GDINA R Package Handout for Cognitive Diagnosis Modeling: A General Framework Approach and Its Implementation in R

Jimmy de la Torre

The University of Hong Kong

Kevin Carl Santos

The University of Hong Kong

Yan Sun

Rutgers, The State University of New Jersey

Faculty of Education
The University of Hong Kong
December 1, 2018

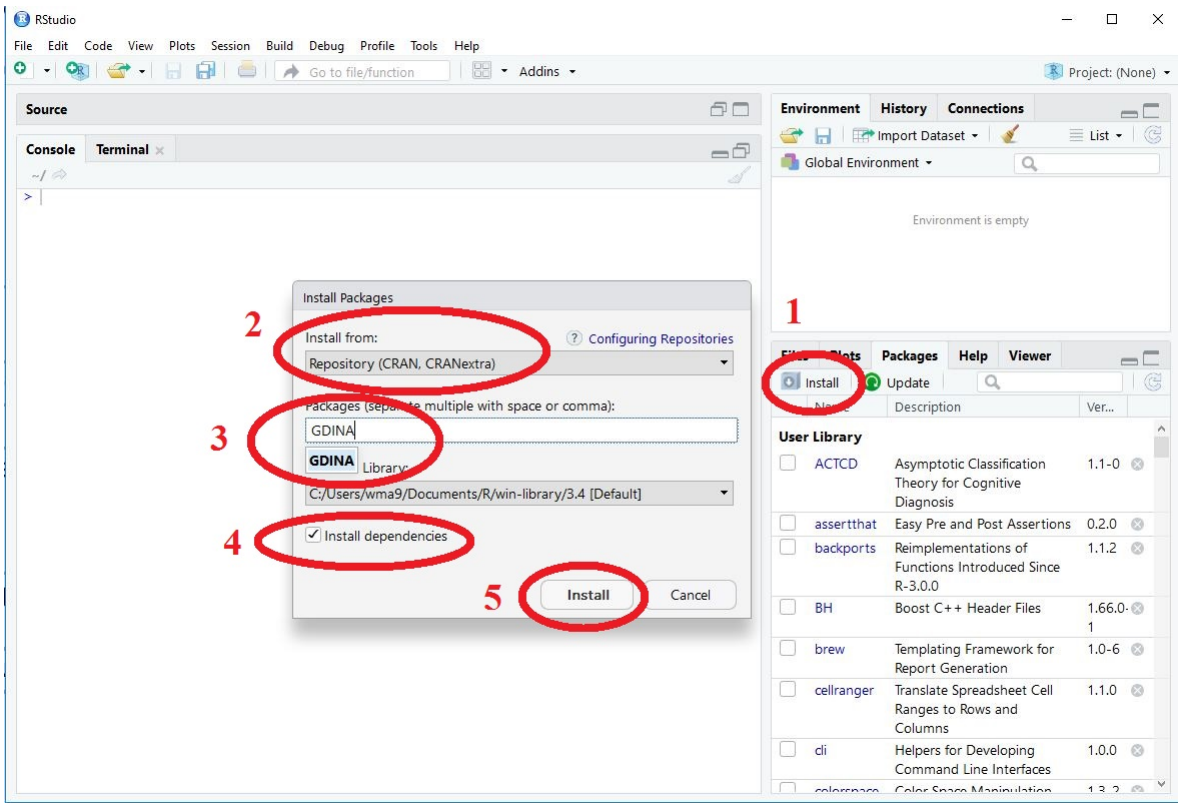
Contents

1 Installation

Please download and install the latest R software (R 3.5.1) from https://cran.r-project.org/.

In addition, please download and install the latest Rstudio, which is a free graphic user interface for R, and will be used in this training session. It can be downloaded from https://www.rstudio.com

After you installed R and Rstudio, open **Rstudio** and install the `GDINA` R package as follows:



After you see `package GDINA successfully unpacked and MD5 sums checked`, use `library` to load it:

Console Terminal

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/alabama_2015.3-1.zip'
Content type 'application/zip' length 72742 bytes (71 KB)
downloaded 71 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/ggplot2_2.2.1.zip'
Content type 'application/zip' length 3157026 bytes (3.0 MB)
downloaded 3.0 MB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/GDINA_2.1.15.zip'
Content type 'application/zip' length 1664113 bytes (1.6 MB)
downloaded 1.6 MB

package ‘alabama’ successfully unpacked and MD5 sums checked
package ‘ggplot2’ successfully unpacked and MD5 sums checked
package ‘GDINA’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Wma9\AppData\Local\Temp\RtmpAJjQ3r\downloaded_packages
> library(GDINA)
==============================================
GDINA Package for Cognitive Diagnosis Modeling
        Version 2.1.15 (2018-6-6)
==============================================
```

As shown below, run library(GDINA)

if you see the same output, the package is successfully installed.

2 About R

R is one of the most popular programming language (No. 7) according to IEEE Spectrum's interactive ranking.

Currently, the CRAN package repository features more than 12,000 available packages.

Here are some useful resources for R learners:

"Official" R introduction - https://cran.r-project.org/doc/manuals/R-intro.pdf

R for data science - http://r4ds.had.co.nz/

Advanced R - http://adv-r.had.co.nz/

3 Features of the GDINA Package

- Estimating G-DINA model and a variety of widely-used models subsumed by the G-DINA model, including the DINA model, DINO model, *additive*-CDM (*A*-CDM), linear logistic model (LLM), reduced reparametrized unified model (RRUM), multiple-strategy DINA model for dichotomous responses
- Estimating models within the G-DINA model framework using user-specified design matrix and link functions
- Estimating *Bugs*-DINA, DINO and G-DINA models for dichotomous responses
- Estimating sequential G-DINA model for ordinal and nominal responses
- Estimating diagnostic tree model for polytomous response
- Modelling independent, saturated, higher-order, loglinear smoothed, and structured joint attribute distribution
- Accommodating multiple-group model analysis
- Imposing monotonic constrained success probabilities
- Accommodating binary and polytomous attributes
- Validating Q-matrix under the general model framework
- Evaluating absolute and relative item and model fit
- Comparing models at the test and item levels
- Detecting differential item functioning using Wald and likelihood ratio test
- Simulating data based on all aforementioned CDMs
- Providing graphical user interface for users less familiar with R

4 Model Estimation

`data1` and `data2` are two data sets, each consisting of responses of 837 individuals to 15 items generated based on a proportional reasoning assessment. `Q1` and `Q2` are the associated Q-matrices, each involving 3 attributes.

In this training session, `data1` and `Q1` will be used for illustration, and `data2` and `Q2` will be used for exercises. Please download all four files as well as the R_code.R file, to a local folder.

Notes

To conduct CDM anlyses, you need to have at least
(1) a $N \times J$ response matrix (Missing values allowed)
(2) a $J \times K$ Q-matrix

Now please open your Rstudio. From your Rstudio, please open Handout_GDINA_R_code.R file [File → Open File... → Select Handout_GDINA_R_code.R]. Before fitting the CDMs, we need to specify working directory as follows [Sessions → Set Working Directory → Choose Directory... → Select the folder containing data, Q and Handout_GDINA_R_code.R files]:

After setting the working directory, we can use the code below to read data and Q-matrix into R.

```
# Load the GDINA package
library(GDINA)

## Warning:  package 'GDINA' was built under R version 3.4.4
## ==============================================
## GDINA Package for Cognitive Diagnosis Modeling
##        Version 2.1.15 (2018-6-6)
## ==============================================

# data
data1 <- read.table(file = "data1.dat", header = TRUE)
head(data1)

##   Item1 Item2 Item3 Item4 Item5 Item6 Item7 Item8 Item9 Item10 Item11
## 1     1     1     1     1     1     0     1     1     0      1      1
## 2     1     1     1     1     0     0     0     0     1      0      1
## 3     1     0     1     0     1     1     1     1     1      0      1
## 4     1     0     1     0     1     0     0     0     1      1      1
## 5     1     0     0     0     0     0     0     0     0      0      1
## 6     1     0     1     1     1     1     1     1     0      1      1
##   Item12 Item13 Item14 Item15
## 1      1      0      1      0
## 2      1      0      0      0
```

```
## 3       0       0       0       0
## 4       0       0       1       0
## 5       0       0       1       1
## 6       0       1       0       0

# Q-matrix
Q1 <- read.table(file = "Q1.txt")
Q1

##    V1 V2 V3
## 1   1  0  0
## 2   0  0  1
## 3   0  1  0
## 4   0  0  1
## 5   1  1  0
## 6   1  1  0
## 7   1  1  0
## 8   0  1  1
## 9   0  1  1
## 10  0  1  1
## 11  0  1  1
## 12  1  0  1
## 13  1  0  1
## 14  1  1  1
## 15  1  1  1
```

In R_code.R file, you can select the corresponding code and click "Run" as shown below:

4.1 Estimation of the G-DINA Model

The `GDINA` function is the main function of the package. It takes various arguments as in:

Code

```
GDINA(dat, Q, model = "GDINA", ...)
```

To estimate G-DINA model, call `GDINA` function and specify the data and Q-matrix as the first two arguments.

```
# Fit the data using G-DINA model
fit1 <- GDINA(dat = data1, Q = Q1)
```

Returned `fit1` contains (almost) all estimation results. It is called an object of class `GDINA`, and we can apply various `methods` to it. To print some general model estimation information, type `fit1` in Rstudio console:

```
fit1

## Call:
## GDINA(dat = data1, Q = Q1)
##
##   GDINA version 2.1.15 (2018-6-6)
## ================================================
## Data
## ------------------------------------------------
## # of individuals    groups    items
##              837         1       15
## ================================================
## Model
## ------------------------------------------------
## Fitted model(s)       = GDINA
## Attribute structure   = saturated
## Attribute level       = Dichotomous
## ================================================
## Estimation
## ------------------------------------------------
## Number of iterations  = 288
## For the final iteration:
##   Max abs change in item success prob. = 0.0001
##   Max abs change in mixing proportions = 0.0000
##   Change in -2 log-likelihood          = 0.0001
## Time used             = 1.468 secs
```

As shown above, some information about the data, fitted model and calibration was printed. Such information can be used to check the model specification and evaluate convergence (By default, the maximum iterations allowed is 2000).
`summary` is another function that can be applied to objects of class `GDINA`. Specify `fit1` as the input:

```
summary(fit1)

##
## Test Fit Statistics
##
## Loglik = -7431.45
## AIC    = 14996.91  | penalty   = 134
## BIC    = 15313.81  | penalty   = 450.90
## # par  = 67
##
## Attribute Prevalence
##
##    Level0 Level1
## A1 0.3688 0.6312
## A2 0.4601 0.5399
## A3 0.2875 0.7125
```

As shown above, the `summary` function prints some test fit statistics, like AIC and BIC, the number of parameters and attribute prevalence. The attribute prevalence gives the proportion of individuals who master (or do not master) each attribute, labelled as "Level1" (or "Level0").
To extract item parameters, we can use `coef` function, as in

```
coef(fit1)

## $`Item 1`
##   P(0)   P(1)
## 0.9291 0.9808
##
## $`Item 2`
##   P(0)   P(1)
## 0.2613 0.8201
##
## $`Item 3`
##   P(0)   P(1)
## 0.5381 0.8249
##
## $`Item 4`
##   P(0)   P(1)
## 0.4810 0.8707
##
## $`Item 5`
##  P(00)  P(10)  P(01)  P(11)
## 0.2881 0.7283 0.5432 0.8921
##
## $`Item 6`
##  P(00)  P(10)  P(01)  P(11)
## 0.1571 0.4414 0.0464 0.7568
##
## $`Item 7`
##  P(00)  P(10)  P(01)  P(11)
## 0.0001 0.5172 0.5316 0.7762
##
## $`Item 8`
##  P(00)  P(10)  P(01)  P(11)
## 0.0395 0.6688 0.3473 0.8037
##
## $`Item 9`
##  P(00)  P(10)  P(01)  P(11)
## 0.0937 0.4075 0.3453 0.6824
##
## $`Item 10`
##  P(00)  P(10)  P(01)  P(11)
## 0.4337 0.7113 0.3595 0.7748
##
## $`Item 11`
```

```
##  P(00)  P(10)  P(01)  P(11)
## 0.4877 0.6204 0.7261 0.9061
##
## $`Item 12`
##  P(00)  P(10)  P(01)  P(11)
## 0.1883 0.3434 0.3564 0.6847
##
## $`Item 13`
##  P(00)  P(10)  P(01)  P(11)
## 0.1980 0.1912 0.3077 0.5909
##
## $`Item 14`
## P(000) P(100) P(010) P(001) P(110) P(101) P(011) P(111)
## 0.0764 0.9999 0.0241 0.2360 0.6121 0.4106 0.0872 0.6505
##
## $`Item 15`
## P(000) P(100) P(010) P(001) P(110) P(101) P(011) P(111)
## 0.0989 0.7047 0.0072 0.3002 0.3321 0.3307 0.2473 0.6451
```

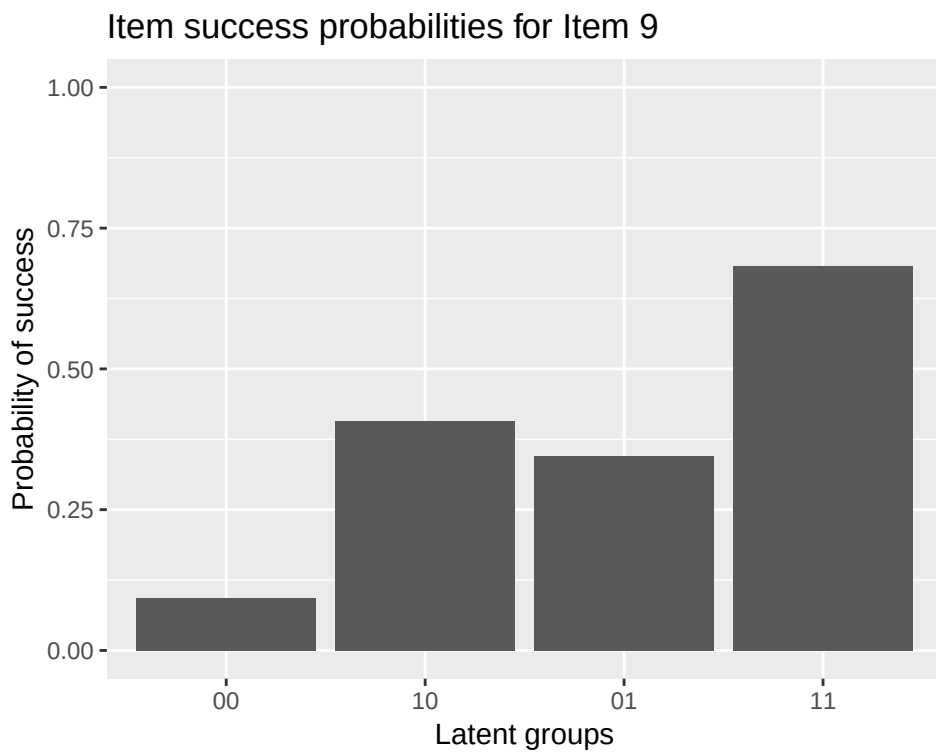
The returned output by `coef` (when we only specify the first argument) is item success probabilities for each latent group (reduced latent classes). `coef` has a few arguments including `what`, `withSE` and `SE.type`. `what` specifies what to show; It can be `"itemprob"` for item success probabilities, `"gs"` for guessing and slip parameters, `"delta"` for delta parameters, `"rrum"` for RRUM parameters when items are estimated using RRUM, `"LCprob"` for item success probabilities for all latent classes, and `"lambda"` for structural parameters of joint attribute distribution (such as higher-order structural parameters).
Set `withSE=TRUE` if you want to estimate standard errors. Note that standard errors can only be calculated for item success probabilities, guessing and slip parameters, and delta parameters. For other parameters, you can use function `bootSE` to calculate standard errors using bootstrap procedures. `SE.type` can be 1, 2 or 3, indicating outer product of gradient (OPG) estimates based on itemwise, incomplete or complete information matrix. Default is 2. Currently, the OPG method based on the complete information matrix assumes that all latent classes are identifiable and is only available for saturated joint attribute distribution.

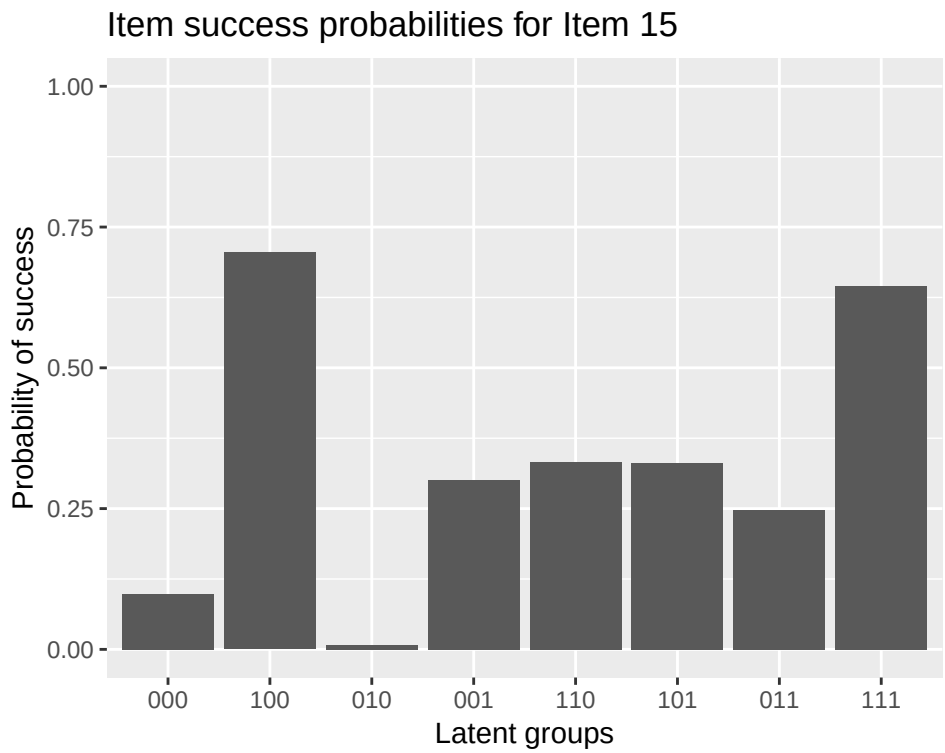
```r
# item success probabilities for item 15
coef(fit1, withSE = TRUE)[[15]]

##      P(000) P(100) P(010) P(001) P(110) P(101) P(011) P(111)
## Est. 0.0989 0.7047 0.0072 0.3002 0.3321 0.3307 0.2473 0.6451
## S.E. 0.0856 0.1520 0.1438 0.0548 0.0602 0.0640 0.0939 0.0452

# delta parameters for item 15
coef(fit1, what = "delta", withSE = TRUE)[[15]]

##          d0     d1      d2     d3     d12     d13    d23   d123
## Est. 0.0989 0.6058 -0.0917 0.2013 -0.2809 -0.5753 0.0388 0.6482
## S.E. 0.0856 0.1835  0.1741 0.1172  0.2581  0.2232 0.2411 0.3334

