Change accordingly!
Problem6c <- ata(items, 2, max_use=1)
Problem6c <- ata_constraint(Problem6c,1, min=5, max=5, forms=1) #5 items
#in form 1
Problem6c <- ata_constraint(Problem6c,1,  min=6, max=6, forms=2) #8 items
#in form 2
Problem6c <- ata_obj_absolute(Problem6c, theta_target, tif_target, forms
= 1) # ATA for form 1
Problem6c <- ata_obj_absolute(Problem6c, theta_target, tif_target, forms
= 2) # ATA for form 2
Problem6c <- ata_solve(Problem6c, as.list=T) #Now Let's solve the ATA
plot(Problem6c) # plotting information function
#End
```

Problem 7: Building equal-length two forms with content constraint

Here, I created two test forms with controlling content distribution. There are three problems listed as Problem 7a, 7b and 7c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all problem 7 examples, for both forms, the test length is 10, and I pulled 2, 3 and 5 items content2 and content3, respectively.

```
# Problem7a: Maximize the information at the fixed theta point of 0 for #
all forms.
Problem7a <- ata(items, 2, len=10, #Equal test length of 0 for both #form
s. Change accordingly!
                 max_use=1) #I want non-overlapping forms.
Problem7a <- ata_obj_relative(Problem7a, 0, # fixed theta point. Change #
accordingly!
                              "max")
Problem7a <- ata_constraint(Problem7a, "content", min=2, max=2, level=1)
#2 items from C 1
Problem7a <- ata_constraint(Problem7a,"content", min=3, max=3, level=2) #
3 items from C 2
Problem7a <- ata_constraint(Problem7a, "content", min=5, max=5, level=3)
#5 items from C 3
Problem7a <- ata_solve(Problem7a, as.list=T) #Now Let's solve the ATA
plot(Problem7a) # plotting information function
#End

# Problem7b: Maximize the information at theta interval of -1 to 1.
Problem7b <- ata(items, 2, len=10, max_use=1)
Problem7b <-  ata_obj_relative(Problem7b, seq(-1, 1, 0.50), #Change the #
```

```
interval accordingly!
                                "max", flatten=0.50)
Problem7b <- ata_constraint(Problem7b, "content", min=2, max=2, level=1)
#2 items from content 1
Problem7b <- ata_constraint(Problem7b,"content", min=3, max=3, level=2) #
3 items from content 2
Problem7b <- ata_constraint(Problem7b, "content", min=5, max=5, level=3)
#5 items from content 3
Problem7b <- ata_solve(Problem7b,as.list=T) #Now Let's solve the ATA
Problem7b$items #see selected items
plot(Problem7b) # plotting information function
#End

# Problem7c: absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0)
tif_target= 8 # desired amount of information
Problem7c <- ata(items, 2, len=10, max_use=1)
Problem7c <- ata_constraint(Problem7c, "content", min=2, max=2, level=1)
#2 items from C 1
Problem7c <- ata_constraint(Problem7c,"content", min=3, max=3, level=2) #
3 items from C 2
Problem7c <- ata_constraint(Problem7c, "content", min=5, max=5, level=3)
#5 items from C 3
Problem7c <- ata_obj_absolute(Problem7c, theta_target, tif_target) # ATA
#for both forms
Problem7c <- ata_solve(Problem7c, as.list=T) #Now Let's solve the ATA
Problem7c$items #see selected items
plot(Problem7c) # plotting information function
#End
```

Problem 8: Building unequal-length two forms with content constraint

Here, I created two unequal-length test forms with controlling content distribution. There are three problems listed as Problem 8a, 8b and 8c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 8 examples in below, for form 1, I pulled 5 items as 1, 2 and 2 from content 1, 2 and 3, respectively. For form 2, I pulled 8 items as 2, 3 and 3 from content 1, 2 and 3, respectively.

```
# Problem8a: Maximize the information at the fixed theta point
Problem8a <- ata(items, 2, #two test forms
                 max_use=1) #I don't want item overlapping across the #fo
rms
Problem8a <-  ata_obj_relative(Problem8a, 0, "max")
#Now let's specify total test lengths for both forms.
Problem8a <- ata_constraint(Problem8a,1, min=5, max=5, forms=1) #5 items
#in form 1
Problem8a <- ata_constraint(Problem8a,1,  min=8, max=8, forms=2) #8 items
#in form 2
#Now let's add content constraints for form 1.  Change accordingly!
Problem8a <- ata_constraint(Problem8a, "content", min=1, max=1, level=1,f
orms = 1)
Problem8a <- ata_constraint(Problem8a,"content", min=2, max=2, level=2, f
orms = 1)
```

```
Problem8a <- ata_constraint(Problem8a, "content", min=2, max=2, level=3,f
orms = 1)
#Now let's add content constraints for form 2.  Change accordingly!
Problem8a <- ata_constraint(Problem8a, "content", min=2, max=2, level=1,f
orms = 2)
Problem8a <- ata_constraint(Problem8a,"content", min=3, max=3, level=2, f
orms = 2)
Problem8a <- ata_constraint(Problem8a, "content", min=3, max=3, level=3,f
orms = 2)
Problem8a <- ata_solve(Problem8a, as.list=T) # ATA is ready to solve
Problem8a$items #see selected items
plot(Problem8a) # plotting information function
#End

# Problem8b: maximize the information at theta interval of -1 to 1.
Problem8b <- ata(items, 2, max_use=1)
Problem8b <-  ata_obj_relative(Problem8b, seq(-1, 1, 0.50), #theta #inter
val. Change accordingly!
                               "max", flatten=0.10)
Problem8b <- ata_constraint(Problem8b,1, min=5, max=5, forms=1) #5 items
#in form 1
Problem8b <- ata_constraint(Problem8b,1,  min=8, max=8, forms=2) #8 items
#in form 2
#Now let's add content constraints for form 1.  Change accordingly!
Problem8b <- ata_constraint(Problem8b, "content", min=1, max=1, level=1,f
orms = 1)
Problem8b <- ata_constraint(Problem8b,"content", min=2, max=2, level=2, f
orms = 1)
Problem8b <- ata_constraint(Problem8b, "content", min=2, max=2, level=3,f
orms = 1)
#Now let's add content constraints for form 2.  Change accordingly!
Problem8b <- ata_constraint(Problem8b, "content", min=2, max=2, level=1,f
orms = 2)
Problem8b <- ata_constraint(Problem8b,"content", min=3, max=3, level=2, f
orms = 2)
Problem8b <- ata_constraint(Problem8b, "content", min=3, max=3, level=3,f
orms = 2)
Problem8b <- ata_solve(Problem8b,as.list=T) # ATA is ready to solve
Problem8b$items #see selected items
plot(Problem8b) # plotting information function
#End

# Problem8c:  absolute information target
theta_target=c(-1.0, -0.5 ,0, 0.5,  1.0) #theta interval I want to #maxim
ize the information
tif_target= 5 # desired amount of information. Change accordingly!
Problem8c <- ata(items, 2, max_use=1)
Problem8c <- ata_constraint(Problem8c,1, min=5, max=5, forms=1) #5 items
#in form 1
Problem8c <- ata_constraint(Problem8c,1,  min=8, max=8, forms=2) #8 items
#in form 2
#Now let's add content constraints for form 1.  Change accordingly!
Problem8c <- ata_constraint(Problem8c, "content", min=1, max=1, level=1,f
```

```
orms = 1)
Problem8c <- ata_constraint(Problem8c,"content", min=2, max=2, level=2, f
orms = 1)
Problem8c <- ata_constraint(Problem8c, "content", min=2, max=2, level=3,f
orms = 1)
#Now let's add content constraints for form 2.  Change accordingly!
Problem8c <- ata_constraint(Problem8c, "content", min=2, max=2, level=1,f
orms = 2)
Problem8c <- ata_constraint(Problem8c,"content", min=3, max=3, level=2, f
orms = 2)
Problem8c <- ata_constraint(Problem8c, "content", min=3, max=3, level=3,f
orms = 2)
Problem8c <- ata_obj_absolute(Problem8c, theta_target, tif_target, forms
= 1) #ATA for form 1
Problem8c <- ata_obj_absolute(Problem8c, theta_target, tif_target, forms
= 2) #ATA for form 2
Problem8c <- ata_solve(Problem8c, as.list=T) #Now ATA is ready to solve
Problem8c$items #see selected items
plot(Problem8c) # plotting information function
#End
```

Problem 9: Building equal-length two forms with two constraints

Here, I create two equal-length test forms with controlling content distribution and word count. There are three problems listed as Problem 9a, 9b and 9c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 9 examples, I pulled 10 items for both forms, and in both forms, there are 2, 3, and 5 items from the content1, content2 and content3, respectively. The average word count of the items in both forms is between 60 and 70.

```
# Problem9a: maximize the information at the fixed theta point of 0 for #
both forms.
Problem9a <- ata(items, 2, # number  of forms
        len=10, # Equal test length of 10 for both forms. Change #accordi
ngly!
max_use=1) #I don't want item overlapping across the forms.
Problem9a <- ata_obj_relative(Problem9a, 0, # fixed theta point. Change #
accordingly!
                        "max")
#Now let's add content constraints for the forms. Change accordingly!
Problem9a <- ata_constraint(Problem9a, "content", min=2, max=2, level=1)
#2 items from content 1
Problem9a <- ata_constraint(Problem9a,"content", min=3, max=3, level=2) #
3 items from content 2
Problem9a <- ata_constraint(Problem9a, "content", min=5, max=5, level=3)
#5 items from content 3
#Now let's add word count constraints before solve the ATA.  Change #acco
rdingly!
Problem9a <-  ata_constraint(Problem9a, "word_count", min=60*10, max=70*1
0)
Problem9a <- ata_solve(Problem9a, as.list=T) #Let's solve the ATA
Problem9a$items #see selected items
plot(Problem9a) # plotting information function
```

```
#End

# Problem9b: maximize the information at theta interval of -1 to 1.
Problem9b <- ata(items, 2, len=10, max_use=1)
Problem9b <-  ata_obj_relative(Problem9b, seq(-1, 1, 0.50), #theta #inter
val. Change accordingly!
                           "max", flatten=0.50)
#Now let's add content constraints for the forms. Change accordingly!
Problem9b <- ata_constraint(Problem9b, "content", min=2, max=2, level=1)
#2 items from content 1
Problem9b <- ata_constraint(Problem9b,"content", min=3, max=3, level=2) #
3 items from content 2
Problem9b <- ata_constraint(Problem9b, "content", min=5, max=5, level=3)
#5 items from content 3
#Now let's add word count constraints before solve the ATA.  Change #acco
rdingly!
Problem9b <-  ata_constraint(Problem9b, "word_count", min=60*10, max=70*1
0)
Problem9b <- ata_solve(Problem9b,as.list=T) #Let's solve the ATA
Problem9b$items #see selected items
plot(Problem9b) # plotting information function
#End

# Problem9c: Absolute information target.
theta_target=0 #the point where we want the absolute information. One can
#specify interval as well!
tif_target= 8 # desired amount of information. Change accordingly!
Problem9c <- ata(items, 2, len=10, max_use=1)
#Now let's add content constraints for the forms. Change accordingly!
Problem9c <- ata_constraint(Problem9c, "content", min=2, max=2, level=1)
#2 items from content 1
Problem9c <- ata_constraint(Problem9c,"content", min=3, max=3, level=2) #
3 items from content 2
Problem9c <- ata_constraint(Problem9c, "content", min=5, max=5, level=3)
#5 items from content 3
#Now let's add word count constraints before solve the ATA.  Change #acco
rdingly!
Problem9c <-  ata_constraint(Problem9c, "word_count", min=60*10, max=70*1
0)
Problem9c <- ata_obj_absolute(Problem9c, theta_target, tif_target) #Speci
fy the ATA
Problem9c <- ata_solve(Problem9c, as.list=T) #Let's solve the ATA
Problem9c$items #see selected items
plot(Problem9c) # plotting information function
#End
```

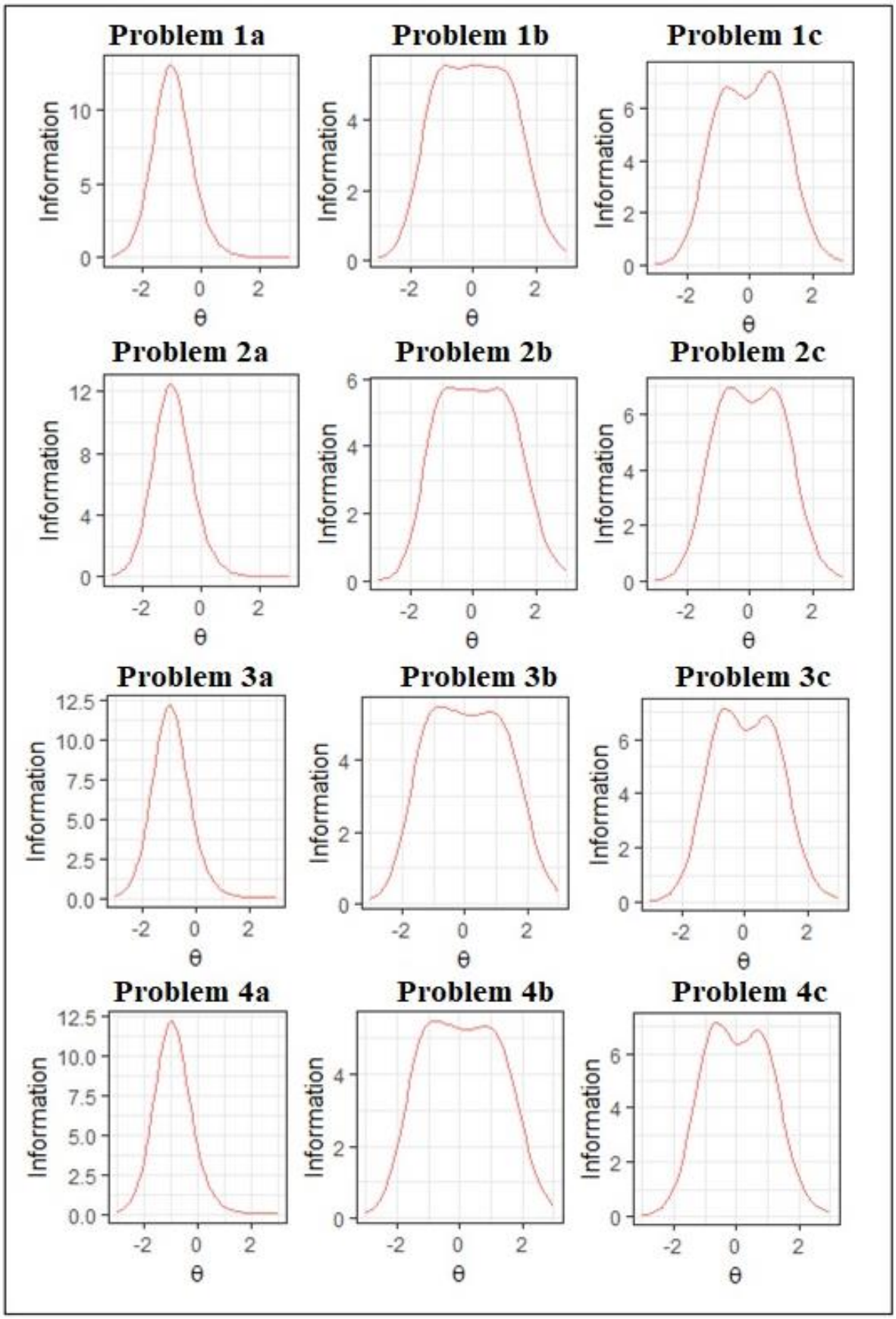

Figure 1. Plots for The Solutions in Examples from 1a to 4c.

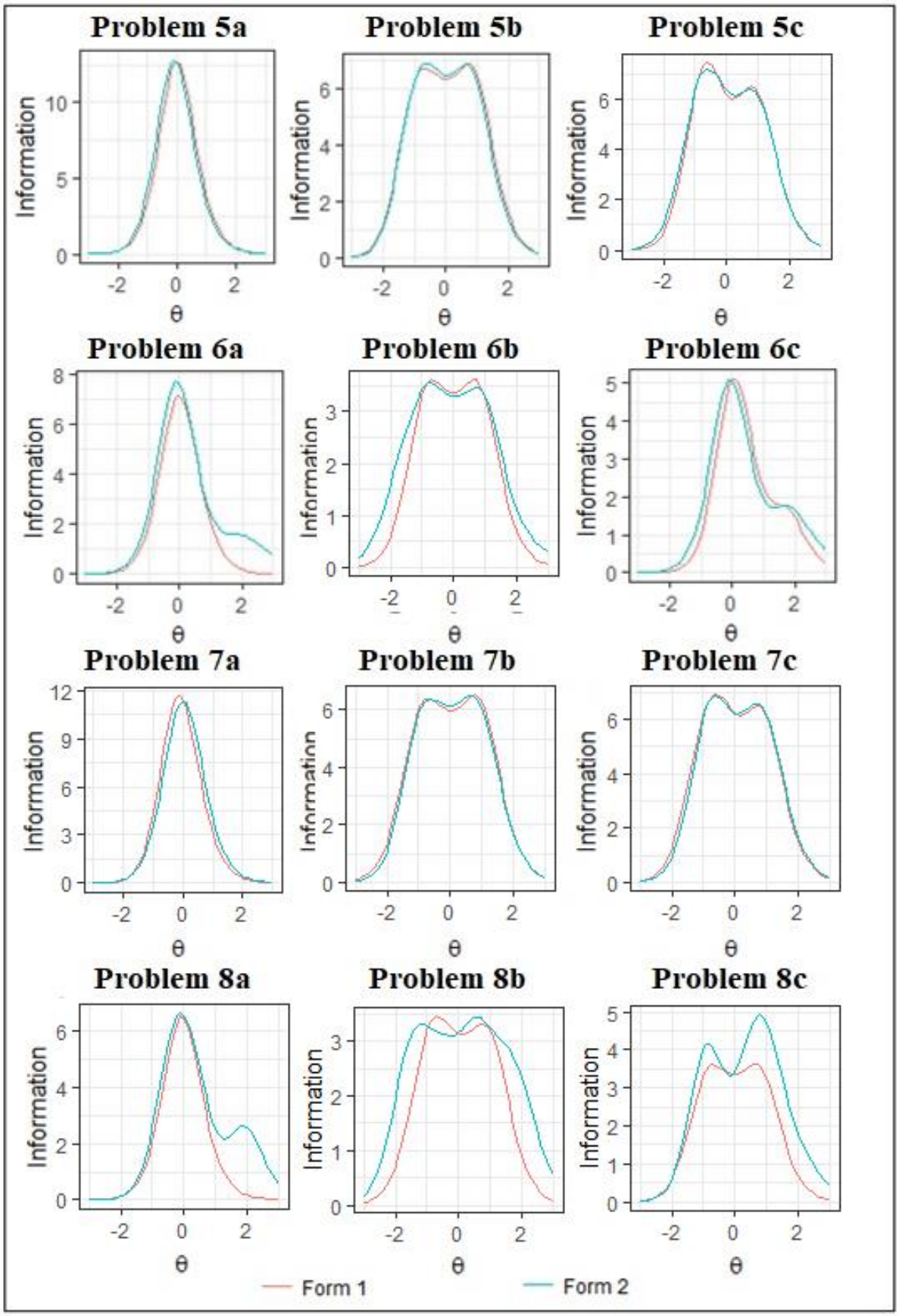

Figure 2. Plots for The Solutions in Examples from 5a to 8c.

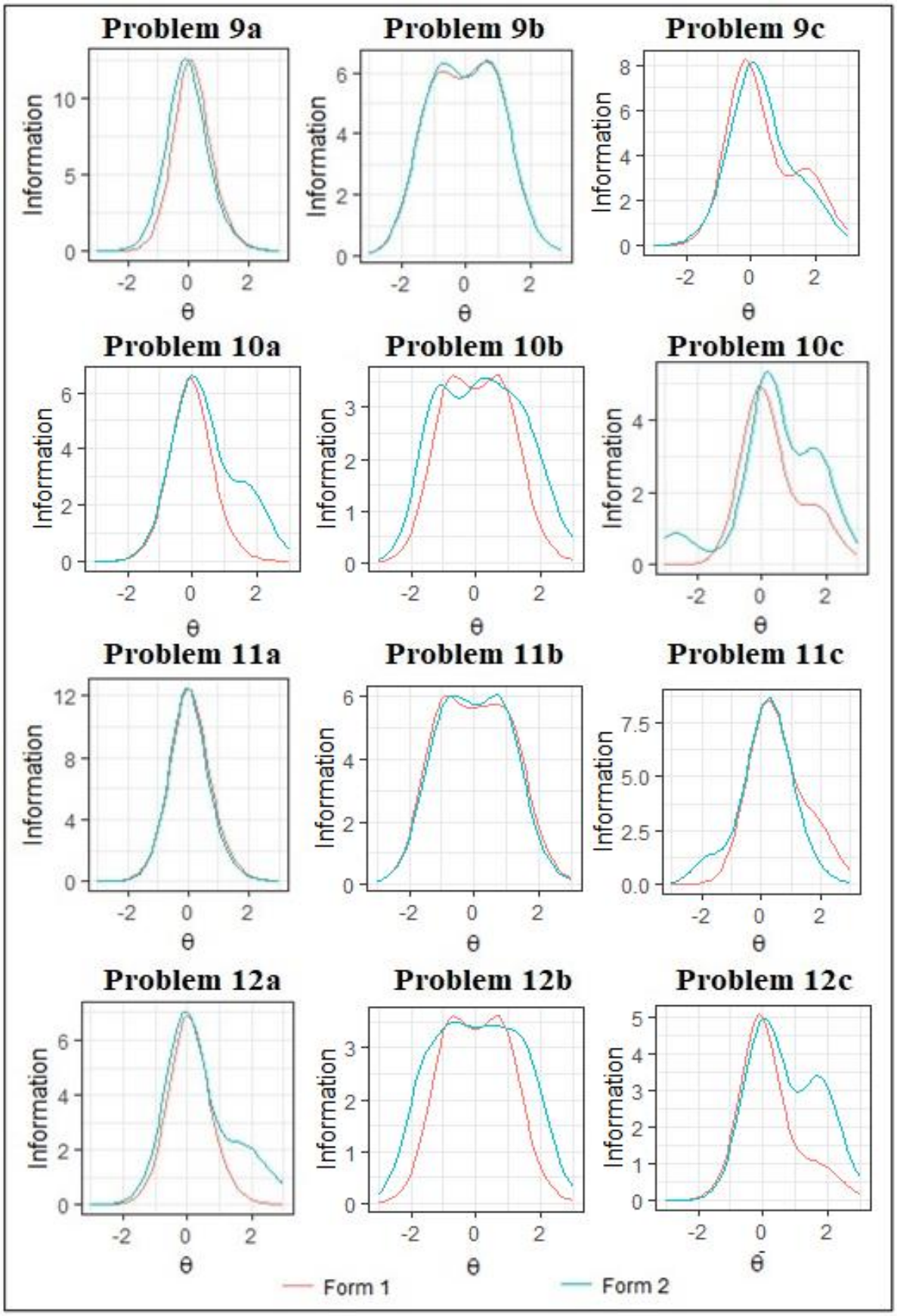

Figure 3. Plots for The Solutions in Examples from 9a to 12c.

Problem 10: Building unequal-length two forms with two constraints

Here, I create two unequal-length test forms with controlling content distribution and word count. There are three problems listed as Problem 10a, 10b and 10c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 10 examples, I pulled 5 items for form 1 and 8 items for form 2. In form 1, there are 1, 2, and 2 items from the content1, content2 and content3, respectively. In form 2, there are 2, 3, and 3 items from the content1, content2 and content3, respectively. The average word count of the items in both forms is between 30 and 80.