# population proportions when saturated model is used for joint attribute
# distribution
coef(fit1, what = "lambda")
```

```
## p(000) p(100) p(010) p(001) p(110) p(101) p(011) p(111)
## 0.0615 0.0314 0.0374 0.1860 0.1571 0.1811 0.0838 0.2616
```

To plot the probabilities of success for each item $P(X_{ij} = 1|\boldsymbol{\alpha}^*_{lj})$, we can use `plot`:

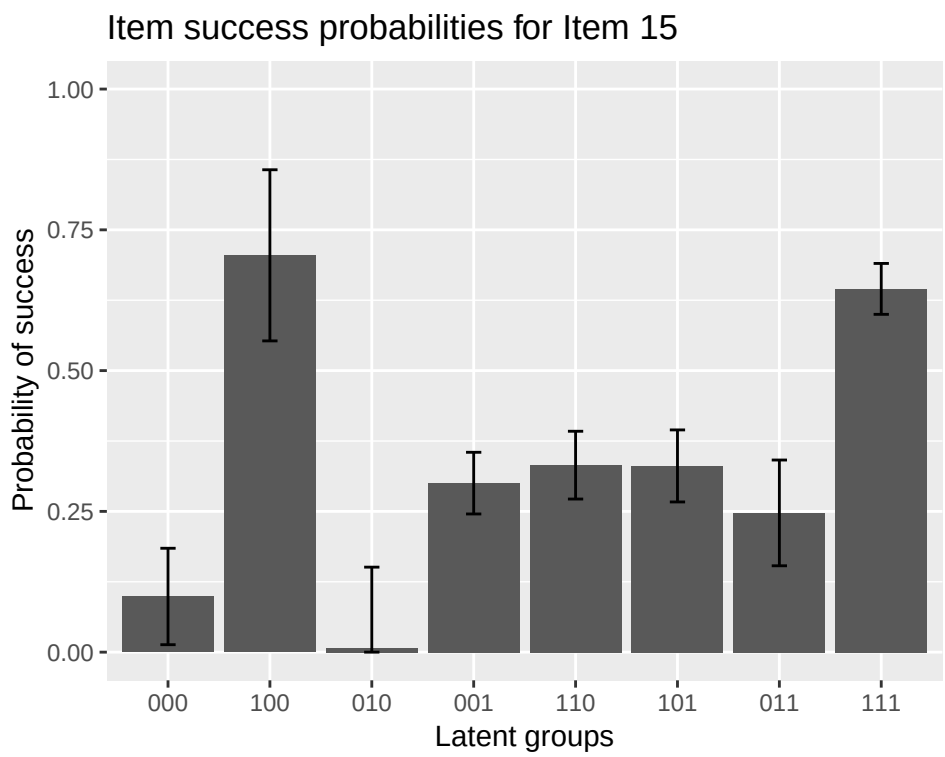
```
plot(fit1, what = "IRF", item = c(9, 15))
```





If we want to add an error bar to the IRF plot of item 15, using the following code:

```
plot(fit1, what = "IRF", item = 15, withSE = TRUE)
```



Individual attribute patterns can be estimated using EAP, MLE and MAP. To obtain attribute estimates, use the function `personparm`:

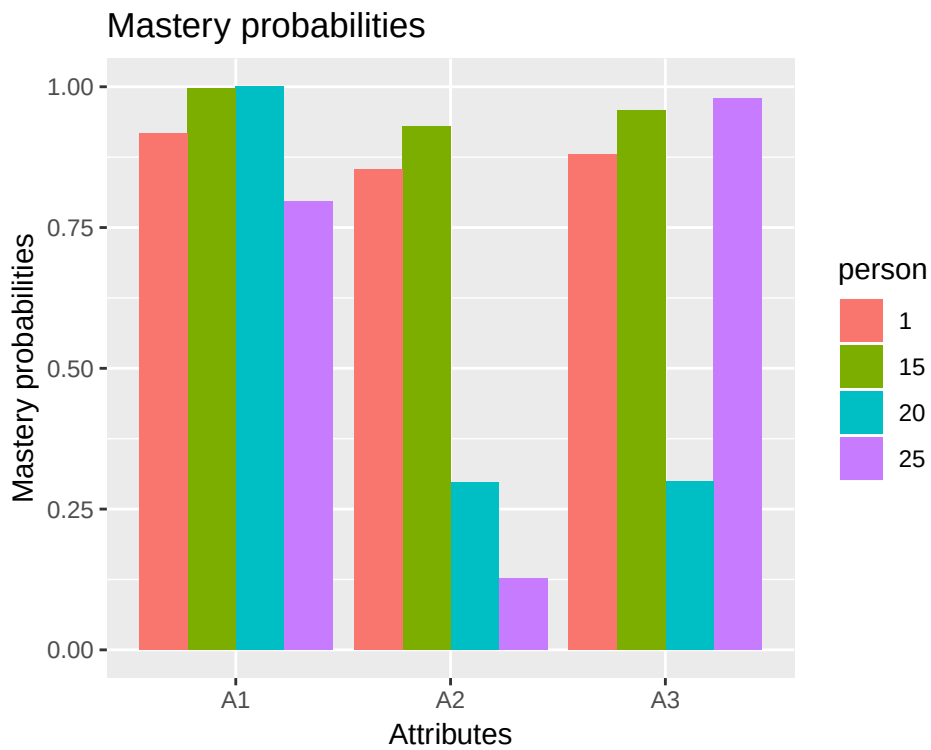
```r
head(personparm(fit1))

##      A1 A2 A3
## [1,]  1  1  1
## [2,]  0  0  1
## [3,]  1  1  0
## [4,]  1  1  0
## [5,]  1  0  0
## [6,]  1  1  0
```

`personparm(fit1)` will return a matrix. We use `head` function to display the first six individuals' estimates. Like `coef` function, `personparm` also has an argument called `what` so that we can specify the type of person parameters that we want to extract. If `what` is not specified as above, EAP estimates of attribute patterns will be returned.

```r
# MAP estimates
head(personparm(fit1, what = "MAP"))

##   A1 A2 A3 multimodes
## 1  1  1  1      FALSE
## 2  0  0  1      FALSE
## 3  1  1  0      FALSE
## 4  1  1  0      FALSE
## 5  1  0  0      FALSE
## 6  1  1  0      FALSE

# MLE estimates
head(personparm(fit1, what = "MLE"))

##   A1 A2 A3 multimodes
## 1  1  1  1      FALSE
## 2  0  0  1      FALSE
## 3  1  1  0      FALSE
## 4  1  1  0      FALSE
## 5  1  0  0      FALSE
## 6  1  1  0      FALSE

# probability of mastering each attribute
head(personparm(fit1, what = "mp"))

##          A1     A2     A3
## [1,] 0.9174 0.8540 0.8797
## [2,] 0.0889 0.0709 0.9849
## [3,] 0.9714 0.9942 0.0348
## [4,] 0.8143 0.7877 0.1100
## [5,] 0.7215 0.0048 0.1471
## [6,] 0.9688 0.9516 0.3836
```

You can also plot probabilities of mastering each attribute for individuals 1, 15, 20 and 25:

```r
plot(fit1, what = "mp", person = c(1, 15, 20, 25))
```



Apart from item and person parameters, many other components of G-DINA estimates may also be of interest. Several methods have been defined:

Methods for objects of class `GDINA`

- AIC: calculate AIC
- BIC: calculate BIC
- logLik: calculate log-likelihood
- deviance: calculate deviance
- npar: calculate the number of parameters
- indlogLik: extract log-likelihood for each individual
- indlogPost: extract log posterior for each individual

To extract other elements, we should use `extract` function. Its first input should be an object returned from GDINA function (or other functions), and its second input (i.e., `what`) specifies what to extract. See its help page ?`extract` for a complete list of elements that can be extracted.
For example, to calculate the discrimination indices, the following code can be used:

```r
extract(fit1, "discrim")

##           P(1)-P(0)          GDI
```

```
## Item 1  0.05173884 0.000623121
## Item 2  0.55878758 0.063954921
## Item 3  0.28680976 0.020434101
## Item 4  0.38972560 0.031109881
## Item 5  0.60408134 0.059419199
## Item 6  0.59975346 0.080395279
## Item 7  0.77614961 0.093990302
## Item 8  0.76417095 0.064163657
## Item 9  0.58865665 0.034829670
## Item 10 0.34105321 0.036291009
## Item 11 0.41839397 0.018267132
## Item 12 0.49641286 0.034608409
## Item 13 0.39284305 0.030681224
## Item 14 0.57406061 0.058915105
## Item 15 0.54624288 0.035189771
```

To calculate the proportion of individuals in each latent class, we can use:

```
extract(fit1, "posterior.prob")

##             000        100        010      001       110       101
## [1,] 0.06151017 0.03144658 0.03741109 0.186019 0.1570853 0.1811396
##             011       111
## [1,] 0.08382109 0.2615672
```

4.2 Estimation of Reduced CDMs

Change the argument `model` to estimate various CDMs. For example, to estimate DINA model, specify `model = "DINA"`:

```
fit2 <- GDINA(dat = data1, Q = Q1, model = "DINA")
```

```
fit2

## Call:
## GDINA(dat = data1, Q = Q1, model = "DINA")
##
##   GDINA version 2.1.15 (2018-6-6)
## ================================================
## Data
## ------------------------------------------------
## # of individuals    groups    items
##              837         1       15
## ================================================
## Model
## ------------------------------------------------
```

```
## Fitted model(s)        = DINA
## Attribute structure    = saturated
## Attribute level        = Dichotomous
## ================================================
## Estimation
## ------------------------------------------------
## Number of iterations   = 111
## For the final iteration:
##   Max abs change in item success prob. = 0.0001
##   Max abs change in mixing proportions = 0.0001
##   Change in -2 log-likelihood          = 0.0005
## Time used              = 0.7879 secs
```

```
# guessing and slip
coef(fit2, what = "gs", withSE = TRUE)
```

```
##          guessing   slip SE[guessing] SE[slip]
## Item 1     0.9144 0.0165       0.0214   0.0076
## Item 2     0.0778 0.2405       0.0761   0.0208
## Item 3     0.5489 0.2287       0.0371   0.0221
## Item 4     0.3659 0.1738       0.0709   0.0178
## Item 5     0.4125 0.0931       0.0294   0.0197
## Item 6     0.1995 0.3007       0.0259   0.0274
## Item 7     0.2202 0.2342       0.0272   0.0264
## Item 8     0.2991 0.2612       0.0317   0.0266
## Item 9     0.2524 0.3843       0.0297   0.0284
## Item 10    0.4350 0.3023       0.0318   0.0264
## Item 11    0.6192 0.1491       0.0298   0.0215
## Item 12    0.2897 0.3783       0.0352   0.0288
## Item 13    0.2172 0.4680       0.0327   0.0288
## Item 14    0.2767 0.3723       0.0250   0.0307
## Item 15    0.2733 0.4671       0.0239   0.0309
```

By specifying `model="RRUM"`, you can estimate RRUM.

```
# Fit RRUM to the data
fit3 <- GDINA(dat = data1, Q = Q1, model = "RRUM")
```

Delta parameters in the G-DINA model equation can be printed. Note that in de la Torre (2011), different parameters were used for different link functions. In the `GDINA` package, however, `delta` is used for all link functions.

```
# print delta parameters
coef(fit3, what = "delta")
```

```
## $`Item 1`
##      d0      d1
## -0.0696  0.0489
```

```
##
## $`Item 2`
##      d0      d1
## -1.3103  1.1263
##
## $`Item 3`
##      d0      d1
## -0.6170  0.3931
##
## $`Item 4`
##      d0      d1
## -0.7270  0.5997
##
## $`Item 5`
##      d0      d1      d2
## -1.1157  0.6901  0.3381
##
## $`Item 6`
##      d0      d1      d2
## -2.2172  1.4583  0.4423
##
## $`Item 7`
##      d0      d1      d2
## -1.9300  0.8115  0.8962
##
## $`Item 8`
##      d0      d1      d2
## -1.6208  1.1678  0.2353
##
## $`Item 9`
##      d0      d1      d2
## -1.7706  0.8433  0.5363
##
## $`Item 10`
##      d0      d1      d2
## -0.9833  0.6704 -0.0170
##
## $`Item 11`
##      d0      d1      d2
## -0.6944  0.2406  0.3473
##
## $`Item 12`
##      d0      d1      d2
## -1.6794  0.6646  0.6528
##
## $`Item 13`
##      d0      d1      d2
```

```
## -1.9916  0.5442  0.9139
##
## $`Item 14`
##      d0      d1      d2      d3
## -1.9565  1.3080  0.2054 -0.0224
##
## $`Item 15`
##      d0      d1      d2      d3
## -1.9267  0.6246  0.3541  0.4253
```

By specifying `what = rrum`, we can print π^* and r_k, which are the parameters based on the original parameterization of RRUM.

```r
# pi and r - original parameterization for RRUM
coef(fit3, what = "rrum")
```

```
## $`Item 1`
##    pi*     r1
## 0.9794 0.9523
##
## $`Item 2`
##    pi*     r1
## 0.8320 0.3242
##
## $`Item 3`
##    pi*     r1
## 0.7994 0.6750
##
## $`Item 4`
##    pi*     r1
## 0.8805 0.5490
##
## $`Item 5`
##    pi*     r1     r2
## 0.9162 0.5015 0.7131
##
## $`Item 6`
##    pi*     r1     r2
## 0.7286 0.2326 0.6426
##
## $`Item 7`
##    pi*     r1     r2
## 0.8007 0.4442 0.4081
##
## $`Item 8`
##    pi*     r1     r2
## 0.8043 0.3111 0.7903
##
```

```
## $`Item 9`
##    pi*     r1     r2
## 0.6764 0.4303 0.5849
##
## $`Item 10`
##    pi*     r1     r2
## 0.7191 0.5115 1.0171
##
## $`Item 11`
##    pi*     r1     r2
## 0.8989 0.7862 0.7066
##
## $`Item 12`
##    pi*     r1     r2
## 0.6962 0.5145 0.5206
##
## $`Item 13`
##    pi*     r1     r2
## 0.5865 0.5803 0.4010
##
## $`Item 14`
##    pi*     r1     r2     r3
## 0.6278 0.2704 0.8143 1.0227
##
## $`Item 15`
##    pi*     r1     r2     r3
## 0.5929 0.5355 0.7018 0.6536
```

It is also possible to specify different CDMs to different items in a single test.

```
# Fit different CDMs to different items
models <- c(rep("GDINA",4),"LLM","ACDM","GDINA",
            "LLM","RRUM","ACDM","RRUM","ACDM","DINA","RRUM","RRUM")
```

```
fit5 <- GDINA(dat = data1, Q = Q1, model = models)
```

```
fit5

## Call:
## GDINA(dat = data1, Q = Q1, model = models)
##
##   GDINA version 2.1.15 (2018-6-6)
## ================================================
## Data
## ------------------------------------------------
## # of individuals    groups    items
```

```
##              837         1       15
## ================================================
## Model
## ------------------------------------------------
## Fitted model(s)        = GDINA LLM ACDM RRUM DINA
## Attribute structure    = saturated
## Attribute level        = Dichotomous
## ================================================
## Estimation
## ------------------------------------------------
## Number of iterations  = 80
## For the final iteration:
##   Max abs change in item success prob. = 0.0001
##   Max abs change in mixing proportions = 0.0001
##   Change in -2 log-likelihood          = 0.0008
## Time used              = 0.9525 secs
```

Notes

- guessing and slip parameters can always be calculated regardless of the CDMs used. It calculates P(0) and 1-P(1), which are the guessing and slip parameters, respectively, when the model is DINA and DINO.
- delta parameters are also available for all models.

4.3 Modeling Joint Attribute Distribution Using Higher-order Models

Three IRT models are available for the higher-order attribute structure: Rasch model, one parameter logistic model (1PL) and two parameter logistic model (2PL). The higher-order structure can be used with any CDMs.

Rasch model

the probability of mastering attribute k for individual i is defined as

$$P(\alpha_k = 1|\theta_i, \lambda_{0k}) = \frac{\exp(\theta_i + \lambda_{0k})}{1 + \exp(\theta_i + \lambda_{0k})}.$$

1PL model

the probability of mastering attribute k for individual i is defined as

$$P(\alpha_k = 1|\theta_i, \lambda_{0k}, \lambda_1) = \frac{\exp(\lambda_1\theta_i + \lambda_{0k})}{1 + \exp(\lambda_1\theta_i + \lambda_{0k})}.$$

2PL model

the probability of mastering attribute k for individual i is defined as

$$P(\alpha_k = 1|\theta_i, \lambda_{0k}, \lambda_{1k}) = \frac{\exp(\lambda_{1k}\theta_i + \lambda_{0k})}{1 + \exp(\lambda_{1k}\theta_i + \lambda_{0k})},$$

where θ_i is the ability of examinee i. λ_{0k} and λ_{1k} are the intercept and slope parameters for attribute k, respectively. In the Rasch model, $\lambda_{1k} = 1 \forall k$; whereas in the 1PL model, a common slope parameter λ_1 is estimated. To specify a higher-order structure for attributes, use the following code. By default, the Rasch model is used as the higher-order IRT model.

```
# higher-order G-DINA model (By default: Rasch model)
HOfit1 <- GDINA(dat = data1, Q = Q1, model = "GDINA", att.dist = "higher.order")
```

```
# print higher-order structural parameters
coef(HOfit1, "lambda")

##    slope intercept
## A1     1    0.4761
## A2     1    0.1751
## A3     1    0.0802

# print higher-order person ability EAP estimates
head(personparm(HOfit1, "HO"))

##       EAP     SE
## 1  0.5028 0.8801
## 2 -0.3696 0.8325
## 3  0.0570 0.8268
## 4 -0.3793 0.8763
## 5 -0.7743 0.8436
## 6  0.2552 0.8831
```

To specify other arguments to control the specification of the higher-order models, use `higher.order` argument. For example, to use 2PL model as the higher-order IRT model:

```
HOfit2 <- GDINA(dat = data1, Q = Q1, model = "GDINA",
                att.dist = "higher.order",
                higher.order = list(model = "2PL"))
```

The argument `higher.order` can take many options, including

`higher.order` argument

model - Can be "2PL", "1PL" or "Rasch".
nquad - a scalar specifying the number of integral nodes. Default = 25.
SlopeRange - a vector of length two specifying the range of slope parameters. Default = [0.1, 5].
InterceptRange - a vector of length two specifying the range of intercept parameters. Default = [-4, 4].
SlopePrior - a vector of length two specifying the mean and variance of log(slope) parameters, which are assumed to be normally distributed. Default: mean = 0 and sd = 0.25.
InterceptPrior - a vector of length two specifying the mean and variance of intercept parameters, which are assumed to be normally distributed. Default: mean = 0 and sd = 1.
Prior - logical; indicating whether prior distributions should be imposed to slope and intercept parameters. Default is FALSE.

To better help users utilize the `GDINA` function, we provided many examples in its help page:

GDINA Examples

Example 1 - GDINA, DINA, DINO, ACDM, LLM and RRUM estimation and comparison
Example 2 - Monotonic constraints
Example 3a - Higher-order joint attribute distribution
Example 3b - loglinear smoothed joint attribute distribution
Example 3c - independent joint attribute distribution
Example 4 - structured joint attribute distribution
Example 5 - structured joint attribute distribution
Example 6 - user-specified initial values
Example 7 - Estimation without M-step (Fixed item parameters to estimate person ability)
Example 8 - polytomous attribute model
Example 9 - sequential model for polytomous responses
Example 10a- multiple-group models
Example 10b- multiple-group higher-order models
Example 11 - Bug DINO models
Example 12 - Bug DINA models
Example 13a- logit GDINA (loglinear CDM) using user specified design matrix and link function
Example 13b- RRUM using user specified design matrix and link function
Example 14 - Multiple-strategy DINA model

Exercise

Please use `data2` and `Q2` for the following exercises. Run the following code to read the response data and Q-matrix.