```
# Problem10a: maximize the information at the fixed theta point
Problem10a <- ata(items, 2, #Two test forms. Change accordingly!
max_use=1) #we don't want item-overlapping
Problem10a <-  ata_obj_relative(Problem10a, 0, #fixed theta point. Change
#accordingly!
"max")
Problem10a <- ata_constraint(Problem10a,1, min=5, max=5, forms=1) #5 #ite
ms in form 1
Problem10a <- ata_constraint(Problem10a,1,  min=8, max=8, forms=2) #8 #it
ems in form 2
#Now let's add content constraints for form 1. Change accordingly!
Problem10a <- ata_constraint(Problem10a, "content", min=1, max=1, level=1
,forms = 1)
Problem10a <- ata_constraint(Problem10a,"content", min=2, max=2, level=2,
forms = 1)
Problem10a <- ata_constraint(Problem10a, "content", min=2, max=2, level=3
,forms = 1)
#Now let's add content constraints for form 2. Change accordingly!
Problem10a <- ata_constraint(Problem10a, "content", min=2, max=2, level=1
,forms = 2)
Problem10a <- ata_constraint(Problem10a,"content", min=3, max=3, level=2,
forms = 2)
Problem10a <- ata_constraint(Problem10a, "content", min=3, max=3, level=3
,forms = 2)
#Now let's add word count constraints before solve the ATA. Change #accor
dingly!
Problem10a <-  ata_constraint(Problem10a, "word_count", min=30*10, max=80
*10)
Problem10a <- ata_solve(Problem10a, as.list=T) # Now let's solve the ATA
Problem10a$items #see selected items
plot(Problem10a) # plotting information function
#End

# Problem10b: maximize the information at theta interval of -1 to 1.
Problem10b <- ata(items, 2, max_use=1)
Problem10b <-  ata_obj_relative(Problem10b, seq(-1, 1, 0.50), #theta #int
erval. Change accordingly!
"max", flatten=0.10)
Problem10b <- ata_constraint(Problem10b,1, min=5, max=5, forms=1) #5 #ite
ms in form 1
Problem10b <- ata_constraint(Problem10b,1,  min=8, max=8, forms=2) #8 #it
ems in form 2
#Now let's add content constraints for form 1. Change accordingly!
Problem10b <- ata_constraint(Problem10b, "content", min=1, max=1, level=1
```

```
,forms = 1)
Problem10b <- ata_constraint(Problem10b,"content", min=2, max=2, level=2,
forms = 1)
Problem10b <- ata_constraint(Problem10b, "content", min=2, max=2, level=3
,forms = 1)
#Now let's add content constraints for form 2. Change accordingly!
Problem10b <- ata_constraint(Problem10b, "content", min=2, max=2, level=1
,forms = 2)
Problem10b <- ata_constraint(Problem10b,"content", min=3, max=3, level=2,
forms = 2)
Problem10b <- ata_constraint(Problem10b, "content", min=3, max=3, level=3
,forms = 2)
#Now let's add word count constraints before solve the ATA. Change #accor
dingly!
Problem10b <-  ata_constraint(Problem10b, "word_count", min=30*10, max=80
*10)
Problem10b <- ata_solve(Problem10b,as.list=T) #Solve the ATA
Problem10b$items #see selected items
plot(Problem10b) # plotting information function
#End

# Problem10c: absolute information target
theta_target=0 #One can specify theta interval as well. Change #according
ly!
tif_target= 5 # desired amount of information
Problem10c <- ata(items, 2, max_use=1)
Problem10c <- ata_constraint(Problem10c,1, min=5, max=5, forms=1) #5 #ite
ms in form 1
Problem10c <- ata_constraint(Problem10c,1,  min=8, max=8, forms=2) #8 #it
ems in form 2
#Now let's add content constraints for form 1. Change accordingly!
Problem10c <- ata_constraint(Problem10c, "content", min=1, max=1, level=1
,forms = 1)
Problem10c <- ata_constraint(Problem10c,"content", min=2, max=2, level=2,
forms = 1)
Problem10c <- ata_constraint(Problem10c, "content", min=2, max=2, level=3
,forms = 1)
#Now let's add content constraints for form 2. Change accordingly!
Problem10c <- ata_constraint(Problem10c, "content", min=2, max=2, level=1
,forms = 2)
Problem10c <- ata_constraint(Problem10c,"content", min=3, max=3, level=2,
forms = 2)
Problem10c <- ata_constraint(Problem10c, "content", min=3, max=3, level=3
,forms = 2)
#Now let's add word count constraints before solve the ATA.  Change #acco
rdingly!
Problem10c <-  ata_constraint(Problem10c, "word_count", min=30*10, max=80
*10)
Problem10c <- ata_obj_absolute(Problem10c, theta_target, tif_target, form
s = 1) # ATA for form 1
Problem10c <- ata_obj_absolute(Problem10c, theta_target, tif_target, form
s = 2) # ATA for form 2
Problem10c <- ata_solve(Problem10c, as.list=T) #Solve the ATA
```

```
Problem10c$items #see selected items
plot(Problem10c) # plotting information function
#End
```

Problem 11: Building equal-length two forms with three constraints

Here, I create two equal-length test forms with controlling content distribution, word count and time. There are three problems listed as Problem 11a, 11b and 11c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 11 examples, I pulled 10 items for both forms, and in both forms, there are 2, 3, and 5 items from the content1, content2 and content3, respectively. The average word count of the items in both forms is between 60 and 70. The average time to solve the item is between 200 and 300 seconds.

```
#Problem11a: maximize the information at the fixed theta point
Problem11a <- ata(items, 2, #we are building two forms
len=10, #total test length for a form. Change accordingly!
max_use=1) #Each item can be selected for a form only. Change #accordingl
y!
Problem11a <- ata_obj_relative(Problem11a, 0, # fixed theta point.
                          "max")
#Now let's add content constraints for the forms. Change accordingly!
Problem11a <- ata_constraint(Problem11a, "content", min=2, max=2, level=1
)
Problem11a <- ata_constraint(Problem11a,"content", min=3, max=3, level=2)
Problem11a <- ata_constraint(Problem11a, "content", min=5, max=5, level=3
)
#Now let's add word count constraints before solve the ATA. Change #accor
dingly!
Problem11a <-  ata_constraint(Problem11a, "word_count", min=60*10, max=70
*10)
#Now let's add time constraints before solve the ATA. Change #accordingly
!
Problem11a <-  ata_constraint(Problem11a, "time", min=200*10, max=300*10)
Problem11a <- ata_solve(Problem11a, as.list=T) #Let's solve the ATA
Problem11a$items #see selected items
plot(Problem11a) # plotting information function
#End

# Problem11b: Maximize the information at theta interval of -1 to 1.
Problem11b <- ata(items, 2, len=10, max_use=1)
Problem11b <-  ata_obj_relative(Problem11b, seq(-1, 1, 0.50), #Change #ac
cordingly!
                          "max", flatten=0.50)
#Let's add content distribution constraints. Change accordingly!
Problem11b <- ata_constraint(Problem11b, "content", min=2, max=2, level=1
)
Problem11b <- ata_constraint(Problem11b,"content", min=3, max=3, level=2)
Problem11b <- ata_constraint(Problem11b, "content", min=5, max=5, level=3
)
#Let's add word count constraints. Change accordingly!
Problem11b <-  ata_constraint(Problem11b, "word_count", min=60*10, max=70
*10)
#Let's add time constraints. Change accordingly!
```

```
Problem11b <-  ata_constraint(Problem11b, "time", min=200*10, max=300*10)
Problem11b <- ata_solve(Problem11b,as.list=T) #Now let's solve the ATA
Problem11b$items #see selected items
plot(Problem11b) # plotting information function
#End

# Problem11c:  Absolute information target
theta_target=c(0, 0.5) # either specify fixed theta point or theta #inter
val
tif_target= 8 # desired amount of information. Change accordingly!
Problem11c <- ata(items, 2, len=10, max_use=1)
#Let's add content distribution constraints. Change accordingly!
Problem11c <- ata_constraint(Problem11c, "content", min=2, max=2, level=1
)
Problem11c <- ata_constraint(Problem11c,"content", min=3, max=3, level=2)
Problem11c <- ata_constraint(Problem11c, "content", min=5, max=5, level=3
)
#Let's add word count constraints. Change accordingly!
Problem11c <-  ata_constraint(Problem11c, "word_count", min=60*10, max=70
*10)
#Let's add time constraints. Change accordingly!
Problem11c <-  ata_constraint(Problem11c, "time", min=200*10, max=300*10)
Problem11c <- ata_obj_absolute(Problem11c, theta_target, tif_target) #Spe
cify the ATA problem
Problem11c <- ata_solve(Problem11c, as.list=T) #Let's solve the ATA
Problem11c$items #see selected items
plot(Problem11c) # plotting information function
#End
```

Problem 12: Building unequal-length two forms with three constraints

Here, I create two unequal-length test forms with controlling content distribution, word count and time. There are three problems listed as Problem 12a, 12b and 12c. The codes for these problems are written for a fixed theta point, over a range and absolute amount of information cases, respectively. In all Problem 12 examples, I pulled 5 items for form 1 and 8 items for form 2. In form 1, there are 1, 2, and 2 items from the content1, content2 and content3, respectively. In form 2, there are 2, 3, and 3 items from the content1, content2 and content3, respectively. The average word count of the items in both forms is between 30 and 80. The average time to solve the item is between 100 and 400 seconds.

```
#Problem12a : Maximize the information at the fixed theta point of 0.
Problem12a <- ata(items, 2, #Building two forms
max_use=1) #we don't want item overlapping. Change accordingly!
Problem12a <-  ata_obj_relative(Problem12a, 0, # Change accordingly!
"max")
#Let's specify test lengths for the two forms. Change accordingly!
Problem12a <- ata_constraint(Problem12a,1, min=5, max=5, forms=1) #5 #ite
ms in form 1
Problem12a <- ata_constraint(Problem12a,1,  min=8, max=8, forms=2) #8 #it
ems in form 2
#Let's add content distribution constraints for form 1. Change #according
ly!
Problem12a <- ata_constraint(Problem12a, "content", min=1, max=1, level=1
,forms = 1)
```

```
Problem12a <- ata_constraint(Problem12a,"content", min=2, max=2, level=2,
forms = 1)
Problem12a <- ata_constraint(Problem12a, "content", min=2, max=2, level=3
,forms = 1)
#Let's add content distribution constraints for form 2. Change #according
ly!
Problem12a <- ata_constraint(Problem12a, "content", min=2, max=2, level=1
,forms = 2)
Problem12a <- ata_constraint(Problem12a,"content", min=3, max=3, level=2,
forms = 2)
Problem12a <- ata_constraint(Problem12a, "content", min=3, max=3, level=3
,forms = 2)
#Let's add word count constraints. Change accordingly!
Problem12a <-  ata_constraint(Problem12a, "word_count", min=30*10, max=80
*10)
#Let's add time constraints. Change accordingly!
Problem12a <-  ata_constraint(Problem12a, "time", min=100*10, max=400*10)
Problem12a <- ata_solve(Problem12a, as.list=T) #Now, let's solve the ATA
Problem12a$items #see selected items
plot(Problem12a) # plotting information function
#End

#Problem12b: maximize the information at theta interval of -1 to 1.
Problem12b <- ata(items, 2, max_use=1)
Problem12b <-  ata_obj_relative(Problem12b, seq(-1, 1, 0.50),  # theta #i
nterval. Change accordingly!
"max", flatten=0.10)
Problem12b <- ata_constraint(Problem12b,1, min=5, max=5, forms=1) #5 #ite
ms in form 1
Problem12b <- ata_constraint(Problem12b,1,  min=8, max=8, forms=2) #8 #it
ems in form 2
Problem12b <- ata_constraint(Problem12b, "content", min=1, max=1, level=1
,forms = 1) #Form1C1
Problem12b <- ata_constraint(Problem12b,"content", min=2, max=2, level=2,
forms = 1) #Form1C2
Problem12b <- ata_constraint(Problem12b, "content", min=2, max=2, level=3
,forms = 1) #Form1C3
Problem12b <- ata_constraint(Problem12b, "content", min=2, max=2, level=1
,forms = 2) #Form2C1
Problem12b <- ata_constraint(Problem12b,"content", min=3, max=3, level=2,
forms = 2) #Form2C2
Problem12b <- ata_constraint(Problem12b, "content", min=3, max=3, level=3
,forms = 2) #Form2C3
Problem12b <-  ata_constraint(Problem12b, "word_count", min=30*10, max=80
*10) #word counts
Problem12b <-  ata_constraint(Problem12b, "time", min=100*10, max=400*10)
#time constraints
Problem12b <- ata_solve(Problem12b,as.list=T) #Now, let's solve the ATA
Problem12b$items #see selected items
plot(Problem12b) # plotting information function
#End
```