```
data2 <- read.table(file = "data2.dat",header = TRUE)
Q2 <- read.table(file = "Q2.txt")
```

Fit the G-DINA model to the data, then answer the following questions (Note: Fit all models without monotonicity constraints unless otherwise stated):

- For Item 15, the G-DINA model parameter estimates are P(100) = 0.4_57, P(101) = 0.6_96.
- The corresponding SEs for P(100) and P(101) are 0.0_37 and 0.0_30, respectively.
- The proportion of individuals having an attribute pattern of 111 in the population is estimated to be 0.3_12
- The proportion of individuals who master $\boldsymbol{\alpha}1$ in the population is estimated to be 0.80_7.
- Plot item success probabilities for Items 5, 8 and 15. Which item(s) appear to follow the DINO models?
- Find the EAP estimate of attribute pattern for the third individual.

Fit reduced models for the following questions:

- Fit LLM to the data: the number of parameters for LLM is 5_.
- Fit DINA to the data: guessing and slip parameter estimates of Item 10 are 0.3_16 and 0.2_32, respectively. The corresponding SEs are 0.0_79 and 0.0_12.
- Fit a Rasch higher-order DINA model and print the higher-order structural parameter estimates.

Click HERE for solutions.

5 Model Fit Evaluation

To evaluate the model data fit, we need to fit some CDMs to the data first. This time, we start from the DINO model:

```
# Load the GDINA package
library(GDINA)
data1 <- read.table(file = "data1.dat",header = TRUE)
# Q-matrix
Q1 <- read.table(file = "Q1.txt")
# Fit the DINO model
# verbose can be used to control what
# to be printed during the estimation
fit1 <- GDINA(dat = data1, Q = Q1, model = "DINO", verbose = 0)
```

`summary` and several other functions can be used to print some test level relative model-data fit statistics:

```
summary(fit1)

##
## Test Fit Statistics
##
## Loglik = -7524.80
## AIC    = 15123.60  | penalty   = 74
## BIC    = 15298.61  | penalty   = 249.00
## # par  = 37
##
## Attribute Prevalence
##
##    Level0 Level1
## A1 0.6419 0.3581
## A2 0.5344 0.4656
## A3 0.7551 0.2449

logLik(fit1)

## 'log Lik.' -7524.802 (df=37)

deviance(fit1)

## [1] 15049.6

AIC(fit1)

## [1] 15123.6

BIC(fit1)

## [1] 15298.61

npar(fit1)

## No. of total parameters = 37
## No. of item parameters = 30
## No. of population parameters = 7
```

To evaluate absolute fit, we need to use `itemfit` function with the objects returned from `GDINA` function as the (first) input.

Code

```
itemfit(GDINA.obj, ...)
```

```
ifit <- itemfit(fit1)
```

Print summary information about item fit statistics by typing `ifit` in the console:

```
ifit

## Summary of Item Fit Analysis
##
## Call:
## itemfit(GDINA.obj = fit1)
##
##                          mean[stats] max[stats] max[z.stats] p-value
## Proportion correct           0.0010     0.0035       0.2024  0.8396
## Transformed correlation      0.0387     0.1278       3.6900  0.0002
## Log odds ratio               0.1984     0.9414       3.4676  0.0005
##                          adj.p-value
## Proportion correct           1.0000
## Transformed correlation      0.0235
## Log odds ratio               0.0551
## Note: p-value and adj.p-value are associated with max[z.stats].
##       adj.p-values are based on the bonferroni method.
```

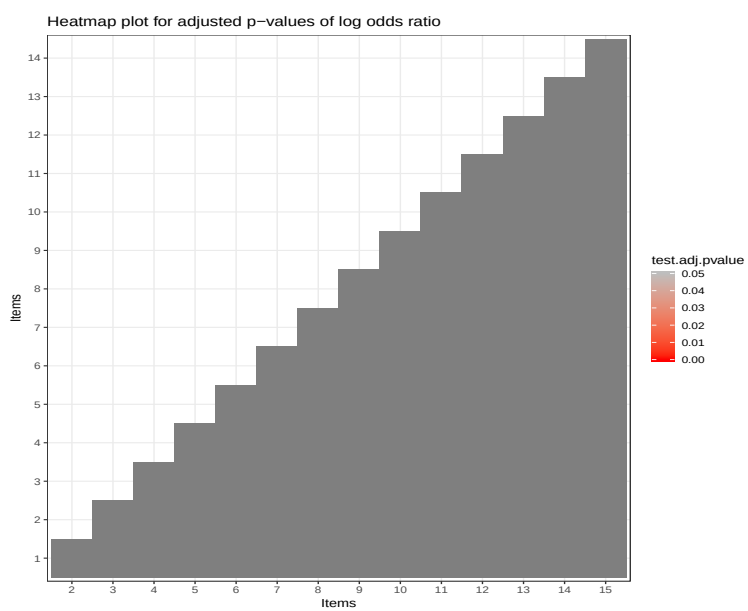
As shown above, the p-values and adjusted p-values for the maximum z-scores of test statistics were reported for the whole test.
Item-level fit information can be printed using `summary` function by specifying `ifit` as the argument as follows. The p-values and adjusted p-values (not applicable for proportions) for maximum z scores were reported for each item.

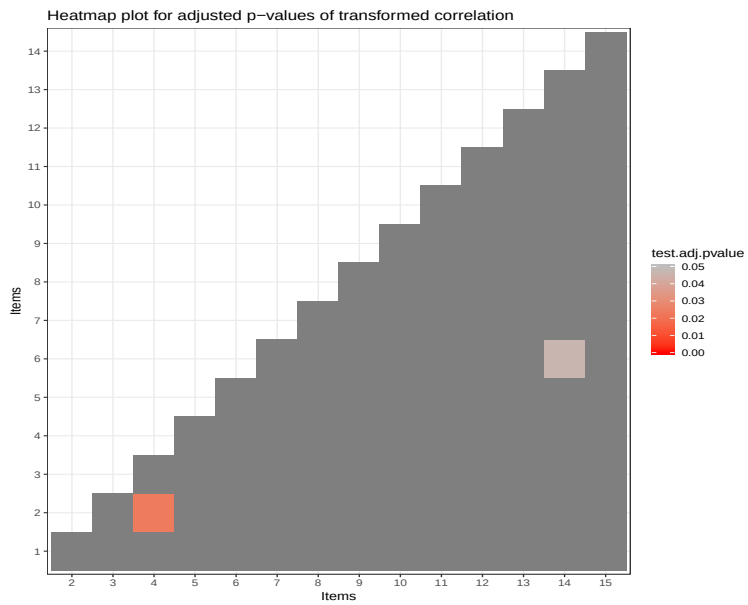
```
summary(ifit)

##
## Item-level fit statistics
##         z.prop pvalue[z.prop] max[z.r] pvalue.max[z.r] adj.pvalue.max[z.r]
## Item 1  0.0150         0.9880   2.7046          0.0068              0.0957
## Item 2  0.0627         0.9500   3.6900          0.0002              0.0031
## Item 3  0.0231         0.9816   2.7046          0.0068              0.0957
## Item 4  0.1167         0.9071   3.6900          0.0002              0.0031
## Item 5  0.0683         0.9455   2.9350          0.0033              0.0467
## Item 6  0.0885         0.9295   3.5147          0.0004              0.0062
## Item 7  0.1043         0.9170   2.4831          0.0130              0.1823
## Item 8  0.2024         0.8396   2.0510          0.0403              0.5638
```

```
## Item 9  0.0098         0.9921   2.0510          0.0403              0.5638
## Item 10 0.0490         0.9609   2.6244          0.0087              0.1215
## Item 11 0.0416         0.9668   2.7795          0.0054              0.0762
## Item 12 0.0409         0.9674   1.8239          0.0682              0.9542
## Item 13 0.0059         0.9953   1.8831          0.0597              0.8356
## Item 14 0.0588         0.9531   3.5147          0.0004              0.0062
## Item 15 0.0124         0.9901   2.2658          0.0235              0.3284
##         max[z.logOR] pvalue.max[z.logOR] adj.pvalue.max[z.logOR]
## Item 1        2.4235              0.0154                  0.2152
## Item 2        3.4532              0.0006                  0.0078
## Item 3        2.5897              0.0096                  0.1345
## Item 4        3.4532              0.0006                  0.0078
## Item 5        2.9651              0.0030                  0.0424
## Item 6        3.4676              0.0005                  0.0074
## Item 7        2.4810              0.0131                  0.1834
## Item 8        2.0383              0.0415                  0.5812
## Item 9        2.0767              0.0378                  0.5296
## Item 10       2.5897              0.0096                  0.1345
## Item 11       2.6774              0.0074                  0.1039
## Item 12       1.8413              0.0656                  0.9182
## Item 13       1.9222              0.0546                  0.7640
## Item 14       3.4676              0.0005                  0.0074
## Item 15       2.2431              0.0249                  0.3485
```

Fit information for item pairs can be presented using heatplot. Specifically, we can plot (adjusted) p values of fit statistics for all item pairs using a heatmap plot:

```
plot(ifit)
```





If we want to compare two nested models, we can use likelihood ratio test. AIC and BIC can also be used when two models are fitted to the same data. To compare whether the DINO model can fit the data as well as the G-DINA model, we use `anova` function with two model fit objects as the inputs:

```
# fit1 is based on the DINO model fit2 is based on the G-DINA model
fit2 <- GDINA(dat = data1, Q = Q1, model = "GDINA", verbose = 0)
anova(fit1, fit2)

##
## Information Criteria and Likelihood Ratio Test
##
##      #par   logLik Deviance      AIC      BIC chisq df p-value
## fit1   37 -7524.80 15049.60 15123.60 15298.61 186.7 30  <0.001
## fit2   67 -7431.45 14862.91 14996.91 15313.81
```

We can also compare the fit of various CDM combinations with that of G-DINA model:

```
# Fit different CDMs to different items
models <- c(rep("GDINA",4),"LLM","ACDM","GDINA",
            "LLM","RRUM","ACDM","RRUM","ACDM","DINA","RRUM","RRUM")
```

```
fit3 <- GDINA(dat = data1, Q = Q1, model = models)
```

```
anova(fit1, fit2, fit3)

##
## Information Criteria and Likelihood Ratio Test
##
##      #par   logLik Deviance      AIC      BIC chisq df p-value
## fit1   37 -7524.80 15049.60 15123.60 15298.61 186.7 30  <0.001
```

```
## fit2   67 -7431.45 14862.91 14996.91 15313.81
## fit3   50 -7443.62 14887.25 14987.25 15223.74 24.34 17    0.11
##
## Notes: In LR tests, models were tested against fit2
##        LR test(s) do NOT check whether models are nested or not.
```

Notes

Likelihood ratio test (LRT) can only be used when two models are nested, but anova function does NOT check whether two models are nested or not.
Null hypothesis of the LRT is the simpler model fits the data as well as the complex model.
LRT, AIC and BIC are all **relative** fit statistics.
`modelfit` is another function that can be used to check model-data fit.

Exercise

Fit each of the DINA, DINO, and G-DINA models to the data2 without monotonic constraints, then answer the following questions:

1. Based on the transformed correlation statistics, which model(s) cannot fit data well?
2. Based on the heatmap plot, is there misfit between item pairs for DINO model?
3. Based on AIC, which model is preferred (i.e., has the smallest value of AIC)
4. Based on BIC, which model is preferred (i.e., has the smallest value of BIC)?
5. Based on likelihood ratio test, can DINA fit the data as well as the G-DINA model without monotonic constraints?
6. Fit the following models to the data:

   `Items 1-4: GDINA, Items 5: LLM, Items 6: DINA, Items 7: ACDM, Items 8: DINO, Items 9-10: LLM, Item 11: RRUM, Item 12: LLM, Item 13-14: ACDM, Item 15: RRUM`

   How many parameters need to be estimated? Can these models fit the data as well as the G-DINA model without monotonic constraints in terms of both absolute and relative fit?
7. OPTIONAL Check the help page for `modelfit` function for other methods that can be used to evaluate absolute model-data fit.

Click HERE for solutions.

6 Model Comparison

Item level model comparison aims to evaluate whether the saturated G-DINA model can be replaced by reduced CDMs without a significant loss in model data fit for each item. The `GDINA` package allows to conduct item-level model comparsion using the Wald test, likelihood ratio (LR) test or Lagrange multiplier (LM) test. For Wald test, see de la Torre and Lee (2013), and Ma, Iaconangelo and de la Torre (2016) for details. For LR test and a two-step LR approximation procedure, see Sorrel, de la Torre, Abad, and Olea (2017) and Ma (2017). For LM test, which is only applicable for DINA, DINO and ACDM, see Sorrel, Abad, Olea, de la Torre, and Barrada (2017). In this session, we implement the Wald test for item level model comparison as an example. We need to fit the G-DINA model to the data first.

```
data1 <- read.table(file = "data1.dat", header = TRUE)
Q1 <- read.table(file = "Q1.txt")
fit1 <- GDINA(dat = data1, Q = Q1, verbose = 0)
```

We use `modelcomp` function with the object returned from `GDINA` function as the first input.

Code

```
modelcomp(GDINA.obj, ...)
```

By default, the `DINA, DINO, ACDM, LLM` and `RRUM` will be compared with the G-DINA model for each item. Test statistics and p-values can be printed for all items **requiring two or more attributes**.

```
mc <- modelcomp(fit1)
mc

##
##  Wald statistics for items requiring two or more attributes:
##            DINA    DINO    ACDM     LLM    RRUM
## Item 5  29.9253 22.3096  0.5710  0.0062  2.8804
## Item 6  20.0336 63.8689 11.2618  2.0608  1.0263
## Item 7  50.5385 15.6908  4.3437  0.0005  0.0005
## Item 8  42.5703 67.8317  1.8594  0.7657  1.0512
## Item 9  11.9323 34.3091  0.0375  0.2629  0.9811
## Item 10 30.4916 59.7346  1.1380  1.2235  1.0074
## Item 11  6.9477 26.4988  0.1448  1.4234  0.0073
## Item 12  3.8234 38.5168  2.1114  0.7511  0.0130
## Item 13  3.5836 42.3570  5.9603  3.2246  1.7784
## Item 14 67.3184 98.4561 11.5480  4.4219  8.7383
## Item 15 18.5780 54.1882 16.1332 11.0710 13.9791
##
## p-values for items requiring two or more attributes:
##           DINA  DINO   ACDM    LLM   RRUM
## Item 5  0.0000 0e+00 0.4499 0.9371 0.0897
## Item 6  0.0000 0e+00 0.0008 0.1511 0.3110
## Item 7  0.0000 4e-04 0.0371 0.9824 0.9823
## Item 8  0.0000 0e+00 0.1727 0.3816 0.3052
## Item 9  0.0026 0e+00 0.8464 0.6081 0.3219
```

```
## Item 10 0.0000 0e+00 0.2861 0.2687 0.3155
## Item 11 0.0310 0e+00 0.7036 0.2328 0.9317
## Item 12 0.1478 0e+00 0.1462 0.3861 0.9093
## Item 13 0.1667 0e+00 0.0146 0.0725 0.1823
## Item 14 0.0000 0e+00 0.0210 0.3519 0.0680
## Item 15 0.0049 0e+00 0.0028 0.0258 0.0074
```

To extract the p-values and test statistics:

```
# test statistics
extract(mc, what = "stats")

##            DINA    DINO    ACDM     LLM    RRUM
## Item 5  29.9253 22.3096  0.5710  0.0062  2.8804
## Item 6  20.0336 63.8689 11.2618  2.0608  1.0263
## Item 7  50.5385 15.6908  4.3437  0.0005  0.0005
## Item 8  42.5703 67.8317  1.8594  0.7657  1.0512
## Item 9  11.9323 34.3091  0.0375  0.2629  0.9811
## Item 10 30.4916 59.7346  1.1380  1.2235  1.0074
## Item 11  6.9477 26.4988  0.1448  1.4234  0.0073
## Item 12  3.8234 38.5168  2.1114  0.7511  0.0130
## Item 13  3.5836 42.3570  5.9603  3.2246  1.7784
## Item 14 67.3184 98.4561 11.5480  4.4219  8.7383
## Item 15 18.5780 54.1882 16.1332 11.0710 13.9791

# p-values
extract(mc, what = "pvalues")

##           DINA  DINO   ACDM    LLM   RRUM
## Item 5  0.0000 0e+00 0.4499 0.9371 0.0897
## Item 6  0.0000 0e+00 0.0008 0.1511 0.3110
## Item 7  0.0000 4e-04 0.0371 0.9824 0.9823
## Item 8  0.0000 0e+00 0.1727 0.3816 0.3052
## Item 9  0.0026 0e+00 0.8464 0.6081 0.3219
## Item 10 0.0000 0e+00 0.2861 0.2687 0.3155
## Item 11 0.0310 0e+00 0.7036 0.2328 0.9317
## Item 12 0.1478 0e+00 0.1462 0.3861 0.9093
## Item 13 0.1667 0e+00 0.0146 0.0725 0.1823
## Item 14 0.0000 0e+00 0.0210 0.3519 0.0680
## Item 15 0.0049 0e+00 0.0028 0.0258 0.0074

# degrees of freedom
extract(mc, what = "df")

##         DINA DINO ACDM LLM RRUM
## Item 5     2    2    1   1    1
## Item 6     2    2    1   1    1
## Item 7     2    2    1   1    1
```

```
## Item 8      2     2     1    1     1
## Item 9      2     2     1    1     1
## Item 10     2     2     1    1     1
## Item 11     2     2     1    1     1
## Item 12     2     2     1    1     1
## Item 13     2     2     1    1     1
## Item 14     6     6     4    4     4
## Item 15     6     6     4    4     4
```

It is also possible to conduct the hypothesis test to a subset of items using some reduced models. For example,

```
mc2 <- modelcomp(fit1, item = c(4, 7, 10), models = c("DINA", "LLM"))
mc2

##
##  Wald statistics for items requiring two or more attributes:
##            DINA    LLM
## Item 7  50.5385 0.0005
## Item 10 30.4916 1.2235
##
## p-values for items requiring two or more attributes:
##         DINA    LLM
## Item 7     0 0.9824
## Item 10    0 0.2687
```

Exercise

We will still use `data2` and `Q2` for the following exercise.
Evaluate whether any reduced CDM can be used in place of the saturated model for each item.

1. Which model should be selected for each item if we believe the model with the largest non-significant p-value is the most appropriate?
2. Fit the selected model to the data. Does this combination of models have a good absolute fit?
3. Can the CDM combinations fit data as well as the saturated model?
4. OPTIONAL Check the help page for `modelcomp` function and conduct item-level model selection (comparison) using likelihood ratio test.

Click HERE for solutions.

7 Q-matrix Validation

In this section, the implementation of the Q-matrix validation procedure proposed by de la Torre and Chiu (2016) is introduced.