```
#Problem12c: Absolute information target
theta_target=0 #the theta point where you want the absolute amount of #in
formation
tif_target= 5 # desired amount of information. Change accordingly!
Problem12c <- ata(items, 2, max_use=1)
Problem12c <- ata_constraint(Problem12c,1, min=5, max=5, forms=1) #5 #ite
ms in form 1
Problem12c <- ata_constraint(Problem12c,1,  min=8, max=8, forms=2) #8 #it
ems in form 2
Problem12c <- ata_constraint(Problem12c, "content", min=1, max=1, level=1
,forms = 1) #Form1 C1
Problem12c <- ata_constraint(Problem12c,"content", min=2, max=2, level=2,
forms = 1) #Form1 C2
Problem12c <- ata_constraint(Problem12c, "content", min=2, max=2, level=3
,forms = 1) #Form1 C3
Problem12c <- ata_constraint(Problem12c, "content", min=2, max=2, level=1
,forms = 2) #Form2 C1
Problem12c <- ata_constraint(Problem12c,"content", min=3, max=3, level=2,
forms = 2) #Form2 C2
Problem12c <- ata_constraint(Problem12c, "content", min=3, max=3, level=3
,forms = 2) #Form2 C3
Problem12c <-  ata_constraint(Problem12c, "word_count", min=30*10, max=80
*10) #word counts
Problem12c <-  ata_constraint(Problem12c, "time", min=100*10, max=400*10)
#time constraints
Problem12c <- ata_obj_absolute(Problem12c, theta_target, tif_target, form
s = 1) # ATA for form 1
Problem12c <- ata_obj_absolute(Problem12c, theta_target, tif_target, form
s = 2) # ATA for form 2
Problem12c <- ata_solve(Problem12c, as.list=T) #Now, ATA is ready to #sol
ve!
Problem12c$items #see selected items
plot(Problem12c) # plotting information function
#End of the tutorial
```

Important Notes

1. It is important to note that finding a solution in any example depends on the psychometric characteristics of the items in the pool.
2. In this paper, I used a simulated item pool. Thus, when you replicate the item pool, you may or may not find the same solutions.
3. The item pool was generated based on the 3PL Item Response Theory Model. In case you use different model than the 3PL, you should change the item parameters accordingly. For example, when you use Rasch model, you should fix all discrimination parameters at 1 and all pseudo-guessing parameters at 0.
4. All of the codes were carefully written, and their functionality was checked again and again by the author. In case you have problems to run any example, you can try the following steps first. If you still cannot find a solution, please do not hesitate contacting the author.
    a. If you use simulated item pool, you may want to re-generate the item pool, and try again.

b. In case you do not find a solution for your own cases, you can try relaxing the constraints. For example, you can specify lower amount of absolute information (see Problems 1c, 2c, 3c and 4c) or maximize the information in a narrower theta interval (see Problems 1b, 2b, 3b and 4b).

c. In same cases, you may want to allow item overlapping, especially when you have limited number of total items.

5. Finding a solution also depends on the constraints you specify. The likelihood of finding a solution becomes difficult as you use strict restrictions.

6. For the demonstration purposes, the constraints used in this study are the hypothetical constraints (e.g., content area, word count and time). You can use your own constraints or more logical ones.

REFERENCES

Berkelaar, M., & others (2015). Package 'lpSolve'. Retrieved from https://cran.r-project.org/web/packages/lpSolve/lpSolve.pdf

Berkelaar, M., Eikland, K., & Notebaert, P. (2004). lp_solve reference guide menu. Retrieved from http://lpsolve.sourceforge.net/5.5/

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord & M. R. Novick (Eds.), *Statistical theories of mental test scores* (pp. 397–479). Reading, MA: Addison-Wesley.

Cor, K., Alves, C., & Gierl, M. (2008). Conducting automated test assembly using the Premium Solver Platform Version 7.0 with Microsoft Excel and the large-Scale LP/QP solver engine add-in. *Applied Psychological Measurement*, *32*, 652-663. doi: 10.1177/0146621608316603

Cor, K., Alves, C., & Gierl, M. (2009). Three applications of automated test assembly within a user-friendly modeling environment. *Practical Assessment, Research, & Evaluation*, *14*(14), 1-23. Retrieved from http://pareonline.net/getvn.asp?v=14%26n=14

Diao, Q., & van der Linden, W. J. (2011). Automated test assembly using lp_solve version 5.5 in R. *Applied Psychological Measurement*, *35*(5), 398-409. doi: 10.1177/0146621610392211

Donoghue, J. R. (2015). *Comparison of integer programming (IP) solvers for automated test assembly (ATA)* (Research Report No. ETS RR-15-05). Retrieved from https://onlinelibrary.wiley.com/doi/epdf/10.1002/ets2.12051

Gierl, M. J., Daniels, L., & Zhang, X. (2017). Creating parallel forms to support on-demand testing for undergraduate students in psychology. *Journal of Measurement and Evaluation in Education and Psychology*, *8*(3), 288-302. doi: 10.21031/epod.305350

Han, K. T., & Rudner, L. M. (2014). Item *pool construction using mixed integer quadratic programming* (MIQP) (Research Report No. RR-14-01). Retrieved from https://files.eric.ed.gov/fulltext/ED558455.pdf

International Business Machines (2006). *ILOG CPLEX 10.0 user's manual*. Paris: ILOG SA.

Konis, K. (2016). Package 'lpSolveAPI'. Retrieved from https://cran.r-project.org/web/packages/lpSolveAPI/lpSolveAPI.pdf

Luecht, R. M. (1998). Computer-assisted test assembly using optimization heuristics. *Applied Psychological Measurement*, *22*(3), 224-236. doi: 10.1177/01466216980223003

Luo, X. (2018). Package 'xxIRT'. Retrieved from https://cran.r-project.org/web/packages/xxIRT/xxIRT.pdf

Van der Linden, W. J. (2006). *Linear models for optimal test design*. New York, NY: Springer Science & Business Media.

Test Formları Oluşturma Üzerine Öğretici R Çalışması

Giriş

Eğitimde ve psikolojide ölçmenin temel amaçlarından biri öğrenci yeteneğini veya bilgisini en doğru veya en az hata ile ölçmektir. Ancak iyi bir ölçme aracı olmadan bunu başarabilmek oldukça zordur. Geliştirdiğimiz test formu veya ölçme aracının zorluk derecesi, içerik alanı, maddelerin kelime sayısı gibi psikometrik özelliklerinin önceden belirlenmesi gerekir.

İstenilen özelliklerde test formu oluşturmak özellikle bilgisayar ortamında bireye uyarlanmış testler, bireye uyarlanmış çok aşamalı testler, kâğıt-kalem testleri için büyük öneme sahiptir. Bireye uyarlanmış testlerde aynı maddeleri çok fazla kişinin almasını engellemek için, birden fazla test formu oluşturmak bir gerekliliktir. Ancak oluşturulan test formlarının birbirine birçok açıdan benzer olması bir zorunluluktur. Örneğin birbirine paralel formlar oluşturulmak istendiğinde test formaların zorluk derecesi (kolay, zor form gibi), maddelerin konu alanı (doğal sayılar, kümeler, fonksiyonlar gibi), formlardaki maddelerin uzunlukları (kelime sayısı) birbiriyle aynı veya benzer olmalıdır. Bunu sağlama işlemine *otomatik test derleme* adı verilmektedir.

Otomatik Test Derleme (OTD) çok bilinmeyenli karmaşık denklemleri veya kısıtlamaları çözmek için kullanılan tamsayı (integer) programlama metodudur. OTD psikometride istenilen kriterlere sahip test formu oluşturmada kullanılmaktadır. İstenilen özellik veya kısıtlama ile kastedilen örneğin test formunun zorluk derecesi, maddelerin içerik veya konu alanları, maddelerin uzunluları olabilir.

Literatürde otomatik test derleme ile alakalı birçok faydalı kaynak bulunmaktadır. Wim J. Van der Linden (2005) tarafından yazılan kitap bu alandaki en faydalı kaynakların başında gelmektedir. Yazar kitabında eşik-dayanaklı ve norm dayanaklı test formlarının nasıl oluşturulduğunu detaylıca anlatmaktadır. Cor, Alves ve Gierl (2008, 2009) ve Gierl, Daniels ve Zhang (2017) Microsoft Excel'de isteğe bağlı testlerin nasıl hazırlanabileceğini göstermişlerdir. Diao ve van der Linden (2011) karmaşık otomatik test derlemenin R'da nasıl yapılabileceğini anlatmışlardır. Ancak yazarlar yanlızca üç farklı problem üzerinde durmuş ve sadece bir problem durumuna ait R kodunu okuyucuyla paylaşmışlardır.

Otomatik test derleme yapmak için kullanılabilecek çok sayıda bilgisayar programı bulunmaktadır. Bunlardan bazıları ILOG CPLEX, LINGO 12.0, LPSolve IDE, "IpSolve ve "lpSolveAPI" R paketleri.

Çalışmanın amacı

Bu çalışmanın amacı, araştırmacılar ve uygulayıcılar için farklı kısıtlamalar altında isteğe bağlı test formlarını nasıl oluşturabileceklerini göstermektir. Otomatik test derleme hakkında literatürde bol miktarda çalışma olmasına rağmen araştırmacıların kullanabileceği ücretsiz R kodları sınırlı miktarda bulunmaktadır. Çalışmada birbirinden farklı otuz dokuz farklı problem için açıklamalı R kodu verilmiştir. Kelime sınırı nedeniyle otuz altı tanesi bu belgede verilmiştir. Geriye kalan üç problem daha karmaşık durumlar altında otomatik test derlemenin nasıl çözülebileceğini göstermekte olup, Ek A'da verilmiştir. Örnekler, istenilen koşullar altında tek bir test formu, çoklu test formları ve daha karmaşık test formları oluşturmayı içermektedir. Açıklamalı R kodlarının yanı sıra her bir problem için form bilgi fonksiyonu da verilmiştir.

"xxIRT" R Paketi