```
# Load the GDINA package
library(GDINA)
data1 <- read.table(file = "data1.dat", header = TRUE)
# Q-matrix
Q1 <- read.table(file = "Q1.txt")
# Fit the data using G-DINA model
fit <- GDINA(dat = data1, Q = Q1, model = "GDINA", verbose = 0)
```

To conduct the Q-matrix validation, we use `Qval` function with the objects returned from `GDINA` function as the first input. `eps` specifies the PVAF cutoff.

Code

```
Qval(GDINA.obj, eps = 0.95, ...)
```

```
# Conduct Q-matrix validation The default eps = 0.95
Qvalid <- Qval(fit)
```

The suggested Q-matrix will be printed if we type `Qvalid` in the console:

```
# Print the suggested Q-matrix
Qvalid

##
## Q-matrix validation based on PVAF method
##
## Suggested Q-matrix:
##
##    A1 A2 A3
## 1  1  0  1*
## 2  0  0  1
## 3  0  1  0
## 4  0  0  1
## 5  1  1  0
## 6  1  1  0
## 7  1  1  0
## 8  0  1  1
## 9  0  1  1
## 10 0  1  0*
## 11 0  1  1
## 12 1  0  1
## 13 1  0  1
```

```
## 14 1  1  1
## 15 1  1  1
## Note: * denotes a modified element.
```

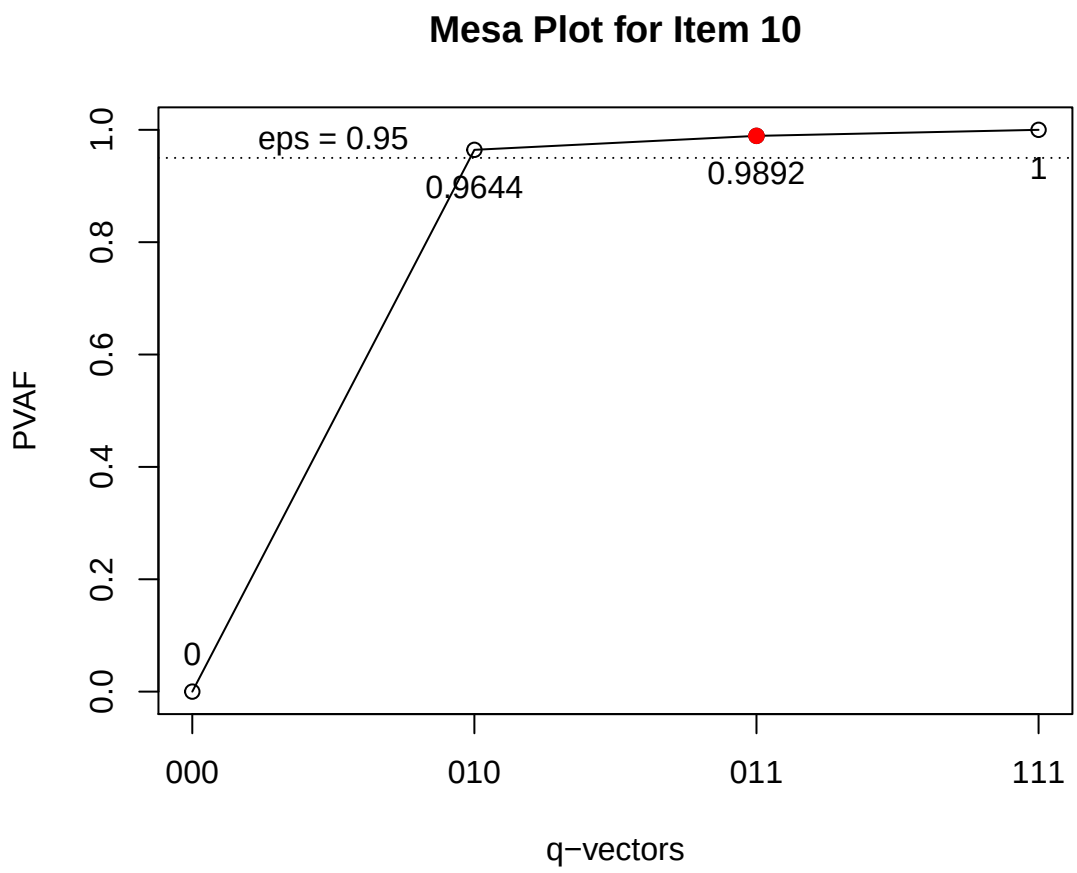
It is typically informative to plot PVAF of the best q-vectors given the number of attributes required for each item using the `plot` function, which takes the following form:

Code

```
plot(Qval.obj, item, eps, ...)
```

where argument `Qval.obj` takes Q-validation results from `Qval`. For example, to draw mesa plot for item 10, we need to run

```
plot(Qvalid, item = 10)
```



`extract` function can be used to extract various components:

Code

```
extract(object, what = c("sug.Q", "varsigma", "PVAF", "eps"), ...)
```

For example, ς^2 and PVAF can also be extracted as in:

```
# print varsigma^2
t(extract(Qvalid, what = "varsigma"))

##          100    010    001    110    101    011    111
##  [1,] 0.0006 0.0001 0.0000 0.0006 0.0007 0.0002 0.0007
##  [2,] 0.0001 0.0018 0.0640 0.0020 0.0641 0.0640 0.0642
##  [3,] 0.0014 0.0204 0.0009 0.0206 0.0025 0.0206 0.0209
##  [4,] 0.0000 0.0002 0.0311 0.0005 0.0312 0.0314 0.0316
##  [5,] 0.0503 0.0258 0.0005 0.0594 0.0506 0.0262 0.0596
##  [6,] 0.0654 0.0237 0.0005 0.0804 0.0660 0.0241 0.0804
##  [7,] 0.0615 0.0578 0.0009 0.0940 0.0621 0.0587 0.0942
##  [8,] 0.0049 0.0549 0.0022 0.0550 0.0090 0.0642 0.0644
##  [9,] 0.0028 0.0207 0.0084 0.0209 0.0122 0.0348 0.0351
## [10,] 0.0057 0.0354 0.0008 0.0356 0.0063 0.0363 0.0367
## [11,] 0.0002 0.0039 0.0114 0.0039 0.0118 0.0183 0.0185
## [12,] 0.0172 0.0002 0.0150 0.0184 0.0346 0.0163 0.0348
## [13,] 0.0087 0.0001 0.0172 0.0099 0.0307 0.0189 0.0309
## [14,] 0.0441 0.0074 0.0005 0.0482 0.0475 0.0078 0.0589
## [15,] 0.0152 0.0045 0.0045 0.0185 0.0206 0.0142 0.0352

# print PVAF
t(extract(Qvalid, what = "PVAF"))

##          100    010    001    110    101    011 111
##  [1,] 0.8355 0.1929 0.0355 0.8628 0.9558 0.2307   1
##  [2,] 0.0015 0.0274 0.9959 0.0316 0.9975 0.9968   1
##  [3,] 0.0652 0.9792 0.0435 0.9878 0.1180 0.9888   1
##  [4,] 0.0002 0.0074 0.9844 0.0147 0.9873 0.9923   1
##  [5,] 0.8441 0.4328 0.0076 0.9965 0.8483 0.4396   1
##  [6,] 0.8126 0.2946 0.0067 0.9994 0.8199 0.2993   1
##  [7,] 0.6533 0.6135 0.0099 0.9978 0.6595 0.6227   1
##  [8,] 0.0763 0.8526 0.0339 0.8542 0.1402 0.9970   1
##  [9,] 0.0802 0.5900 0.2393 0.5937 0.3467 0.9910   1
## [10,] 0.1546 0.9644 0.0206 0.9708 0.1718 0.9892   1
## [11,] 0.0105 0.2102 0.6145 0.2126 0.6385 0.9853   1
## [12,] 0.4931 0.0048 0.4302 0.5289 0.9946 0.4682   1
## [13,] 0.2808 0.0023 0.5571 0.3204 0.9915 0.6098   1
## [14,] 0.7492 0.1260 0.0081 0.8179 0.8058 0.1321   1
## [15,] 0.4305 0.1289 0.1269 0.5246 0.5858 0.4044   1
```

Exercise

Fit the G-DINA model to the `data2`, then answer the following questions:

1. Conduct Q-matrix validation with `eps = 0.95`. Are there any suggested modifications to the Q-matrix?
2. Change `eps = 0.90`. Do we get a different suggested Q-matrix?
3. Draw a mesa plot for item 5. Which q-vector should be used based on the mesa plot?
4. Does the G-DINA model have good absolute fit in terms of log odds ratio when used with original and suggested Q-matrices based on eps = 0.9?
5. Based on AIC and BIC, which Q-matrix is better when used with G-DINA model?

Click HERE for solutions.

8 Differential Item Functioning, Classification Accuracy, and Data Simulation

8.1 Differential Item Functioning

To conduct a DIF analysis, we need to have responses from two groups. We assume that the first 400 individuals are Group 1 and the rest 437 are Group 2 for both datasets.

To do a DIF analysis, use `dif` function:

Code

```
dif(dat, Q, group, method = "wald", ...)
```

`group` must be a numerical vector indicating the group each individual belongs to. Only two groups are allowed.
`...` specifies the arguments that need to be passed to `GDINA` function for model estimation.

We need to create a group indicator variable first:

```
Q1 <- read.table(file = "Q1.txt")
data1 <- read.table(file = "data1.dat", header = TRUE)
gr <- c(rep(1, 400), rep(2, 437))
```

The DIF analysis can be conducted as below:

```
difout <- dif(dat = data1, Q = Q1, group = gr)
```

To print the results, type `difout` in the console:

```
difout

##
## Differential Item Functioning Detection
##          Wald stat. df p.value adj.pvalue
## Item 1       0.3868  2  0.8241     1.0000
## Item 2      23.8766  2  0.0000     0.0001
## Item 3       0.8323  2  0.6596     1.0000
## Item 4       4.8614  2  0.0880     1.0000
## Item 5      19.0396  4  0.0008     0.0116
## Item 6      25.0965  4  0.0000     0.0007
## Item 7      18.1684  4  0.0011     0.0172
## Item 8      12.0553  4  0.0169     0.2542
## Item 9       3.2523  4  0.5165     1.0000
## Item 10     11.4504  4  0.0219     0.3291
## Item 11      4.1846  4  0.3816     1.0000
## Item 12      3.7879  4  0.4355     1.0000
## Item 13      1.4090  4  0.8426     1.0000
```

```
## Item 14    20.0089  8  0.0103     0.1545
## Item 15     6.2162  8  0.6230     1.0000
##
## Note: adjusted pvalues are based on the bonferroni correction.
```

We can add additional options to model calibration. For example, we can fit the DINA model:

```
difout2 <- dif(dat = data1, Q = Q1, group = gr, model = "DINA")
```

```
difout2

##
## Differential Item Functioning Detection
##         Wald stat. df p.value adj.pvalue
## Item 1      1.0688  2  0.5860     1.0000
## Item 2      0.4146  2  0.8128     1.0000
## Item 3      0.0029  2  0.9985     1.0000
## Item 4      3.3032  2  0.1917     1.0000
## Item 5      1.1772  2  0.5551     1.0000
## Item 6      0.1394  2  0.9327     1.0000
## Item 7      0.6540  2  0.7211     1.0000
## Item 8      4.4055  2  0.1105     1.0000
## Item 9      0.5409  2  0.7630     1.0000
## Item 10     8.2544  2  0.0161     0.2419
## Item 11     0.9503  2  0.6218     1.0000
## Item 12     2.2913  2  0.3180     1.0000
## Item 13     2.0861  2  0.3524     1.0000
## Item 14     0.3255  2  0.8498     1.0000
## Item 15     0.0070  2  0.9965     1.0000
##
## Note: adjusted pvalues are based on the bonferroni correction.
```

Another way for DIF detection is based on the likelihood ratio test as shown below, which however can be slow.

```
difout3 <- dif(dat = data1, Q = Q1, group = gr, model = "DINA", method = "LR")
```

```
difout3

##
## Differential Item Functioning Detection
##           neg2LL LRstat df p.value adj.pvalue
## Item 1  15008.41 0.9208  2  0.6310     1.0000
## Item 2  15007.59 0.1004  2  0.9510     1.0000
## Item 3  15007.49 0.0017  2  0.9992     1.0000
## Item 4  15009.82 2.3269  2  0.3124     1.0000
```

```
## Item 5  15008.58 1.0887  2  0.5802     1.0000
## Item 6  15007.60 0.1074  2  0.9477     1.0000
## Item 7  15007.90 0.4138  2  0.8131     1.0000
## Item 8  15011.46 3.9715  2  0.1373     1.0000
## Item 9  15007.81 0.3207  2  0.8518     1.0000
## Item 10 15014.60 7.1142  2  0.0285     0.4278
## Item 11 15008.32 0.8358  2  0.6584     1.0000
## Item 12 15008.65 1.1577  2  0.5605     1.0000
## Item 13 15008.66 1.1733  2  0.5562     1.0000
## Item 14 15007.79 0.2973  2  0.8619     1.0000
## Item 15 15007.49 0.0057  2  0.9972     1.0000
##
## Note: adjusted pvalues are based on the bonferroni correction.
```

There are another two options related to LR approach - LR.type and LR.approx. You can check the help page for more details.

8.2 Classification Accuracy

The function `CA` calculates test-, pattern- and attribute-level classification accuracy indices based on the estimates from the GDINA function using approaches in Iaconangelo (2017) and Wang, Song, Chen, Meng, and Ding (2015). It is only applicable for dichotomous attributes.

```
fit1 <- GDINA(dat = data1, Q = Q1, verbose = 0)
CA(fit1)

## Classification Accuracy
##
## Test level accuracy =  0.6974
##
## Pattern level accuracy:
##
##    000    100    010    001    110    101    011    111
## 0.6598 0.6693 0.5154 0.7916 0.6990 0.5582 0.5846 0.8001
##
## Attribute level accuracy:
##
##     A1     A2     A3
## 0.8910 0.8642 0.8678
```

8.3 Data Simulation

`simGDINA` is the function that can be used to simulate data based on the G-DINA model, DINA model, DINO model, *A*-CDM, LLM, RRUM or their combinations. This function can accept various formats of inputs.

The simplest way of simulating data is to specify guessing and slip parameters as follows:

```
  # # -- guessing and slip parameters for each item
  # # need to be specified in a matrix or data frame
  # # of dimension J x 2 (column 1 is guessing and
  # # column 2 is slip)

N <- 5000
Q <- Q1
J <- nrow(Q)
gs <- data.frame(guess = rep(0.1,J),slip = rep(0.2,J))
gs

##    guess slip
## 1    0.1  0.2
## 2    0.1  0.2
## 3    0.1  0.2
## 4    0.1  0.2
## 5    0.1  0.2
## 6    0.1  0.2
## 7    0.1  0.2
## 8    0.1  0.2
## 9    0.1  0.2
## 10   0.1  0.2
## 11   0.1  0.2
## 12   0.1  0.2
## 13   0.1  0.2
## 14   0.1  0.2
## 15   0.1  0.2
```

```
# Simulated DINA model
set.seed(12345)
sim <- simGDINA(N, Q, gs.parm = gs, model = "DINA")
```

To obtain the simulated data, use `extract` function with the argument `what = "dat"`:

```
dat <- extract(sim, what = "dat")
```

To obtain the simulated true person parameters, use `extract` function with the argument `what = "attribute"`:

```
att <- extract(sim, what = "attribute")
```

It is simple to fit a CDM to simulated data and evaluate classification rates in `GDINA` package:

```
# fit DINA model
fitsim <- GDINA(dat, Q, model = "DINA", verbose = 0)
# evaluate classification rates
ClassRate(att, personparm(fitsim))
```

```
## $PCA
## [1] 0.9309333
##
## $PCV
## [1] 0.9986 0.9648 0.8294
```

We can also simulate the DINO model in the same manner by specifying `model = "DINO"`. For A-CDM, LLM and RRUM, they have $K_j + 1$ item parameters, so we need additional assumptions to generate data. By default, the contribution of each attribute to success probabilities is generated randomly but we can also assume each attribute contributes equally (`type = "equal"`). Note that the contributions of each attribute for A-CDM, LLM and RRUM are always positive, which implies that monotonic constraints are always satisfied.
The G-DINA model can also be simulated. To simulate the G-DINA model, all `delta` are simulated randomly. By default, monotonic constrained item success probabilities are generated. To generate item success probabilities without monotonic constraints, set `mono.constraint = FALSE`.
By default, attribute patterns are generated from uniform distribution.

```
m <- c(rep("DINA",3),rep("DINO",3),
       rep("ACDM",3),rep("LLM",2),
       rep("RRUM",2),rep("GDINA",2))
set.seed(12345)
sim2 <- simGDINA(N, Q, gs.parm = gs, model = m)
# print item parameters
extract(sim2, what = "catprob.parm")
```

```
## $`Item 1`
## P(0) P(1)
##  0.1  0.8
##
## $`Item 2`
## P(0) P(1)
##  0.1  0.8
##
## $`Item 3`
## P(0) P(1)
##  0.1  0.8
##
## $`Item 4`
## P(0) P(1)
##  0.1  0.8
##
## $`Item 5`
## P(00) P(10) P(01) P(11)
##   0.1   0.8   0.8   0.8
##
## $`Item 6`
## P(00) P(10) P(01) P(11)
##   0.1   0.8   0.8   0.8
```

```
##
## $`Item 7`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.6046327 0.2953673 0.8000000
##
## $`Item 8`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.7130412 0.1869588 0.8000000
##
## $`Item 9`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.6326876 0.2673124 0.8000000
##
## $`Item 10`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.7267548 0.1431770 0.8000000
##
## $`Item 11`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.3632198 0.4379432 0.8000000
##
## $`Item 12`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.1413347 0.5660324 0.8000000
##
## $`Item 13`
##     P(00)     P(10)     P(01)     P(11)
## 0.1000000 0.1966031 0.4069112 0.8000000
##
## $`Item 14`
##    P(000)    P(100)    P(010)    P(001)    P(110)    P(101)    P(011)
## 0.1000000 0.4564570 0.6093937 0.7928159 0.6159763 0.7939105 0.7981011
##    P(111)
## 0.8000000
##
## $`Item 15`
##    P(000)    P(100)    P(010)    P(001)    P(110)    P(101)    P(011)
## 0.1000000 0.1007956 0.3738423 0.4237463 0.5392529 0.5751828 0.4910820
##    P(111)
## 0.8000000
```

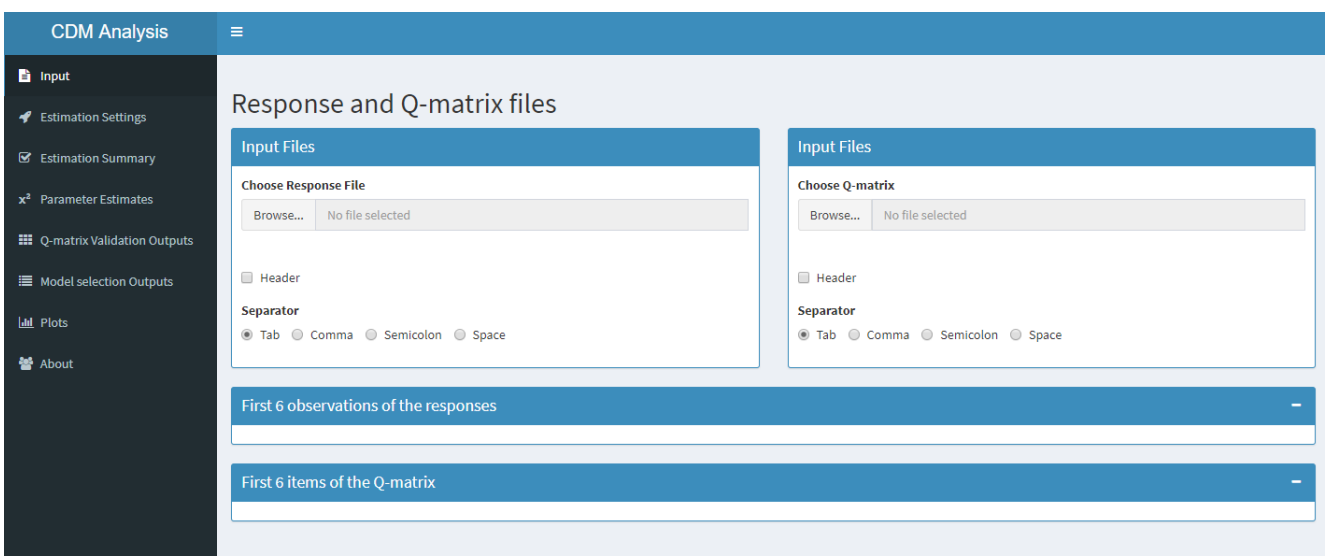
Data can also be simulated using item success probabilities or delta parameters. The attributes can be simulated from a higher-order or multivariate nomal distribution. For example, we can simulate data using the estimates of `data1` and `Q1`.