Bu çalışmada, verilen tüm örnekleri çözmek için "xxIRT" R paketi versiyon 2.1.0 (Luo, 2018) kullanılmıştır. Bu paket yeni yayımlanmış olup, OTD problemlerini çözmek için "lpSolveAPI" R paketini kullanmaktadır. Paket içerisinde bulunan ve kullanıcı tarafından bilinmesi gereken önemli fonksiyonlardan bazıları *ata, ata_obj_relative, ata_obj_absolute, ata_constraint, ata_solve* fonksiyonlarıdır.

Fonksiyon *ata* kaç tane test formu oluşturulacağı (örneğin iki), formun uzunluğu (örneğin 5 madde) ve madde havuzu olarak neyi kullanacağı bilgilerini kullanarak OTD için problemi tanımlar. Diğer iki temel fonksiyondan *ata_obj_relative* fonksiyonu test bilgi fonksiyonunu sabit bir yetenek seviyesi noktasında maksimum hale getirirken, *ata_obj_absolute* ise test bilgi fonksiyonunu bir yetenek seviyesi aralığında maksimum hale getirir. Test bilgi fonksiyonunun verilen bir yetenek seviyesi aralığında maksimum hale getirilmesi istendiğinde, kullanıcı tarafından tanımlanan yetenek aralığının yanı sıra, yetenek seviyesi artış miktarları da belirlenebilir. Herhangi bir yetenek seviyesinde veya yetenek aralığında mutlak test bilgi fonksiyonuna sahip bir test formu oluşturulmak istendiğinde de istendiğinde *ata_obj_absolute* fonksiyonu kullanılır. Bu durumda da artış noktaları kullanıcı tarafından belirlenebilir. Yetenek seviyesi için sabit bir nokta değil de aralık belirlenmesi durumunda artış noktaları önemlidir, çünkü bu test bilgisi fonksiyonunun şeklini önemli ölçüde değiştirebilir.

Daha önce tartışıldığı gibi, kısıtlamalar bir OTD probleminin önemli unsurlarıdır. Kısıtlamaları oluşturmak için kullanılacak fonksiyon *ata_constraint* fonksiyonudur. Bu fonksiyon belirtilen kısıtlamayı OTD probleminin modeline ekler. Formdaki konu alanlarından kaçar tane madde seçileceği, formdaki maddelerin toplam sayılarının hangi aralıkta olacağı gibi birçok kısıtlama bu fonksiyon kullanılarak modele eklenebilir. Tüm kısıtlamalar da girildikten sonra OTD modeli çözmek için *ata_solve* fonksiyonu kullanılır. Bu fonksiyon kullanılıp, test formları için uygun maddeler seçildikten sonra hangi maddelerin seçildiği ve oluşturulan formların bilgi fonksiyonlarının grafikleri de çizilebilir. Temel fonksiyonlar veya komutlar hakkında daha fazla bilgi için "xxIRT" paketine başvurabilir. Bu fonksiyonlarının kullanımı gösteren açıklamalı R kodları aşağıda vermiştir.

Örnek R Kodları

Çalışmada verilen problem durumlarını çözmek için öncelikli olarak 1000 maddeden oluşan 3 parametreli madde tepki kuramına göre madde havuzu oluşturulmuştur. Bununla birlikte her bir madde için rastgele a) içerik alanı (örneğin cebirsel ifadeler, sayılar, denklemler gibi), b) kelime sayısı (örneğin, her bir madde için 30 ile 150 arasında değişen) ve c) maddeyi çözmek için gerekli zaman (örneğin, her bir madde için 100 saniye ile 400 saniye arasında) atanmıştır. Rastgele atanan bu değerler problem durumuna bağlı olarak çözülmesi istenen otomatik test derleme işleminde kullanılmak içindir.

Çalışmanın ana metninde 12 farklı OTD problemi sunulmuş olup, her bir problem 3 farklı durum altında çözülmüştür. Her bir problemin a şıkkı sabit bir yetenek seviyesi noktasında bilgi fonksiyonu maksimum hale çıkarılmak istendiğinde, b şıkkı test bilgi fonksiyonu yetenek seviyesi istenilen yetenek seviyesi aralığında maksimum hale getirilmek istendiğinde, c şıkkı ise istenilen miktarda test bilgisi elde edilmek istendiğinde OTD'nin nasıl çözüleceğini göstermektedir. Çalışmada listelenen toplamda 36 örnek için, simülasyonla üretilmiş aynı madde havuzu kullanılmıştır. Madde sayısı, madde parametrelerinin dağılımı ve varsayımsal kısıtlamalar kullanıcı tarafından değiştirilebilir. Aşağıda her bir problem durumunda çözülen OTD problemindeki test formlarının özellikleri verilmiştir.

Problem 1: Herhangi bir kısıtlama olmaksızın tek bir test formunun nasıl oluşturulacağını göstermektedir. Formda 10 madde yer almaktadır.

Problem 2: İçerik ağırlıklandırılması göz önünde bulundurularak tek bir test formunun nasıl oluşturulacağını göstermektedir. Test formu için 3 farklı içerikten sırasıyla 2, 3 ve 5 madde olmak üzere 10 madde seçilmiştir.

Problem 3: İçerik ağırlıklandırma ve kelime sayısı göz önünde bulundurularak iki farklı kısıtlamayla tek bir test formunun nasıl oluşturulabileceğini göstermektedir. Test formu için 3 farklı içerikten sırasıyla 2, 3 ve 5 madde olmak üzere 10 madde seçilmiştir. Formdaki maddelerin ortalama kelime sayısının 60 ile 70 arasında olması istenmiştir.

Problem 4: İçerik ağırlıklandırma, kelime sayısı ve zaman göz önünde bulundurularak üç farklı kısıtlamayla tek bir test formunun nasıl oluşturulabileceğini göstermektedir. Test formu için 3 farklı içerikten sırasıyla 2, 3 ve 5 madde olmak üzere 10 madde seçilmiştir. Formdaki maddelerin ortalama

kelime sayısının 60 ile 70 arasında olması istenmiştir. Maddeleri çözmek için gereken ortalama zaman 200 saniye ile 300 saniye arasında olacak şekilde ayarlanmıştır.

Problem 5: Herhangi bir kısıtlama olmaksızın eşit test uzunluğuna sahip iki test formunun nasıl oluşturulacağını göstermektedir. Her iki formda da 10 madde yer almaktadır.

Problem 6: Herhangi bir kısıtlama olmaksızın farklı test uzunluğuna sahip iki test formunun nasıl oluşturulacağını göstermektedir. Birinci form için 5, ikinci form için 8 madde seçilmiştir.

Problem 7: İçerik ağırlıklandırılması göz önünde bulundurularak eşit test uzunluğuna sahip iki test formunun nasıl oluşturulacağını göstermektedir. Her iki test formu için 3 farklı içerikten sırasıyla 2, 3 ve 5 madde olmak üzere 10 madde seçilmiştir.

Problem 8: İçerik ağırlıklandırılması göz önünde bulundurularak farklı test uzunluğuna sahip iki test formunun nasıl oluşturulacağını göstermektedir. Birinci test formu için üç farklı içerik alanından sırasıyla 1, 2 ve 2 olmak üzere toplam 5 madde, ikinci test formu için üç farklı içerik alanından sırasıyla 2, 3 ve 3 olmak üzere toplam 8 madde seçilmiştir.

Problem 9: İçerik ağırlıklandırma ve kelime sayısı göz önünde bulundurularak iki farklı kısıtlamayla eşit test uzunluğuna sahip iki test formunun nasıl oluşturulabileceğini göstermektedir. Her iki test formu için 3 farklı içerikten sırasıyla 2, 3 ve 5 madde olmak üzere 10 madde seçilmiştir. Formlardaki maddelerin ortalama kelime sayısının 60 ile 70 arasında olması istenmiştir.

Problem 10: İçerik ağırlıklandırma ve kelime sayısı göz önünde bulundurularak iki farklı kısıtlamayla farklı test uzunluğuna sahip iki test formunun nasıl oluşturulabileceğini göstermektedir. Birinci test formu için üç farklı içerik alanından sırasıyla 1, 2 ve 2 olmak üzere toplam 5 madde, ikinci test formu için üç farklı içerik alanından sırasıyla 2, 3 ve 3 olmak üzere toplam 8 madde seçilmiştir. Formlardaki maddelerin ortalama kelime sayısının 30 ile 80 arasında olması istenmiştir.

Problem 11: İçerik ağırlıklandırma, kelime sayısı ve zaman göz önünde bulundurularak üç farklı kısıtlamayla eşit test uzunluğuna sahip iki test formunun nasıl oluşturulabileceğini göstermektedir. Her iki test formu için 3 farklı içerikten sırasıyla 2, 3 ve 5 madde olmak üzere 10 madde seçilmiştir. Formlardaki maddelerin ortalama kelime sayısının 60 ile 70 arasında olması istenmiştir. Maddeleri çözmek için gereken ortalama zaman 200 saniye ile 300 saniye arasında olacak şekilde ayarlanmıştır.

Problem 12: İçerik ağırlıklandırma, kelime sayısı ve zaman göz önünde bulundurularak üç farklı kısıtlamayla farklı test uzunluğuna sahip iki test formunun nasıl oluşturulabileceğini göstermektedir. Birinci form için 3 farklı içerikten sırasıyla 1, 2 ve 2 olmak üzere toplam 5 madde, ikinci form için 3 farklı içerikten sırasıyla 2, 3 ve 3 olmak üzere toplam 8 madde seçilmiştir. Formlardaki maddelerin ortalama kelime sayısının 30 ile 80 arasında olması istenmiştir. Maddeleri çözmek için gereken ortalama zaman 100 saniye ile 400 saniye arasında olacak şekilde ayarlanmıştır.

Önemli Notlar

1. Herhangi bir Otomatik Test derleme probleminin çözümü madde havuzundaki maddelerin kalitesine bağlıdır.
2. Bu çalışmada simülasyonla üretilmiş madde havuzu kullanılmıştır. Dolayısıyla bir başkası kodları çalıştırdığında aynı sonuçlara ulaşamayabilir veya daha iyi sonuçlar elde edebilir.
3. Bu çalışmada test bilgi fonksiyonlarının hesaplanması için 3 parametreli Madde Tepki Kuramı Modeli kullanılmıştır. Bir başkası isteğe göre farklı modeller kullanabilir.
4. Çalışmada herhangi bir problemin çözülememesi durumunda aşağıdaki yöntemler denenebilir. Hâlâ problem yaşanması durumunda yazar ile irtibata geçmekten çekinmeyiniz.
    a. Simülasyonla üretilmiş madde havuzu kullanılmışsa, madde havuzu tekrardan üretilebilir.

b. Bazı kısıtlamalar gevşetilebilir. Örneğin yetenek seviyesi aralığı daraltılabilir veya mutlak bir bilgi seviyesi miktarı istenildiğinde, istenilen bilgi miktarı azaltılabilir.

c. Bazı durumlarda formlardaki ortak madde olmasına izin verilebilir.

5. Unutulmamalıdır ki herhangi bir OTD problemini çözmek belirtilen kısıtlamalara bağlıdır. Kısıtlamaların zorluğu veya miktarı arttıkça OTD probleminin çözülme imkânı azalır.
6. Bu çalışmada gösterim amacıyla içerik ağırlıklandırma, kelime sayısı ve zaman olmak üzere varsayımsal kısıtlamalar kullanılmıştır. Kullanıcılar kendi durumlarına göre farklı kısıtlamalar kullanabilirler.

Appendix A. Three Complex Examples

Preparing for the analysis