```
fit1 <- GDINA(data1,Q1,verbose = 0)
ip <- coef(fit1)
delta <- coef(fit1,"delta")
```

```
set.seed(12345)
N <- 3000
K <- 3
#higher-order person ability
theta <- rnorm(N)
#higher-order attribute parameters
lambda <- data.frame(a=rep(1,K),b=rnorm(K))
lambda

##   a          b
## 1 1  0.5098921
## 2 1 -0.7153643
## 3 1 -0.4055215

sim2 <- simGDINA(N,Q1,catprob.parm = ip,
                 att.dist = "higher.order",
                 higher.order.parm = list(theta = theta,lambda = lambda))

sim3 <- simGDINA(N,Q1,delta.parm = delta,
                 att.dist = "higher.order",
                 higher.order.parm = list(theta = theta,lambda = lambda))
```

8.4 Graphical User Interface

A GUI is provided if you are not comfortable with coding in R. Basically, you need to run function `startGDINA` without any input in the console. A window as shown below will appear in your browser.



Optional exercises

You can choose to answer any of the questions below that you are interested in using `data2`.

1. Fit the G-DINA model to the data and evaluate its classification accuracy.
2. Read the help page of `simGDINA` and simulate responses of 500 individuals to 15 items using the G-DINA model based on `Q2`. Make sure that the guessing and slip parameters are equal to 0.2 for all items.
3. Analyze `data2` using the GUI.
4. Assume the first 400 individuals are group 1 and the rest are group 2. Conduct a DIF analysis using Wald test.

Click HERE for solutions.

9 Putting It Together

In this section, we will demonstrate some of the main functions of the package. In doing so, we will analyze an artificial clinical assessment data that is designed as an instrument to examine multiple mental disorders. Experts in clinical psychology have developed a Q-matrix, named as `Q3.txt`, for this dataset. The data, which is named as `data3.dat`, contains responses of 1210 subjects to 33 items. Each item asks whether a symptom happens to a patient or not.

We will go through all major analyses including: (1) Fitting CDMs to the data; (2) Validating and correcting the Q-matrix; (3) Examining model-data fit; (4) Finding a simpler set of CDMs for the data; (5) Detecting potential DIF items; (6) Calculating the classification accuracy.

Step 1: Read in the data and the Q-matrix

The GDINA package does not require any specific function for importing the data. For example, you can use the `read.table` function for importing text files.

Practice Exercise:
Read the data and the Q-matrix named `data3.dat` and `Q3.txt`, respectively.

Step 2: Fit the saturated CDM

To estimate G-DINA model, call `GDINA` function and specify the data and Q-matrix as the first two arguments. `GDINA` function can accept various arguments, but by default, the fitted model is the G-DINA model.

```
GDINA(dat, Q, model = "GDINA", ...)
```

Returned object contains all estimation results. It is called an object of class `GDINA`, and we can apply various methods to it (e.g., `summary, itemparm, plot, personparm, extract`).

Change the argument `model` to estimate various CDMs. For example, to estimate DINA model, specify `model = "DINA"`. We can also compare the fit of various CDM combinations with that of G-DINA model. For example,

```
models <- c(rep("GDINA",2), "ACDM", rep("DINA",15), rep("DINO",15))
fit <- GDINA(dat, Q, model = models)
```

Another interesting argument is `mono.constraint`. When `mono.constraint = TRUE`, mastering of additional attributes will not lead to a lower probability of success.

Practice Exercise:

Fit the saturated G-DINA model to the data with monotonicity constraints imposed using the `GDINA` function.

Evaluate the model fit to the data by generating the heatmap plots using the `plot` function.

Step 3: Q-matrix validation

To conduct the Q-matrix validation, we use `Qval` function with the objects returned from the `GDINA` function as the first input. `eps` specifies the PVAF cutoff.

```
Qval(GDINA.obj = "GDINA", eps = 0.95, ...)
```

It is typically informative to plot PVAF of all possible q-vectors for each item. To do so, one can use the `mesaplot` function. The argument `item` specifies for which item should be mesaplot be created.

```
mesaplot(Qval.obj, item, ...)
```

Practice Exercise:

If there is any misspecification in the Q-matrix, it will dramatically affect the item parameter estimation and, eventually, the classification results. To address this, perform the Q-matrix validation using `eps`=0.9 using the fitted model in step (2).

Generate the MESA plots for the items that have suggested changes. Compare the suggested q-vectors based on the MESA plots and ϵ. **Hint**: There should be four items with suggested q-vectors, but we only need to change the q-vector for two of them.

Modify the Q-matrix based on the suggested q-vectors of the MESA plots. **Hint**: To modify the Q-matrix, you can use `Q3[,]=c()`.

Step 4: Model-data fit

To evaluate the model data fit, we need to fit some CDMs to the data first. `summary` and several other functions can be used to print some test level relative model-data fit statistics.

To evaluate absolute fit, we need to use `itemfit` function with the objects returned from the `GDINA` function as the input.

We can also plot (adjusted) p-values of fit statistics for all item pairs using the `plot` function with the objects returned from `itemfit` function as the input.

If we want to compare two nested models, we can use likelihood ratio test. AIC and BIC can also be used when two models are fitted to the same data. To compare if a nested model can fit the data as well as the G-DINA model, we use `anova` function with two model fit objects as the inputs.

Practice Exercise:

Re-fit the G-DINA model with monotonicity constraints once again, but this time using the corrected Q-matrix. Check the model fit statistics via the heatmap plots using `plot` function. Is there any misfitting item?

Plot the item response function (IRF) of Item 16 using the `plot` function. Because the disorders are being diagnosed by the items, one could argue that having one is enough for the symptom to manifest itself. Hence, it would be interesting to determine whether fitting the DINO model to Item 16 while fitting the GDINA model for other items can improve the overall model. Afterwards, check the model fit statistics using heat plots. Is Item 16 problematic?

Using the fitted model above, check the model fit statistics using heat plots. Is there any misfitting item?

Step 5: Model comparison

To implement the Wald test for item level model comparison, we need to fit the G-DINA model to the data first.

To evaluate whether reduced CDMs can be used in place of the G-DINA model for each item without loss of model data fit significantly, we use `modelcomp` function with the object returned from the `GDINA` function as the first input. By default, the `DINA, DINO, ACDM, LLM` and `RRUM` will be compared with the G-DINA model for each item. Wald statistics and p-values can be printed for all items **requiring two or more attributes**.

Practice Exercise:

Fit the reduced or simpler models, namely, the DINA, DINO, ACDM, LLM, and RRUM, to the data. Compare these models with the fitted model in step (4). Is any of these reduced models fit as good as the saturated model? Perform a likelihood ratio (LR) test using the `anova` function.

Using the `modelcomp` function, conduct the Wald test for item fit evaluation. Based on the maximum p-value, choose the "best" model for each item. Afterwards, fit these models to the data. Then, using LR test, determine whether this model fits the data equally well or not as the saturated model.

Fit the model using suggested models by the Wald-test.

Step 6: DIF Detection

To conduct a DIF analysis, we need to have responses from two groups.

```
dif(dat, Q, group, method = "wald", ...)
```

`group` must be a scalar indicating which column in the data is the group indicator, or a numerical vector indicating the group each individual belongs to. If it is a vector, its length must be equal to the number of

individuals. Only two groups are allowed.

The Wald test is used by default. If you want to use the likelihood ratio test, specify `method = "LR"`. The argument `difitem` can be used to explore DIF only for a subset of the items

We can also plot item success probabilities with or without error bars for both groups using the `plot` function and the argument `errorbar = TRUE`.

Practice Exercise:

In this dataset, the first 600 subjects are males, whereas the rest are females. Perform differential item functioning (DIF) analysis using the Wald and the LR test using the `dif` function. Is there any item exhibiting DIF?

Because LR test requires refitting the model which takes a while, we only evaluate item 7 using the LR test.

Step 7: Classification Accuracy

The `CA` function calculates test-, pattern- and attribute-level classification accuracy indices based on G-DINA estimates from the `GDINA` function. The argument `what` can be used to specify a person parameter estimation method.

```
CA(GDINA.obj, what = "MAP")
```

Practice Exercise:

A way to evaluate the usefulness of a CDM is its classification accuracy. Compute the classification accuracy for the final model resulting of the model comparison analysis

Step 8: Item Selection

Some item discrimination indices can be computed considering the item parameter estimates. The simplest item discrimination index is computed as $1 - g_j - s_j$, where g_j and s_j are the guessing and slip parameters for item j, respectively. The discrimination index can be printed using `extract(obj,what = "discrim")`.

Item selection might improve classification accuracy, particularly if some items have a negative or close to 0 item discrimination index.

Practice Exercise:

Compute the discrimination index of each item based on the guessing and slip parameter estimates. Which two items are the most high and low discrimination? Hint: The two items with the most high and low discrimination indices are 0_ and 0_.

Remove the two items with the lowest discrimination indices. Fit the saturated model and obtain its estimated attribute patterns. Afterwards, compute the classification accuracy. What do you observe when these rates are compared with those in step (7)? Hint: The two items with the lowest discrimination indices are 0_ and 2_.

Re-do step (7) but this time removing the two most highly discriminating items. What can you observe when these rates are compared with previous results? Hint: The two items with the highest discrimination indices are _1 and _2.

Let's compare the classification rates by attaching them using `rbind`.
Click HERE for solutions.

10 Answers to Exercises

10.1 Answers to Section 4

```
library(GDINA)
data2 <- read.table(file = "data2.dat", header = TRUE)
# Q-matrix
Q2 <- read.table(file = "Q2.txt")
```

Fit the G-DINA model to the data, then answer the following questions:

```
exefit1 <- GDINA(dat = data2, Q = Q2, model = "GDINA", verbose = 0)
```

1. For Item 15, the G-DINA model parameter estimates are P(100) = 0.4657, P(101) = 0.6696.
2. The corresponding SEs for P(100) and P(101) are 0.0737 and 0.0630, respectively.

   ```
   coef(exefit1, withSE = TRUE)[[15]]
   ```

3. The proportion of individuals having an attribute pattern of 111 in the population is estimated to be 0.3112

   ```
   extract(exefit1, "posterior.prob")
   ```

4. The proportion of individuals who master $\boldsymbol{\alpha}1$ in the population is estimated to be 0.8037.

```
summary(exefit1)
```

5. Plot item success probabilities for Items 5, 8 and 15. Which item(s) appear to follow the DINO models?

```
plot(exefit1, item = c(5, 8, 15))
```

6. Find the EAP estimate of attribute pattern for the third individual.

```
personparm(exefit1)[3, ]

## A1 A2 A3
##  1  0  1
```

7. Fit LLM to the data: the number of parameters for LLM is 50.

```
exefit2 <- GDINA(dat = data2, Q = Q2, model = "LLM")
npar(exefit2)
```

8. Fit DINA to the data: guessing and slip parameter estimates of Item 10 are 0.3516 and 0.2432, respectively. The corresponding SEs are 0.0279 and 0.0312.

```
exefit3 <- GDINA(dat = data2, Q = Q2, model = "DINA")
coef(exefit3, "gs", withSE = T)
```

9. Fit a Rasch higher-order DINA model and print higher-order structural parameters.

```
exefit4 <- GDINA(dat = data2, Q = Q2, model = "DINA", att.dist = "higher.order'
    verbose = 0)
coef(exefit4, "lambda")

##    slope intercept
## A1     1    2.9437
## A2     1    0.1255
## A3     1    1.4977
```

10.2 Answers to Section 5

Fit each of the DINA, DINO, and G-DINA model to the data2 without monotonic constraints, then answer the following questions:

```r
data2 <- read.table(file = "data2.dat", header = TRUE)
Q2 <- read.table(file = "Q2.txt")
fit.dina <- GDINA(data2, Q2, model = "DINA")
fit.dino <- GDINA(data2, Q2, model = "DINO")
fit.gdina <- GDINA(data2, Q2, model = "GDINA")
```

1. Based on the transformed correlation statistics, which model(s) cannot fit data well? [Only G-DINA can fit data well.]

```r
ift.dina <- itemfit(fit.dina)
ift.dino <- itemfit(fit.dino)
ift.gdina <- itemfit(fit.gdina)
ift.dina
ift.dino
ift.gdina
```

2. Based on the heatmap plot, is there misfit between item pairs for DINO model? [Yes]

```r
plot(ift.dino)
plot(ift.dino, adjusted = FALSE)
```

3. Based on AIC, which model is preferred (i.e., has the smallest value of AIC)? [G-DINA]

```r
AIC(fit.dina)
AIC(fit.dino)
AIC(fit.gdina)
```

4. Based on BIC, which model is preferred (i.e., has the smallest value of BIC)? [DINO]

```r
BIC(fit.dina)
BIC(fit.dino)
BIC(fit.gdina)
```

5. Based on likelihood ratio test, can DINA fit the data as well as the G-DINA model without monotonic constraints? [No]

```r
anova(fit.dina, fit.dino, fit.gdina)
```

6. Fit the following models to the data:

   Items 1-4: GDINA, Items 5: LLM, Items 6: DINA, Items 7: ACDM, Items 8: DINO, Items 9-10: LLM, Item 11: RRUM, Item 12: LLM, Item 13-14: ACDM, Item 15: RRUM

   How many parameters need to be estimated? [48] Can these models fit the data as well as the G-DINA model without monotonic constraints in terms of both absolute and relative fit? [Yes]

```
fittedCDM <- c(rep("GDINA", 4), "LLM", "DINA", "ACDM", "DINO", "LLM", "LLM",
    "RRUM", "LLM", "ACDM", "ACDM", "RRUM")

exefit <- GDINA(data2, Q2, model = fittedCDM)
npar(exefit)
itemfit(exefit)
anova(exefit, fit.gdina)
```

10.3 Answers to Section 6

We will still use `data2` and `Q2` for the following exercise.

```
data2 <- read.table(file = "data2.dat", header = TRUE)
Q2 <- read.table(file = "Q2.txt")
```

Evaluate whether any of the five reduced CDMs can be used in place of the saturated model for each item using the Wald test.

```
exefit2 <- GDINA(dat = data2, Q = Q2, model = "GDINA")
exemcl <- modelcomp(exefit2)
```

1. Which model should be selected for each item if we believe the model with the largest non-significant p-value is the most appropriate?

```
p <- extract(exemcl, "pvalues")
# which is max p-value
m <- apply(p, 1, function(x) {
    if (all(x < 0.05)) {
        return(0)
    } else {
        return(which.max(x))
    }
})
# create model
model <- c(rep(0, 4), m)
```

2. Fit the selected model to the data. Does this combination of models have a good absolute fit? [Yes]

```
exefit3 <- GDINA(data2, Q2, model = model)
itemfit(exefit3)
```

3. Can the CDM combinations fit data as well as the saturated model? [Yes]

```
anova(exefit2, exefit3)
```

10.4 Answers to Section 7

Fit the G-DINA model to the `data2`, then answer the following questions:

```
library("GDINA")
data2 <- read.table(file = "data2.dat", header = TRUE)
Q2 <- read.table(file = "Q2.txt")
fit <- GDINA(data2, Q2)
```

1. Conduct Q-matrix validation with `eps = 0.95`. Are there any suggested modifications to the Q-matrix? [Yes]

```
Qv <- Qval(fit)
Qv
```

2. Change `eps = 0.90`. Do we get a different suggested Q-matrix? [Yes]

```
Qv2 <- Qval(fit, eps = 0.9)
Qv2
```

3. Draw a mesa plot for item 5. Which q-vector should be used based on the mesa plot? [0,1,0]

```
# you can use either Qv or Qv2 to draw mesa plot - they are identical
plot(Qv, item = 5)
plot(Qv2, item = 5)
```

4. Does the G-DINA model have good absolute fit in terms of log odds ratio when used with original and suggested Q-matrices based on eps = 0.9? [Yes and Yes]

```
new.Q <- extract(Qv2, "sug.Q")
new.fit <- GDINA(data2, new.Q, verbose = 0)
itemfit(fit)
itemfit(new.fit)
```

5. Based on AIC and BIC, which Q-matrix is better when used with G-DINA model? [The new one.]

```
AIC(fit)
AIC(new.fit)

BIC(fit)
BIC(new.fit)
```

10.5 Answers to Section 8

You can choose to answer any of the questions below that you are interested in using `data2`.

```
library("GDINA")
data2 <- read.table(file = "data2.dat", header = TRUE)
Q2 <- read.table(file = "Q2.txt")
```

1. Fit the G-DINA model to the data and evaluate its classification accuracy.

```
fit <- GDINA(data2, Q2)
CA(fit)
```

2. Read the help page of `simGDINA` and simulate responses of 500 individuals to 15 items using the G-DINA model based on `Q2`. Make sure that the guessing and slip parameters are equal to 0.2 for all items.

```
gs <- matrix(0.2, nrow = 15, ncol = 2)
sim <- simGDINA(N = 500, Q = Q2, gs.parm = gs, model = "GDINA")
simdat <- extract(sim, "dat")
```

3. Analyze `data2` using the GUI.

4. Assume the first 400 individuals are group 1 and the rest are group 2. Conduct a DIF analysis under the G-DINA model using the Wald test.

```
gr <- c(rep(1, 400), rep(2, 437))
difout <- dif(dat = data2, Q = Q2, group = gr, model = "GDINA")
difout
```

10.6 Answers to Section 9

1. **Step 1: Read in the data and the Q-matrix.**

   Read the data and the Q-matrix named `data3.dat` and `Q3.txt`, respectively.

```
library("GDINA")
data3 <- read.table(file = "data3.dat")
Q3 <- read.table(file = "Q3.txt")
head(data3[,1:10])

##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10
## 1  1  0  0  0  0  1  0  0  0   0
## 2  0  0  0  0  0  0  0  0  0   0
## 3  0  0  0  1  0  1  0  1  0   0
## 4  0  1  0  0  0  0  0  0  0   0
## 5  0  0  0  0  0  0  0  0  0   0
## 6  0  0  0  0  0  0  0  0  0   0
```

2. **Step 2: Fit the saturated CDM.**

   Fit the saturated G-DINA model to the data with monotonicity constraints imposed using the `GDINA` function.

```
mod <- GDINA(dat=data3,Q=Q3,model="GDINA",
             mono.constraint = TRUE,verbose = 0)
```

   As mentioned, we can do a lot of things with this object. For example, we can apply the `summary` and `coef` functions.