```
# Do not run!
#Install the "xxIRT" package first
install.packages("xxIRT",repos = "http://cran.us.r-project.org")
require("xxIRT")

# Let's generate an item pool
set.seed(10)
items=as.data.frame(cbind(
a=runif(1000, 0.5, 1.5), #a parameters from a uniform distribution. #Chan
ge
#accordingly!
b=runif(1000, -2, 2), #b parameters from a uniform distribution. Change
#accordingly!
c=runif(1000, 0, 0.20), #c parameters from a uniform distribution. Change
#accordingly!
content=sample(1:3,1000,replace = T), #3 content areas (e.g., algebra,num
bers#equations)
word_count=sample(30:150,1000,replace = T), #assigning random word counts
for #each item.
time=sample(100:400,1000,replace = T))) #assigning random time between fo
r #e#ach item.
#End

# DO NOT RUN
#BUILDING MORE COMPLEX TEST FORMS

#Example 1: maximize the information at the different fixed theta points

#Pulling five sets of item (5 test forms)
#For Form 1 and 2 maximize the information at the fixed theta point of -1
(two easy forms)
#For Form 3 maximize the information at the fixed theta point of 0 (1 med
ium #form)
#For Form 4 and 5 maximize the information at the fixed theta point of 1
(two #hard forms)
#Test length for form 1 and 2 is 10 (2, 3, 5 items from Contents 1, 2 and
3, #respectively)
#Test length for form 3 is 15 (5, 6, 4 items from Contents 1, 2 and 3,res
pect#ively)
#Test length for form 4 and 5 is 20 (4, 7, 9 items from Contents 1, 2 and
3, #respectively)
#For Forms 1 and 2, average word count across the items in the forms is
#between 30 and 80
#For Form3, average word count across the items in the forms is between
# 50 and 90
#For Forms 4 and 5, average word count across the items in the forms is
#between 20 and 100
#For Form 1 and 2, the average time to solve the item is between 200 and
250 #seconds
#For Form 3, average time to solve the item is between 200 and 300 second
```

```
s
#For Form 4 and 5, the average time to solve the item is between 200 and
400 #seconds
Ex1 <- ata(items, 5, #building 5 test forms at the same time!
#Change accordingly!
           max_use=1) #we don't want item overlapping!
Ex1 <-  ata_obj_relative(Ex1, -1, #fixed theta point for forms 1 and 2.
# Change accordingly!
                         "max",
                         forms=c(1,2) #the specified theta point of -1 wa
s
#for both forms 1 and 2 only!
)
Ex1 <-  ata_obj_relative(Ex1, 0, #fixed theta point for form 3.
#Change accordingly!
                         "max",
                         forms=3 #the specified theta point of 0 was for
both #form 3 only!
)
Ex1 <-  ata_obj_relative(Ex1, 1, #fixed theta point for forms 4 and 5. Ch
ange #accordingly!
                         "max",
                         forms=c(4,5) #the specified theta point of 1 was
for #both forms 4 and 5 only!
)
Ex1 <- ata_constraint(Ex1,1, min=10, max=10, forms=c(1,2)) #Test length f
or
#forms 1 & 2
Ex1 <- ata_constraint(Ex1,1,  min=15, max=15, forms=3) #Test length for f
orm #3
Ex1 <- ata_constraint(Ex1,1,  min=20, max=20, forms=c(4,5)) #Test length
for #forms 4 & 5
#For forms 1 and 2, specify content distributions for content 1, 2, and 3
,
#respectively.
Ex1 <- ata_constraint(Ex1, "content", min=2, max=2, level=1,forms=c(1,2))
Ex1 <- ata_constraint(Ex1,"content", min=3, max=3, level=2, forms=c(1,2))
Ex1 <- ata_constraint(Ex1, "content", min=5, max=5, level=3,forms=c(1,2))
#For form 3, specify content distributions for content 1, 2, and 3,
# respectively.
Ex1 <- ata_constraint(Ex1, "content", min=5, max=5, level=1,forms = 3)
Ex1 <- ata_constraint(Ex1,"content", min=6, max=6, level=2, forms = 3)
Ex1 <- ata_constraint(Ex1, "content", min=4, max=4, level=3,forms = 3)
#For forms 4 and 5, specify content distributions for content 1, 2, and 3
,
#respectively.
Ex1 <- ata_constraint(Ex1, "content", min=4, max=4, level=1,forms=c(4,5))
Ex1 <- ata_constraint(Ex1,"content", min=7, max=7, level=2, forms=c(4,5))
Ex1 <- ata_constraint(Ex1, "content", min=9, max=9, level=3,forms=c(4,5))
#For forms 1 and 2, specify word counts.
Ex1 <-  ata_constraint(Ex1, "word_count", min=30*10, max=80*10,forms=c(1,
2))
#For form 3, specify word counts.
```

```
Ex1 <-  ata_constraint(Ex1, "word_count", min=50*10, max=90*10,forms=3)
#For forms 4 and 5, specify word counts.
Ex1 <-  ata_constraint(Ex1, "word_count", min=40*10, max=100*10,forms=c(4
,5))
#For forms 1 and 2, specify time as seconds.
Ex1 <-  ata_constraint(Ex1, "time", min=250*10, max=400*10,forms=c(1,2))
#For form 3, specify time as seconds.
Ex1 <-  ata_constraint(Ex1, "time", min=200*10, max=300*10,forms=3)
#For forms 4 and 5, specify time as seconds.
Ex1 <-  ata_constraint(Ex1, "time", min=200*10, max=400*10,forms=c(4,5))
Ex1 <- ata_solve(Ex1, as.list=T) # Now, solve the ATA
Ex1$items #see selected items
plot(Ex1) # plotting information function
#End
```

```
#Example 2: maximize the information at the different theta intervals

#Pulling five sets of item (5 test forms)
#For forms 1 and 2 maximize the information at theta interval of -1.5 to
-0.5
#For form 3 maximize the information at theta interval of 0 to 0.5
#For forms 4 and 5 maximize the information at theta interval of 0.5 to 1
.5
#Test length for forms 1 and 2 is 10 (2, 3, 5 items from Contents 1, 2 an
d 3, #respectively)
#Test length for form 3 is 15 (5, 6, 4 items from Contents 1, 2 and 3,
#respectively)
#Test length for forms 4 and 5 is 20 (4, 7, 9 items from Contents 1, 2 an
d 3, #respectively)
#For forms 1 and 2, average word count across the items in the forms is
# between 30 and 80
#For form 3, average word count across the items in the forms is between
#50 and 90
#For forms 4 and 5, average word count across the items in the forms is
#between 20 and 100
#For forms 1 and 2 average time to solve the item is between 200 and 250
#seconds
#For form 3 average time to solve the item is between 200 and 300 seconds
#For forms 4 and 5 average time to solve the item is between 200 and 400
#seconds
Ex2 <- ata(items, 5, max_use=1)
Ex2 <-  ata_obj_relative(Ex2, seq(-1.5, -0.5, 0.10), #theta interval of -
1.5 #to -0.5
                         "max", flatten=0.50,
                         forms=c(1,2) #the specified interval is for Form
s 1 #and 2. Change accordingly!
)
Ex2 <-  ata_obj_relative(Ex2, seq(0, 0.5, 0.10), #theta interval of 0 to
0.5
                         "max", flatten=0.50,
                         forms=3) #the specified interval is for Form 3.
#Change accordingly!
Ex2 <-  ata_obj_relative(Ex2, seq(0.5, 1.5, 0.10), # interval of 0 to 1.5
```

```
                          "max", flatten=0.50,
forms=c(4,5) #the specified interval is for Forms 4&5. Change accordingly
!
)
Ex2 <- ata_constraint(Ex2,1, min=10, max=10, forms=c(1,2)) #Test length f
or
#forms 1 & 2
Ex2 <- ata_constraint(Ex2,1,  min=15, max=15, forms=3) #Test length for f
orm 3
Ex2 <- ata_constraint(Ex2,1,  min=20, max=20, forms=c(4,5)) #Test length
for #forms 4 & 5
#For forms 1 and 2, specify content distributions for content 1, 2, and 3
,
#respectively.
Ex2 <- ata_constraint(Ex2, "content", min=2, max=2, level=1,forms=c(1,2))
Ex2 <- ata_constraint(Ex2,"content", min=3, max=3, level=2, forms=c(1,2))
Ex2 <- ata_constraint(Ex2, "content", min=5, max=5, level=3,forms=c(1,2))
#For form 3, specify content distributions for content 1, 2, and 3,
#respectively.
Ex2 <- ata_constraint(Ex2, "content", min=5, max=5, level=1,forms = 3)
Ex2 <- ata_constraint(Ex2,"content", min=6, max=6, level=2, forms = 3)
Ex2 <- ata_constraint(Ex2, "content", min=4, max=4, level=3,forms = 3)
#For forms 4 and 5, specify content distributions for content 1, 2, and 3
,
#respectively.
Ex2 <- ata_constraint(Ex2, "content", min=4, max=4, level=1,forms=c(4,5))
Ex2 <- ata_constraint(Ex2,"content", min=7, max=7, level=2, forms=c(4,5))
Ex2 <- ata_constraint(Ex2, "content", min=9, max=9, level=3,forms=c(4,5))
#For forms 1 and 2, specify word counts.
Ex2 <-  ata_constraint(Ex2, "word_count", min=30*10, max=80*10,forms=c(1,
2))
#For form 3, specify word counts.
Ex2 <-  ata_constraint(Ex2, "word_count", min=50*10, max=90*10,forms=3)
#For forms 4 and 5, specify word counts.
Ex2 <-  ata_constraint(Ex2, "word_count", min=40*10, max=100*10,forms=c(4
,5))
#For forms 1 and 2, specify time as seconds.
Ex2 <-  ata_constraint(Ex2, "time", min=250*10, max=400*10,forms=c(1,2))
#For form 3, specify time as seconds.
Ex2 <-  ata_constraint(Ex2, "time", min=200*10, max=300*10,forms=3)
#For forms 4 and 5, specify time as seconds.
Ex2 <-  ata_constraint(Ex2, "time", min=200*10, max=400*10,forms=c(4,5))
Ex2 <- ata_solve(Ex2, as.list=T) # Now, solve the ATA
Ex2$items #see selected items
plot(Ex2) # plotting information function
#End
```

```
#Example 3: Specifying different absolute amount of information for diffe
rent forms
```

```
# pulling five sets of item (5 test forms)
# for forms 1 and 2 target theta is -1 and target information is 5
# for form 3 target theta is 0 and target information is 10
# for forms 4 and 5 target theta is 1 and target information is 15
# Test length for forms 1 and 2 is 10 (2, 3, 5 items from Contents 1, 2 a
nd 3#respectively)
#Test length for form 3 is 15 (5, 6, 4 items from Contents 1, 2 and 3,
#respectively)
#Test length for forms 4 and 5 is 20 (4, 7, 9 items from Contents 1, 2 an
d 3, #respectively)
#For forms 1 and 2, average word count across the items in the forms is
#between 30 and 80
#For form 3, average word count across the items in the forms is between
50
#and 90
#For forms 4 and 5, average word count across the items in the forms is
#between 20 and 100
#For forms 1 and 2 average time to solve the item is between 200 and 250
#seconds
#For form 3 average time to solve the item is between 200 and 300 seconds
#For forms 4 and 5 average time to solve the item is between 200 and 400
theta_target1=-1 #theta point (or it can be interval) where you want the
#absolute information.
theta_target2=0 #theta point (or it can be interval) where you want the
# absolute information.
theta_target3=1 #theta point (or it can be interval) where you want the a
bsolute information.
tif_target1= 5 #The amount of information for forms 1 and 2. Change
#accordingly!
tif_target2= 10 #The amount of information for form 3. Change accordingly
!
tif_target3= 15 #The amount of information for forms 4 and 5. Change
#accordingly!
Ex3 <- ata(items, 5, #we are building 5 forms at the same time. Change
#accordingly!
           max_use=1) #we don't want item overlapping. Change accordingly
!
#Specify ATA for forms 1&2
Ex3 <- ata_obj_absolute(Ex3, theta_target1, tif_target1, forms = c(1,2))
#Specify ATA for forms 3
Ex3 <- ata_obj_absolute(Ex3, theta_target2, tif_target2, forms = 3)
#Specify ATA for forms 4&5
Ex3 <- ata_obj_absolute(Ex3, theta_target3, tif_target3, forms = c(4,5))
Ex3 <- ata_constraint(Ex3,1, min=10, max=10, forms=c(1,2)) #Test length f
orms #1&2
Ex3 <- ata_constraint(Ex3,1,  min=15, max=15, forms=3) #Test length forms
3
Ex3 <- ata_constraint(Ex3,1,  min=20, max=20, forms=c(4,5)) #Test length
#forms 4&5
#For forms 1 and 2, specify content distributions for content 1, 2, and 3
,
# respectively.
Ex3 <- ata_constraint(Ex3, "content", min=2, max=2, level=1,forms=c(1,2))
```

```
Ex3 <- ata_constraint(Ex3,"content", min=3, max=3, level=2, forms=c(1,2))
Ex3 <- ata_constraint(Ex3, "content", min=5, max=5, level=3,forms=c(1,2))
#For form 3, specify content distributions for content 1, 2, and 3,
#respectively.
Ex3 <- ata_constraint(Ex3, "content", min=5, max=5, level=1,forms = 3)
Ex3 <- ata_constraint(Ex3,"content", min=6, max=6, level=2, forms = 3)
Ex3 <- ata_constraint(Ex3, "content", min=4, max=4, level=3,forms = 3)
#For forms 4 and 5, specify content distributions for content 1, 2, and 3
,
#respectively.
Ex3 <- ata_constraint(Ex3, "content", min=4, max=4, level=1,forms=c(4,5))
Ex3 <- ata_constraint(Ex3,"content", min=7, max=7, level=2, forms=c(4,5))
Ex3 <- ata_constraint(Ex3, "content", min=9, max=9, level=3,forms=c(4,5))
#For forms 1 and 2, specify word counts.
Ex3 <-  ata_constraint(Ex3, "word_count", min=30*10, max=80*10,forms=c(1,
2))
#For form 3, specify word counts.
Ex3 <-  ata_constraint(Ex3, "word_count", min=50*10, max=90*10,forms=3)
#For forms 4 and 5, specify word counts.
Ex3 <-  ata_constraint(Ex3, "word_count", min=40*10, max=100*10,forms=c(4
,5))
#For forms 1 and 2, specify time as seconds.
Ex3 <-  ata_constraint(Ex3, "time", min=250*10, max=400*10,forms=c(1,2))
#For form 3, specify time as seconds.
Ex3 <-  ata_constraint(Ex3, "time", min=200*10, max=300*10,forms=3)
#For forms 4 and 5, specify time as seconds.
Ex3 <-  ata_constraint(Ex3, "time", min=200*10, max=400*10,forms=c(4,5))
Ex3 <- ata_solve(Ex3, as.list=T) #Now, let's solve the ATA!
Ex3$items #see selected items
plot(Ex3) # plotting information function

# END OF THE CODE #
```

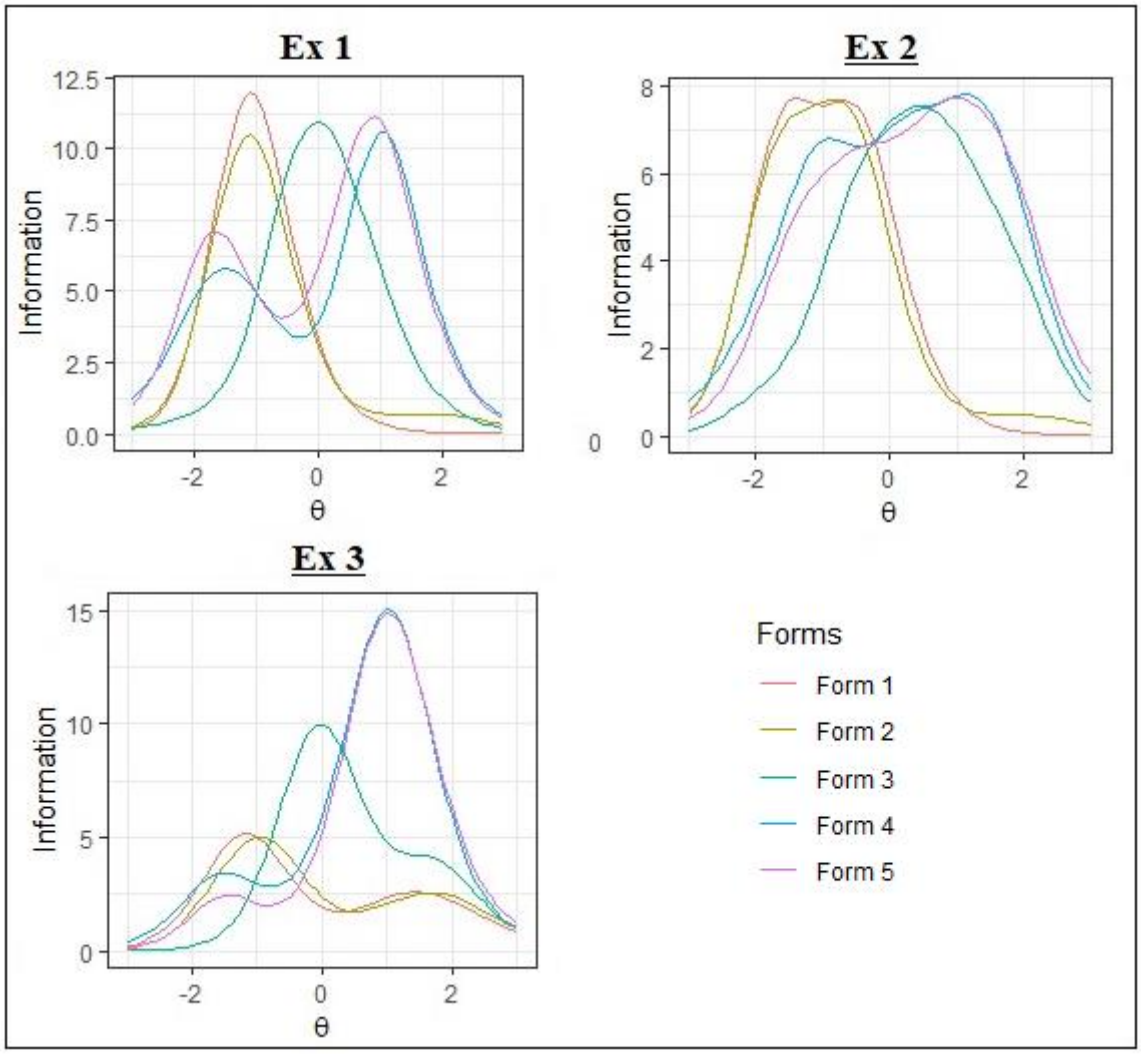

Figure A. 1. Plots for the solutions in Examples from 1 to 3.