```
summary(mod)
```

```
##
## Test Fit Statistics
##
## Loglik = -21242.36
## AIC    = 42694.72  | penalty   = 210
## BIC    = 43230.05  | penalty   = 745.33
## # par  = 105
##
## Attribute Prevalence
##
##    Level0 Level1
## A1 0.5179 0.4821
## A2 0.4782 0.5218
## A3 0.4816 0.5184
## A4 0.5186 0.4814
```

```
coef(mod)
```

```
## $`Item 1`
##   P(0)   P(1)
## 0.2853 0.7674
##
## $`Item 2`
##  P(00)  P(10)  P(01)  P(11)
## 0.0272 0.6015 0.4323 0.9008
##
## $`Item 3`
##   P(0)   P(1)
## 0.0342 0.1983
##
## $`Item 4`
```

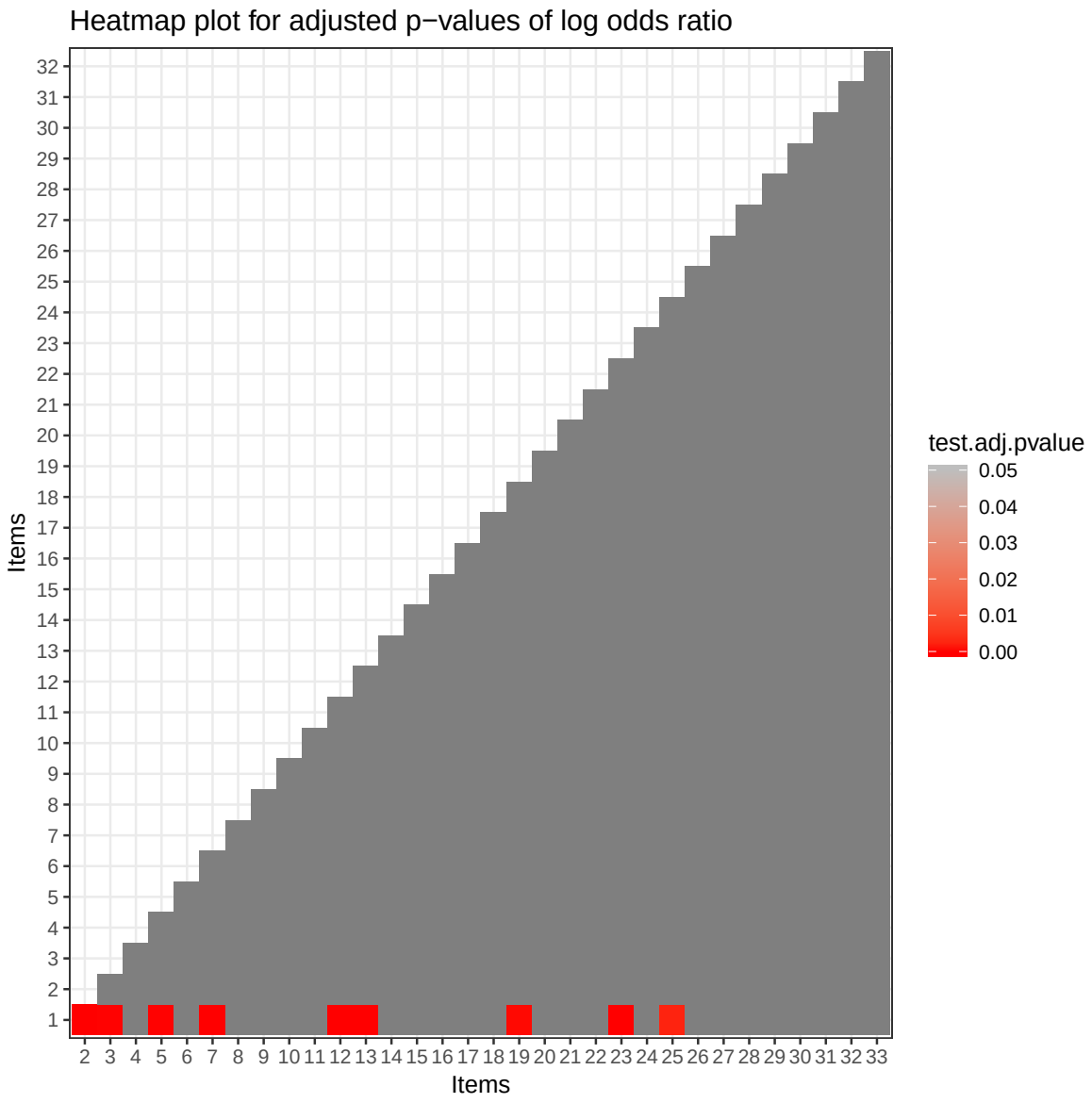
```
##  P(00)  P(10)  P(01)  P(11)
## 0.1735 0.5012 0.5051 0.8959
##
## $`Item 5`
##   P(0)   P(1)
## 0.0422 0.2703
##
## $`Item 6`
##   P(0)   P(1)
## 0.1678 0.9352
##
## $`Item 7`
##  P(00)  P(10)  P(01)  P(11)
## 0.0443 0.4000 0.4143 0.7981
##
## $`Item 8`
##   P(0)   P(1)
## 0.2727 0.7750
##
## $`Item 9`
##  P(00)  P(10)  P(01)  P(11)
## 0.1376 0.7450 0.6616 0.9363
##
## $`Item 10`
##   P(0)   P(1)
## 0.1621 0.7597
##
## $`Item 11`
##   P(0)   P(1)
## 0.1737 0.7951
##
## $`Item 12`
##  P(00)  P(10)  P(01)  P(11)
## 0.0328 0.4421 0.3434 0.7798
##
## $`Item 13`
##  P(00)  P(10)  P(01)  P(11)
## 0.1126 0.3134 0.2997 0.7457
##
## $`Item 14`
##   P(0)   P(1)
## 0.0115 0.2332
##
## $`Item 15`
##   P(0)   P(1)
## 0.0444 0.3605
##
```

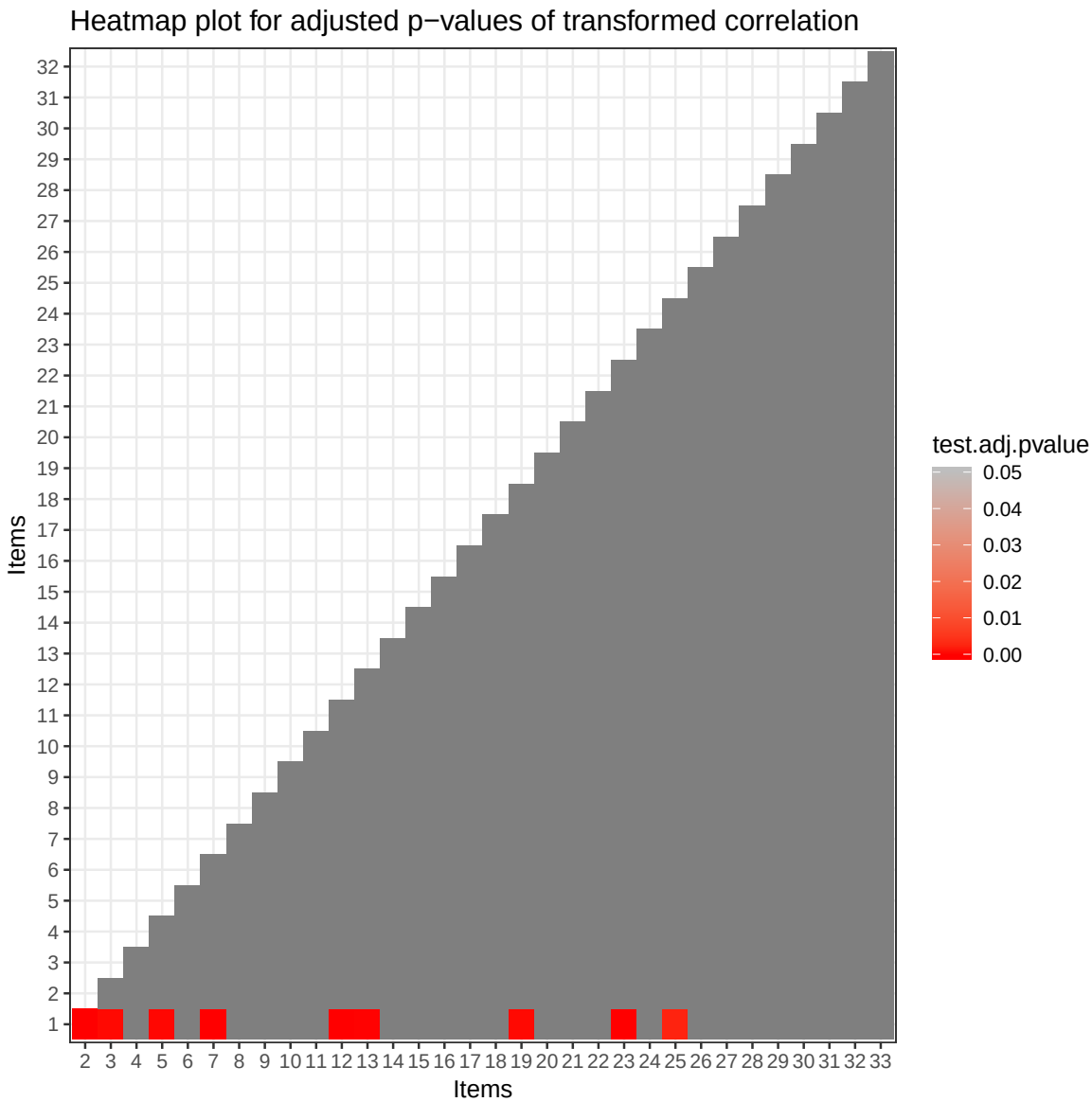
```
## $`Item 16`
##  P(00)  P(10)  P(01)  P(11)
## 0.0453 0.1281 0.3422 0.3981
##
## $`Item 17`
##   P(0)   P(1)
## 0.1191 0.5989
##
## $`Item 18`
##   P(0)   P(1)
## 0.2637 0.7933
##
## $`Item 19`
##  P(00)  P(10)  P(01)  P(11)
## 0.0587 0.3733 0.3707 0.6634
##
## $`Item 20`
##   P(0)   P(1)
## 0.0899 0.2847
##
## $`Item 21`
##   P(0)   P(1)
## 0.0403 0.4247
##
## $`Item 22`
##   P(0)   P(1)
## 0.0184 0.4780
##
## $`Item 23`
##  P(00)  P(10)  P(01)  P(11)
## 0.0074 0.4287 0.2651 0.8091
##
## $`Item 24`
##   P(0)   P(1)
## 0.1826 0.6816
##
## $`Item 25`
##  P(00)  P(10)  P(01)  P(11)
## 0.1671 0.7017 0.4465 0.8609
##
## $`Item 26`
##   P(0)   P(1)
## 0.3387 0.9373
##
## $`Item 27`
##  P(00)  P(10)  P(01)  P(11)
## 0.0605 0.2946 0.1445 0.5665
```

```
##
## $`Item 28`
##   P(0)   P(1)
## 0.0195 0.5713
##
## $`Item 29`
##   P(0)   P(1)
## 0.1974 0.7540
##
## $`Item 30`
##   P(0)   P(1)
## 0.0883 0.6680
##
## $`Item 31`
##   P(0)   P(1)
## 0.0966 0.4476
##
## $`Item 32`
##   P(0)   P(1)
## 0.0799 0.4473
##
## $`Item 33`
##  P(00)  P(10)  P(01)  P(11)
## 0.0634 0.0634 0.5584 0.5584
```

Evaluate the model fit to the data by generating the heatmap plots using the `plot` function.

```
#Model Fit Evaluation
itemfit_initial <- itemfit(mod, p.adjust.methods = "bonferroni")
plot(itemfit_initial)
```


Heatmap plot for adjusted p−values of log odds ratio
Items
Items
test.adj.pvalue
0.05
0.04
0.03
0.02
0.01
0.00



Based on both heatmap plots, it can be seen the Item 1 seems to be a misfitting item.

3. **Step 3: Q-matrix validation**

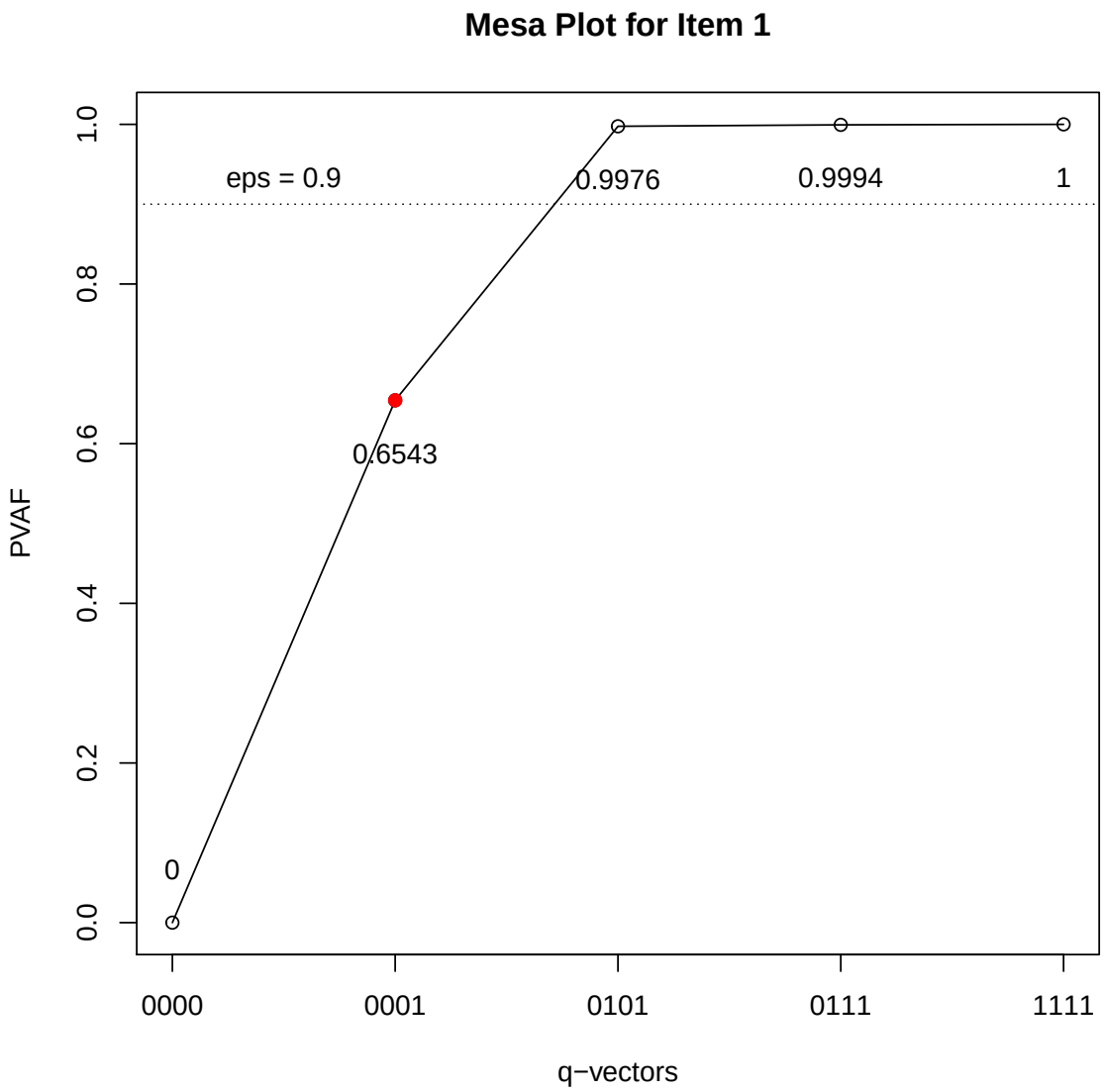
If there is any misspecification in the Q-matrix, it will dramatically affect the item parameter estimation and, eventually, the classification results. To address this, perform the Q-matrix validation using `eps`=0.9 using the fitted model in step (2).

```
Q3valid <- Qval(mod, eps = 0.9)
Q3valid


##
## Q-matrix validation based on PVAF method
```

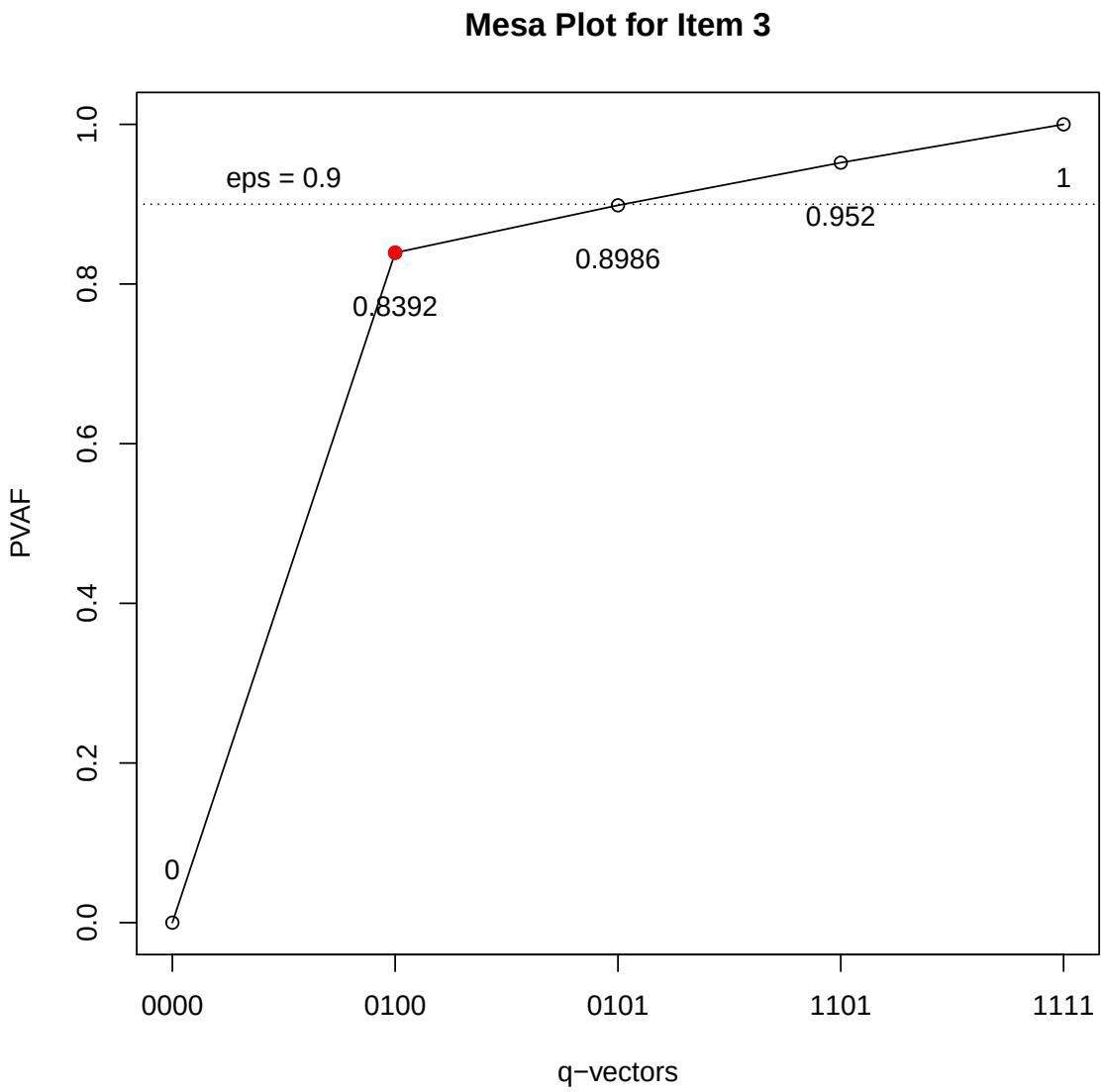
```
##
## Suggested Q-matrix:
##
##    A1 A2 A3 A4
## 1  0  1* 0  1
## 2  0  1  0  1
## 3  1* 1  0  1*
## 4  0  0  1  1
## 5  0  1  0  0
## 6  0  0  0  1
## 7  0  1  0  1
## 8  1  0  0  0
## 9  1  0  1  0
## 10 0  0  1  0
## 11 0  0  1  0
## 12 0  1  0  1
## 13 1  1  0  0
## 14 0  0  1  0
## 15 0  0  1  0
## 16 0  1  0  1
## 17 1  0  0  0
## 18 1  0  0  0
## 19 0  1  0  1
## 20 1* 0  1  0
## 21 1  0  0  0
## 22 0  0  0  1
## 23 0  1  0  1
## 24 1  0  0  0
## 25 1  1  0  0
## 26 1  0  0  0
## 27 1  0  0  1
## 28 0  0  0  1
## 29 0  0  1  0
## 30 1  0  0  0
## 31 0  0  1  0
## 32 1  0  0  0
## 33 0* 0  0  1
## Note: * denotes a modified element.
```

Generate the MESA plots for the items that have suggested changes. Compare the suggested q-vectors based on the MESA plots and ϵ. **Hint**: There should be four items with suggested q-vectors, but we only need to change the q-vector for two of them.

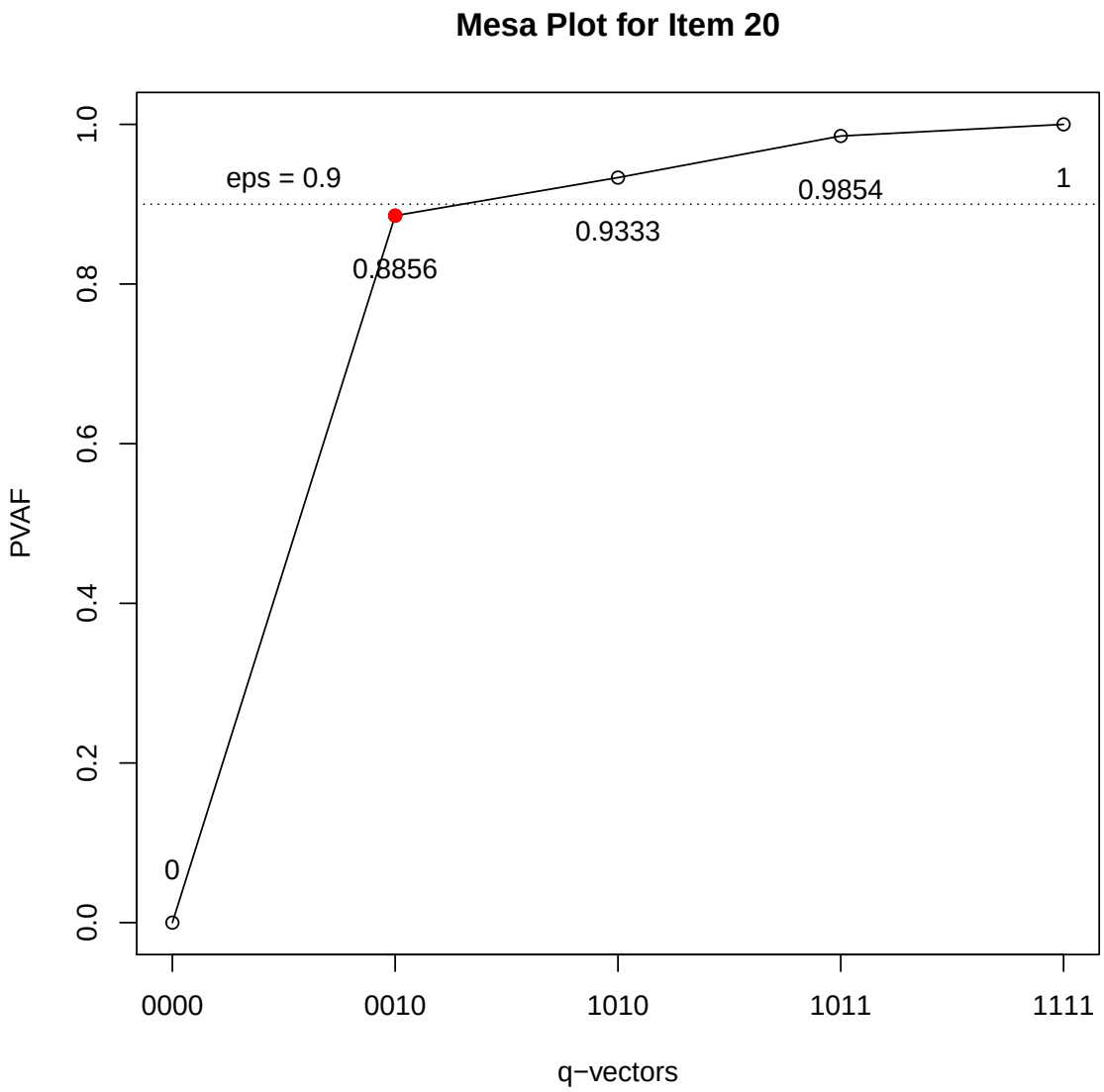
```
plot(Q3valid,item=1,type="best", eps = 0.9)
```



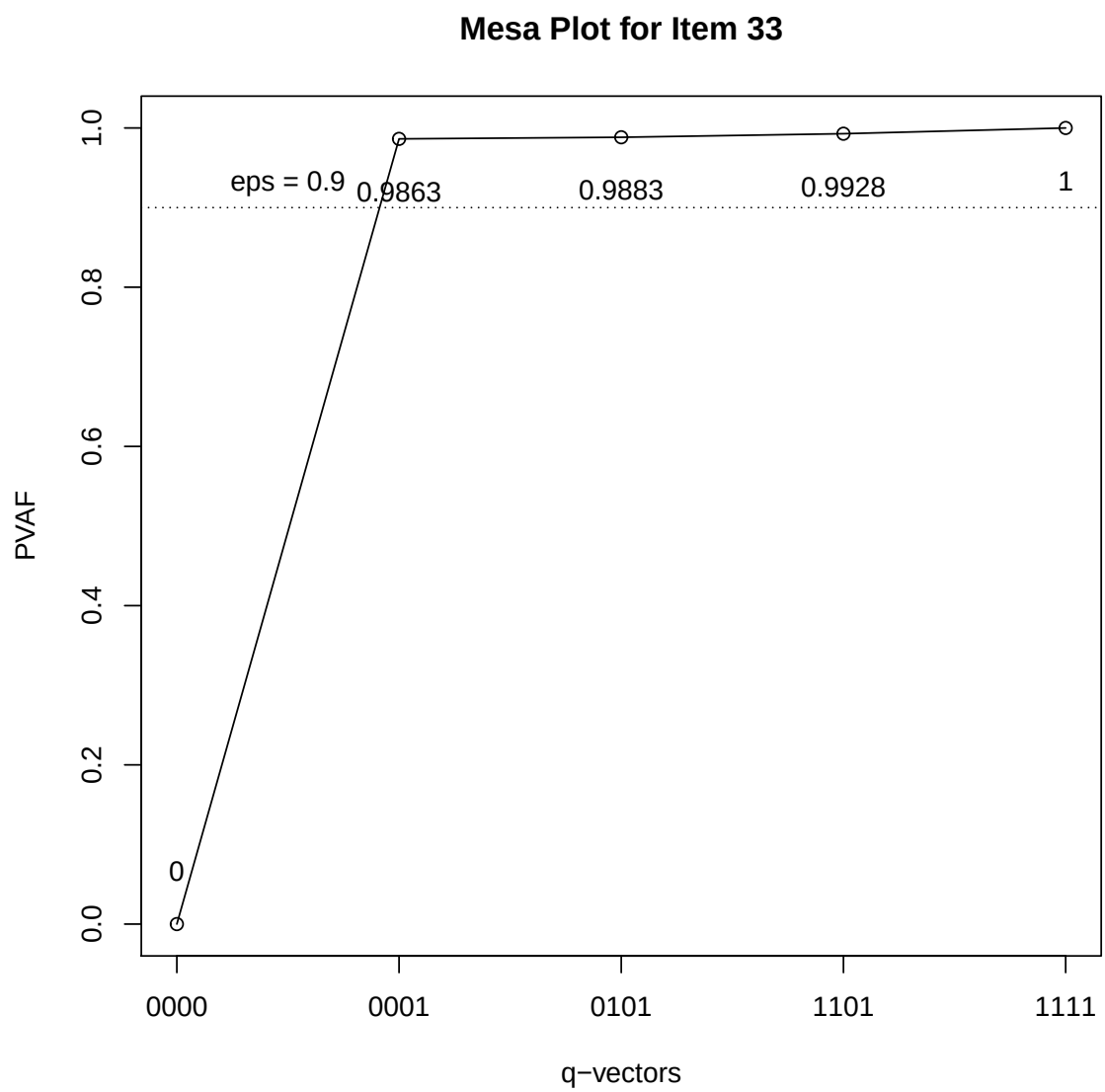

```
plot(Q3valid,item=3,type="best", eps = 0.9)
```




```
plot(Q3valid,item=20,type="best", eps = 0.9)
```




```
plot(Q3valid,item=33,type="best", eps = 0.9)
```



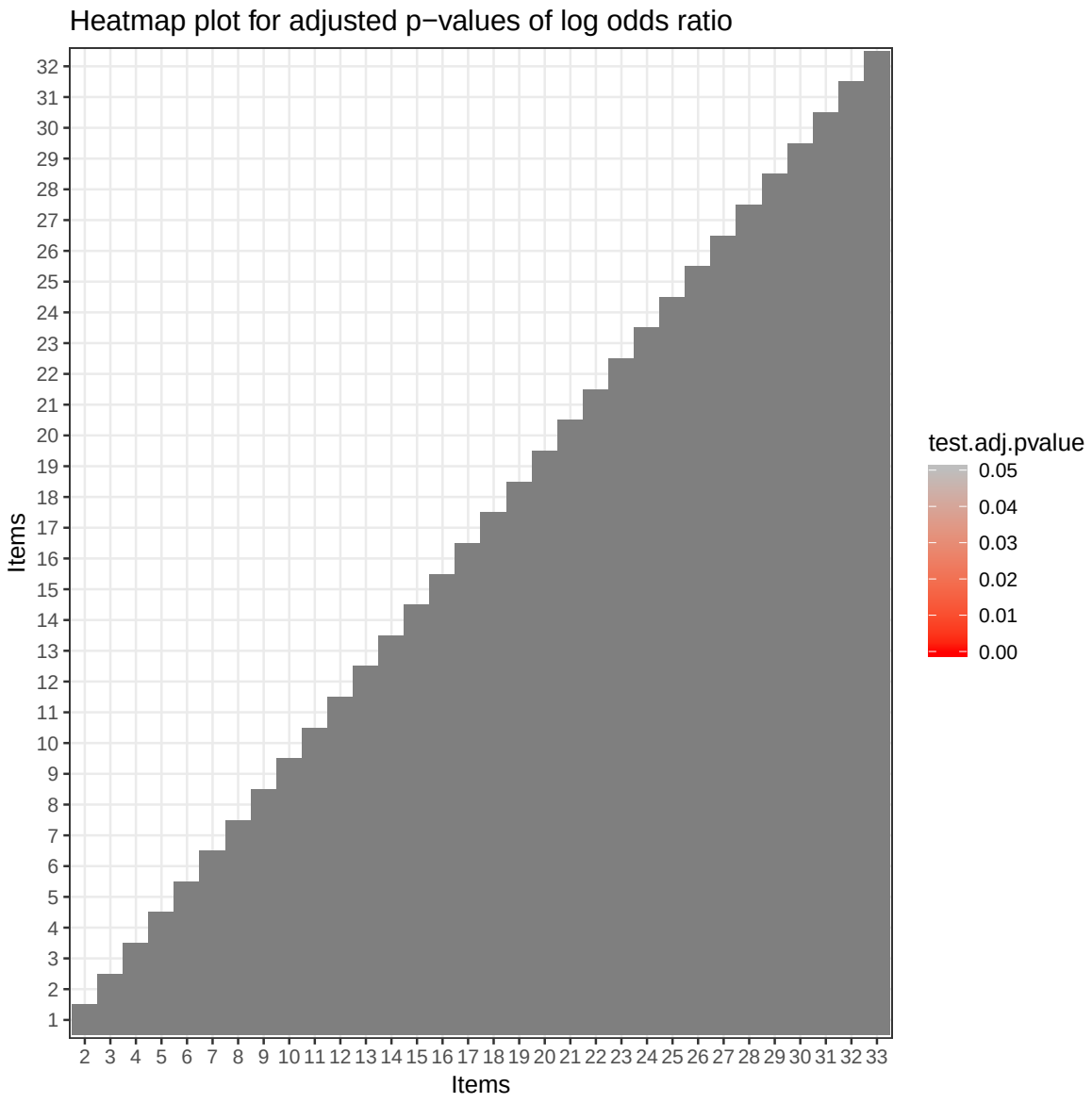
Modify the Q-matrix based on the suggested q-vectors of the MESA plots. Hint: To modify the Q-matrix, you can use `Q3[,]=c()`.

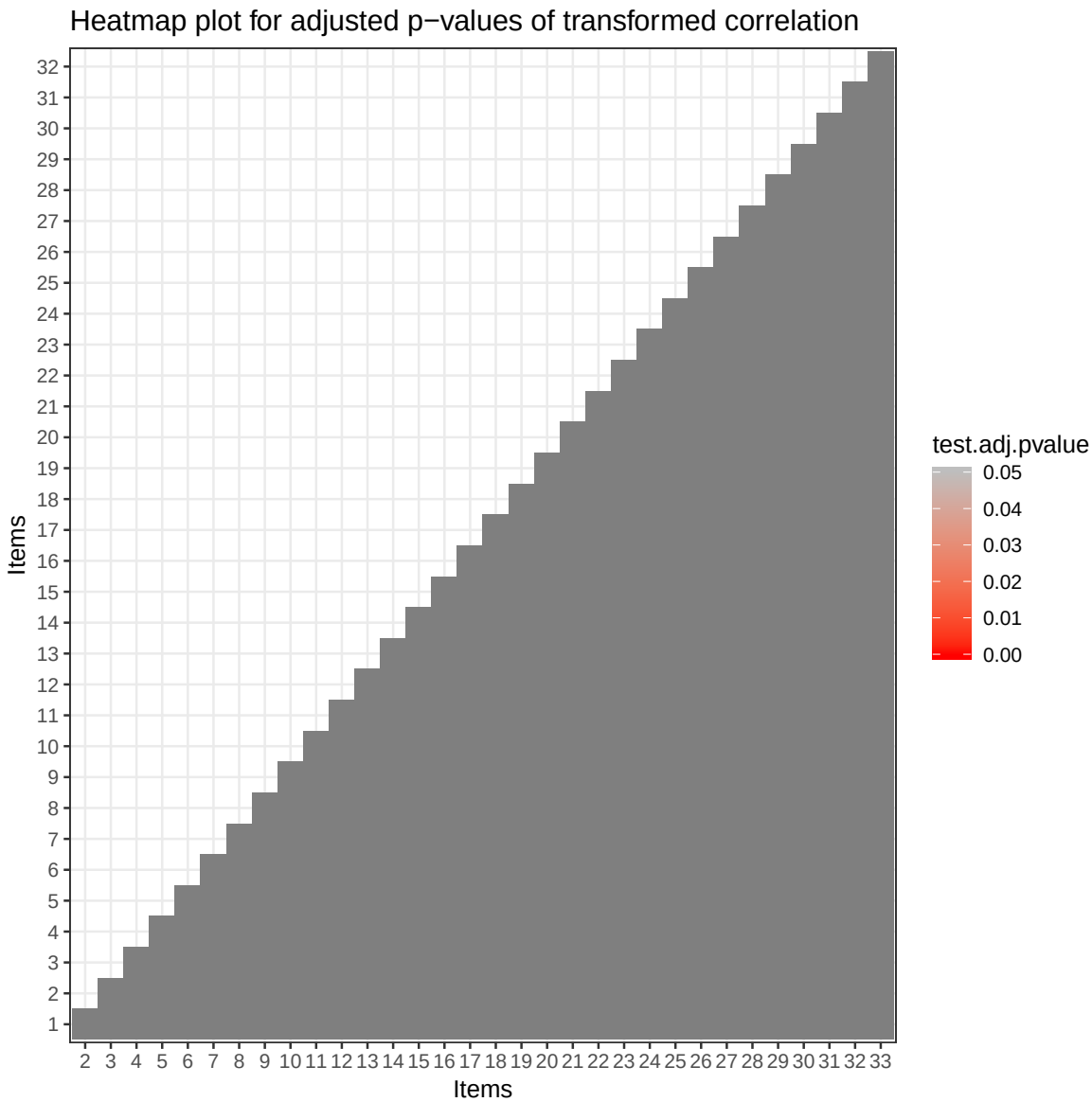
```
Q3[1,] = c(0,1,0,1)
Q3[33,] = c(0,0,0,1)
```

4. **Step 4: Model-data fit**
   Re-fit the G-DINA model with monotonicity constraints once again, but this time using the corrected Q-matrix. Check the model fit statistics using `heatplot`. Is there any misfitting item?

```
mod_rev <- GDINA(dat = data3, Q = Q3, model = "GDINA", mono.constraint = TRUE,
    verbose = 0)
summary(mod_rev)
```

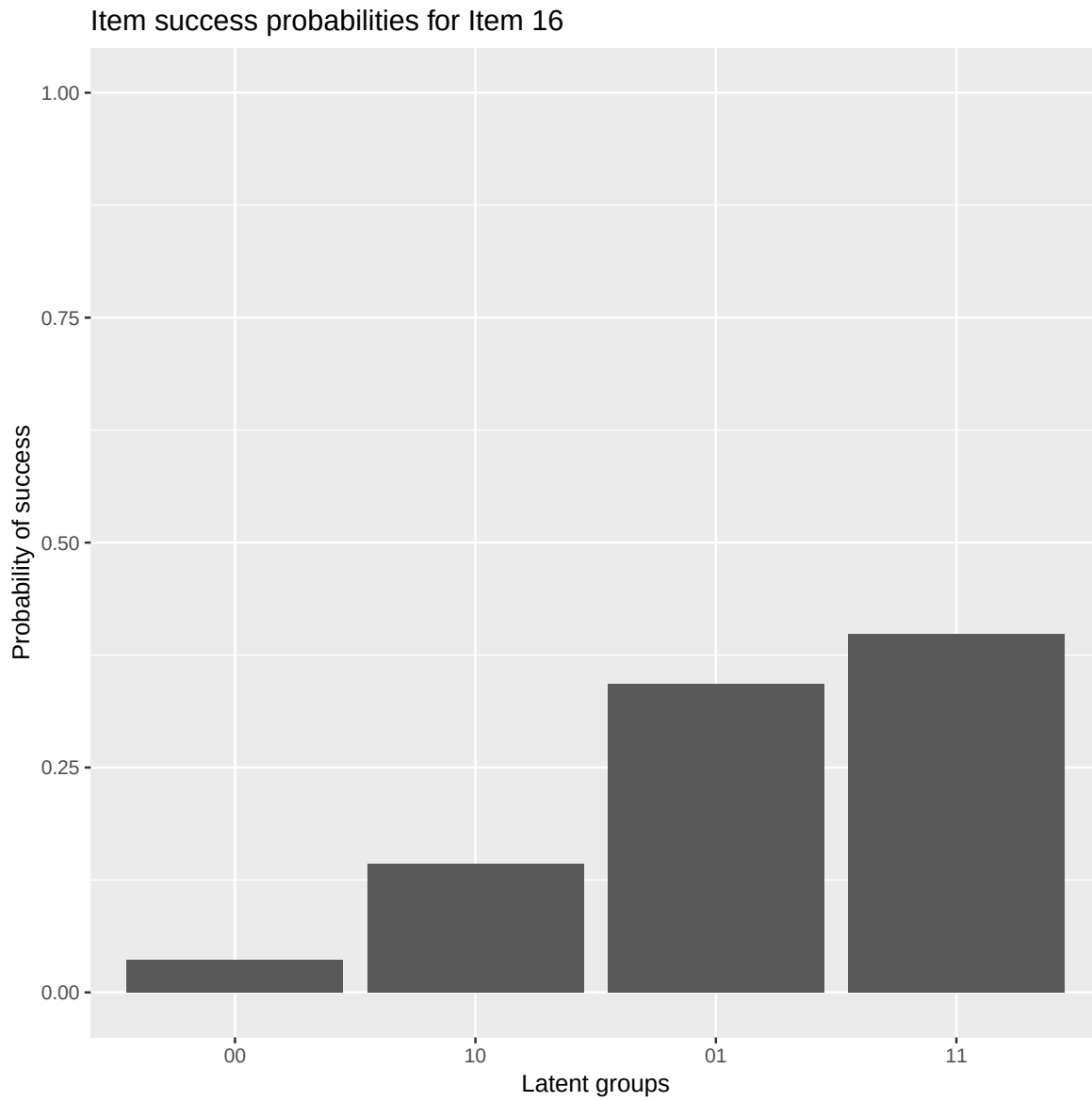
```
##
## Test Fit Statistics
##
## Loglik = -21101.38
## AIC    = 42412.76  | penalty   = 210
## BIC    = 42948.09  | penalty   = 745.33
## # par  = 105
##
## Attribute Prevalence
##
##    Level0 Level1
## A1 0.5176 0.4824
## A2 0.4849 0.5151
## A3 0.4812 0.5188
## A4 0.5225 0.4775

mod_rev

## Call:
## GDINA(dat = data3, Q = Q3, model = "GDINA", mono.constraint = TRUE,
##     verbose = 0)
##
##   GDINA version 2.1.15 (2018-6-6)
## ================================================
## Data
## ------------------------------------------------
## # of individuals    groups    items
##             1210         1       33
## ================================================
## Model
## ------------------------------------------------
## Fitted model(s)       = GDINA
## Attribute structure   = saturated
## Attribute level       = Dichotomous
## ================================================
## Estimation
## ------------------------------------------------
## Number of iterations  = 29
## For the final iteration:
##   Max abs change in item success prob. = 0.0001
##   Max abs change in mixing proportions = 0.0000
##   Change in -2 log-likelihood          = 0.0005
## Time used             = 3.762 secs

# Check model fit using heatplot
itemfit <- itemfit(mod_rev)
plot(itemfit)
```


Heatmap plot for adjusted p−values of log odds ratio
Items
Items
test.adj.pvalue
0.05
0.04
0.03
0.02
0.01
0.00



Plot the item response function (IRF) of Item 16 using the `plot` function. Because the disorders are being diagnosed by the items, one could argue that having one is enough for the symptom to manifest itself. Hence, it is interesting to determine whether fitting the DINO model to Item 16 while fitting the GDINA model for other items can improve the overall model. Afterwards, check the model fit statistics using heat plots. Is Item 16 problematic?

```
plot(x = mod_rev, what="IRF", item=16)
```




```
# Modify item 16 as "DINO" model,
models2 <- c(rep("GDINA",15),"DINO",rep("GDINA",17))
mod_rev2 <- GDINA(dat=data3,Q=Q3,model=models2,mono.constraint = TRUE,verbose =
summary(mod_rev2)

##
## Test Fit Statistics
##
## Loglik = -21124.65
## AIC    = 42455.30  | penalty   = 206
## BIC    = 42980.43  | penalty   = 731.13
## # par  = 103
##
## Attribute Prevalence
```

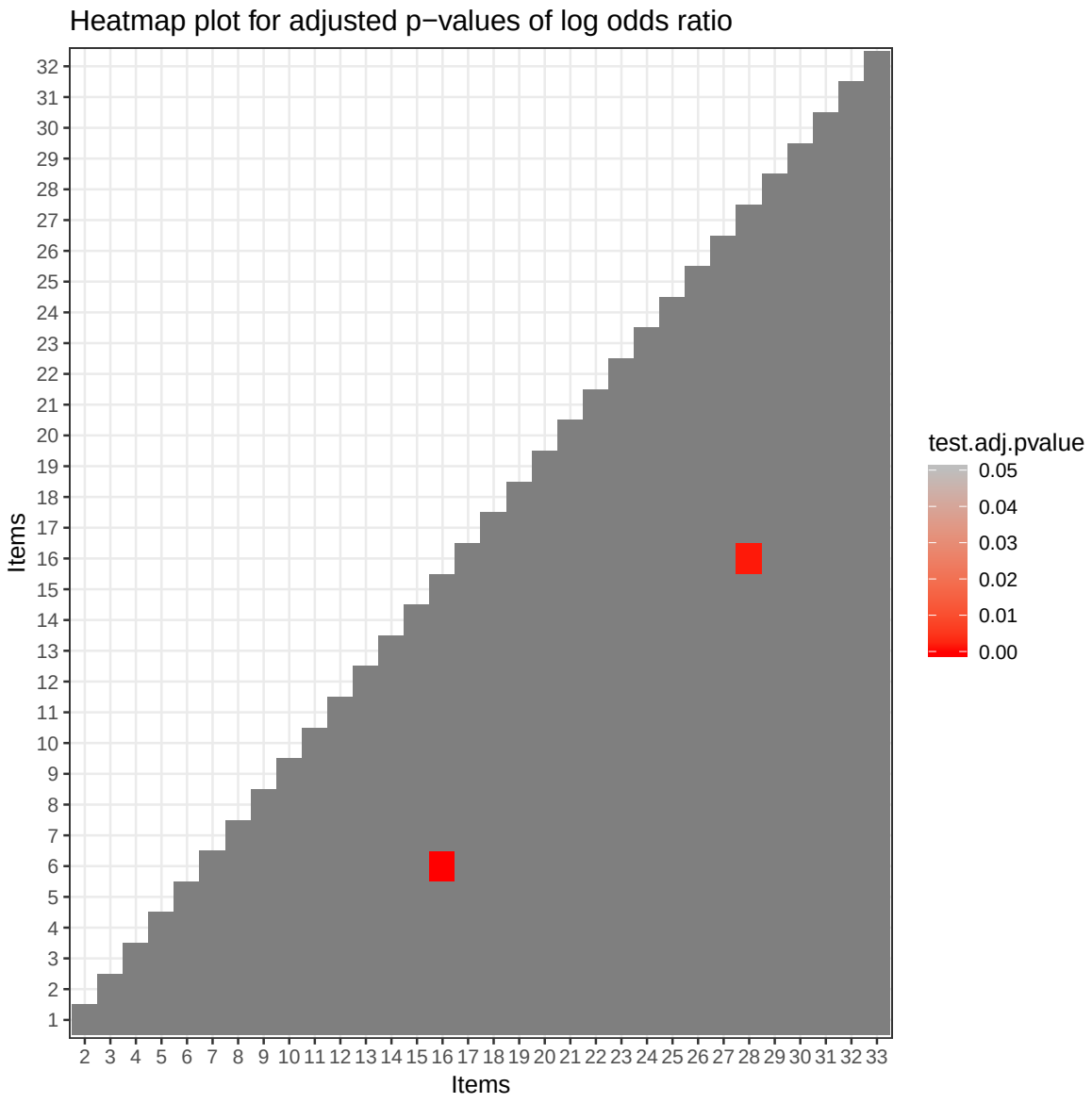
```
##
##    Level0 Level1
## A1 0.5175 0.4825
## A2 0.4912 0.5088
## A3 0.4809 0.5191
## A4 0.5224 0.4776
```

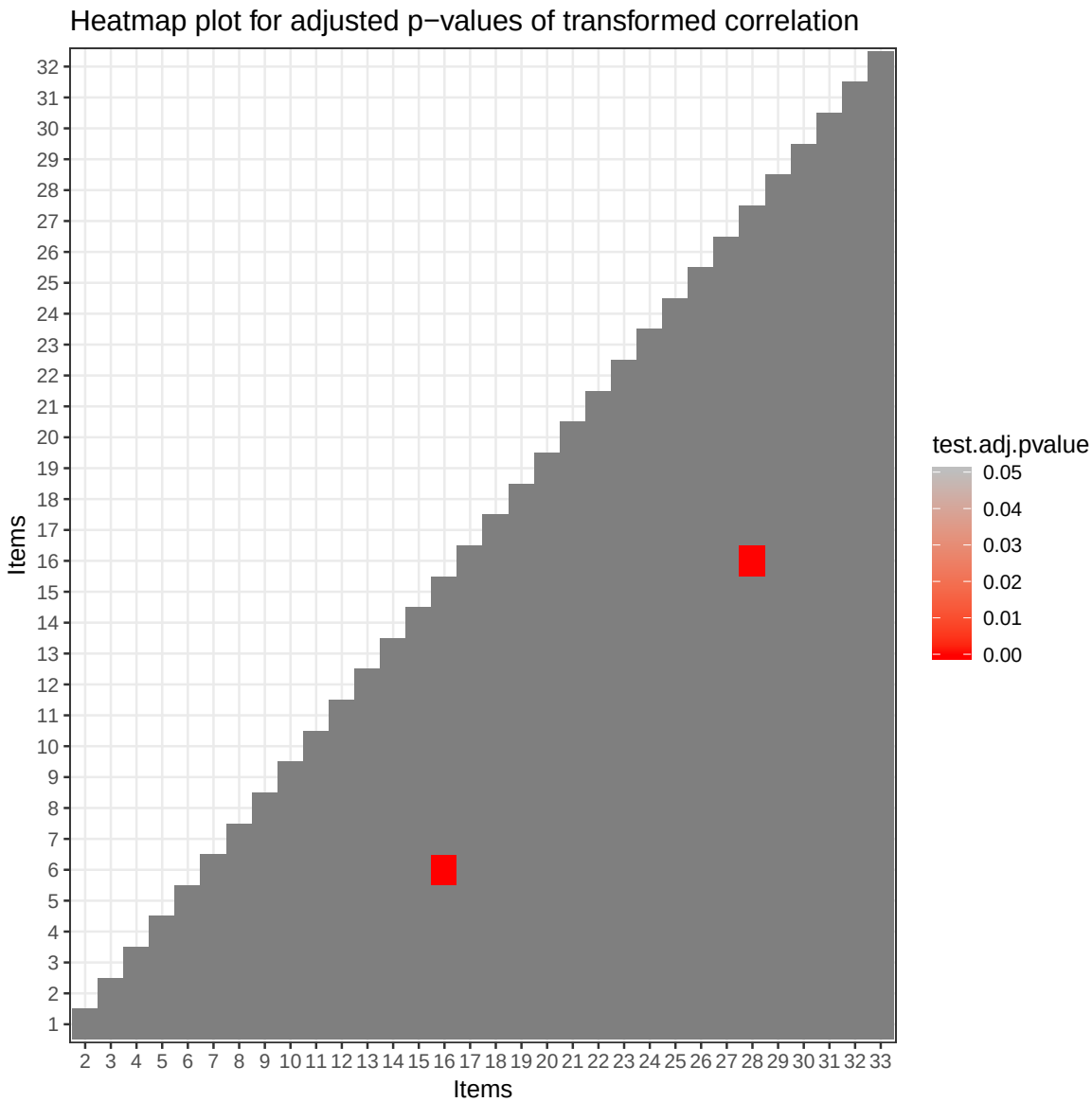
```
mod_rev2
```

```
## Call:
## GDINA(dat = data3, Q = Q3, model = models2, mono.constraint = TRUE,
##     verbose = 0)
##
##   GDINA version 2.1.15 (2018-6-6)
## ================================================
## Data
## ------------------------------------------------
## # of individuals    groups    items
##             1210         1       33
## ================================================
## Model
## ------------------------------------------------
## Fitted model(s)       = GDINA DINO
## Attribute structure   = saturated
## Attribute level       = Dichotomous
## ================================================
## Estimation
## ------------------------------------------------
## Number of iterations  = 29
## For the final iteration:
##   Max abs change in item success prob. = 0.0001
##   Max abs change in mixing proportions = 0.0000
##   Change in -2 log-likelihood          = 0.0004
## Time used             = 4.012 secs
```

Using the fitted model above, check the model fit statistics using heat plots. Is there any misfitting item?

```
itemfit2 <- itemfit(mod_rev2)
plot(itemfit2)
```


Heatmap plot for adjusted p−values of log odds ratio
Items
Items
test.adj.pvalue
0.05
0.04
0.03
0.02
0.01
0.00



Item 16 demonstrated misfit based on the heatplot.

5. **Step 5: Model comparison**

Fit the reduced or simpler models, namely, the DINA, DINO, ACDM, LLM, and RRUM, to the data. Compare these models with the fitted model in step (4). Is any of these reduced models fit as good as the saturated model? Perform a likelihood ratio (LR) test using the `anova` function.

```
mod_dina <- GDINA(dat=data3,Q=Q3, model="DINA",
                    mono.constraint = T,verbose = 0)
mod_dino <- GDINA(dat=data3,Q=Q3, model="DINO",
                  mono.constraint = T,verbose = 0)
mod_acdm <- GDINA(dat=data3,Q=Q3, model="ACDM",
```

```
                  mono.constraint = T,verbose = 0)
mod_rrum <- GDINA(dat=data3,Q=Q3, model="RRUM",
                  mono.constraint = T,verbose = 0)
mod_llm <- GDINA(dat=data3,Q=Q3, model="LLM",
                 mono.constraint = T,verbose = 0)

#Compare all reduced models with GDINA
anova(mod_dina, mod_dino, mod_acdm, mod_rrum, mod_llm, mod_rev)

##
## Information Criteria and Likelihood Ratio Test
##
##          #par    logLik Deviance      AIC      BIC   chisq df p-value
## mod_dina   81 -21633.50 43267.01 43429.01 43841.97 1064.24 24  <0.001
## mod_dino   81 -21528.71 43057.42 43219.42 43632.38  854.65 24  <0.001
## mod_acdm   93 -21141.51 42283.02 42469.02 42943.17   80.26 12  <0.001
## mod_rrum   93 -21267.01 42534.02 42720.02 43194.17  331.26 12  <0.001
## mod_llm    93 -21131.19 42262.38 42448.38 42922.52   59.61 12  <0.001
## mod_rev   105 -21101.38 42202.76 42412.76 42948.09
##
## Notes: In LR tests, models were tested against mod_rev
##        LR test(s) do NOT check whether models are nested or not.
```

Using the `modelcomp` function, conduct the Wald test for item fit evaluation. Based on the maximum p-value, choose the "best" model for each item. Afterwards, fit these models to the data. Then, using LR test, determine whether this model fits the data equally well as the saturated model.

```
wald_rev <- modelcomp(mod_rev)
wald_rev

##
##  Wald statistics for items requiring two or more attributes:
##             DINA     DINO    ACDM    LLM    RRUM
## Item 1  329.9882 138.0298  6.4922 3.4845 29.2646
## Item 2  299.8821 137.9617  2.6198 4.5828 22.7649
## Item 4   81.5175 149.4176  1.1168 2.8815  7.2439
## Item 7  181.2598 101.8750  0.0004 6.5305 20.4424
## Item 9  284.6382  69.3020 35.3018 4.4765 48.9665
## Item 12 161.7453 113.1871  0.8590 3.8183 17.7149
## Item 13  45.8429 117.3944 10.0796 1.2744  1.2630
## Item 16  73.3257  51.7127  0.8157 7.0087  8.5810
## Item 19 102.2771  56.2371  0.0007 4.6881 14.0540
## Item 23 192.8023 152.7983  4.2693 3.8967  9.8455
## Item 25 175.9981  89.1000  6.0858 2.9184 22.5222
## Item 27  49.8721 118.0860 13.9430 0.3322  0.3646
##
```

```
## p-values for items requiring two or more attributes:
##          DINA DINO   ACDM    LLM   RRUM
## Item 1      0    0 0.0108 0.0619 0.0000
## Item 2      0    0 0.1055 0.0323 0.0000
## Item 4      0    0 0.2906 0.0896 0.0071
## Item 7      0    0 0.9832 0.0106 0.0000
## Item 9      0    0 0.0000 0.0344 0.0000
## Item 12     0    0 0.3540 0.0507 0.0000
## Item 13     0    0 0.0015 0.2589 0.2611
## Item 16     0    0 0.3664 0.0081 0.0034
## Item 19     0    0 0.9786 0.0304 0.0002
## Item 23     0    0 0.0388 0.0484 0.0017
## Item 25     0    0 0.0136 0.0876 0.0000
## Item 27     0    0 0.0002 0.5644 0.5459
```

Fit the model using suggested models by the Wald-test.

```
#Fit the model based on Wald test
models_wald <- c("LLM","ACDM","GDINA","ACDM","GDINA",
                    "GDINA","ACDM","GDINA","GDINA","GDINA",
                    "GDINA","ACDM","LLM","GDINA","GDINA",
                    "ACDM","GDINA","GDINA","ACDM","GDINA",
                    "GDINA","GDINA","GDINA","GDINA","LLM",
                    "GDINA","LLM","GDINA","GDINA","GDINA",
                    "GDINA","GDINA","GDINA")

mod_wald <- GDINA(dat=data3,Q=Q3,model=models_wald,
                  mono.constraint = T,verbose = 0)
```

6. **Step 6: DIF Detection**
   In this dataset, the first 600 subjects are males (group 1), whereas the rest are females denoted by (group 2). Perform differential item functioning (DIF) analysis using the Wald and the LR test using the `dif` function. Is there any item exhibiting DIF?

```
#Perform DIF using Wald and LR tests
grp <- c(rep(1,600),rep(2,610))

difout_wald <- dif(dat=data3,Q=Q3,group=grp,
                   method="wald",mono.constraint=T)
difout_wald

##
## Differential Item Functioning Detection
##          Wald stat. df p.value adj.pvalue
## Item 1       3.8579  4  0.4256          1
```

```
## Item 2      4.1710  4  0.3834          1
## Item 3      5.3349  2  0.0694          1
## Item 4     87.4792  4  0.0000          0
## Item 5      0.3387  2  0.8442          1
## Item 6      4.5383  2  0.1034          1
## Item 7      7.1844  4  0.1265          1
## Item 8      0.5179  2  0.7719          1
## Item 9      1.2982  4  0.8617          1
## Item 10     1.1008  2  0.5767          1
## Item 11     0.7156  2  0.6992          1
## Item 12     0.9317  4  0.9200          1
## Item 13     1.1506  4  0.8862          1
## Item 14     0.2975  2  0.8618          1
## Item 15     3.8143  2  0.1485          1
## Item 16     0.3299  4  0.9878          1
## Item 17     1.3223  2  0.5163          1
## Item 18     4.4636  2  0.1073          1
## Item 19     4.8707  4  0.3008          1
## Item 20     0.3823  2  0.8260          1
## Item 21     0.5722  2  0.7512          1
## Item 22     3.4835  2  0.1752          1
## Item 23     1.5682  4  0.8145          1
## Item 24     0.7013  2  0.7042          1
## Item 25     3.0015  4  0.5576          1
## Item 26     1.5703  2  0.4560          1
## Item 27     2.8305  4  0.5866          1
## Item 28     4.1227  2  0.1273          1
## Item 29     0.3394  2  0.8439          1
## Item 30     1.5728  2  0.4555          1
## Item 31     2.3086  2  0.3153          1
## Item 32     2.9635  2  0.2272          1
## Item 33     3.0021  2  0.2229          1
##
## Note: adjusted pvalues are based on the bonferroni correction.
```

Because LR test requires refitting the model which takes a while, we only evaluate item 7 using the LR test.

```
difout_LR <- dif(dat=as.matrix(data3),Q=Q3,group=grp,method="LR",
                   mono.constraint=T, LR.type = "free.all",
                   difitem = 7)
difout_LR
```

```
##
## Differential Item Functioning Detection
##          neg2LL  LRstat df p.value adj.pvalue
```

```
## Item 7 40901.82 10.9361  4  0.0273     0.0273
##
## Note: adjusted pvalues are based on the bonferroni correction.
```

7. **Step 7: Classification Accuracy**
   A way to evaluate the usefulness of a CDM is its classification accuracy. Compute the classification accuracy for the final model resulting of the model comparison analysis

```
CA_all <- CA(mod_wald)
CA_all

## Classification Accuracy
##
## Test level accuracy =  0.8231
##
## Pattern level accuracy:
##
##   0000   1000   0100   0010   0001   1100   1010   1001   0110   0101
## 0.9231 0.8792 0.7906 0.8429 0.8474 0.8062 0.7444 0.7839 0.7935 0.7871
##   0011   1110   1101   1011   0111   1111
## 0.7013 0.8202 0.7633 0.7668 0.8406 0.8551
##
## Attribute level accuracy:
##
##     A1     A2     A3     A4
## 0.9717 0.9256 0.9353 0.9626
```

8. **Step 8: Item Selection**

   Compute the discrimination index of each item based on the guessing and slip parameter estimates. Which two items are the most high and low discrimination? Hint: Use the `coef(obj,what = "gs")` to obtain the guessing and slip parameters, and then, compute the discrimination index as **1-g-s**.

```
#Discrimination Index
disc <- extract(mod_wald,"discrim")
disc

##         P(1)-P(0)         GDI
## Item 1  0.9011000 0.110955262
## Item 2  0.8816008 0.110065440
## Item 3  0.1693889 0.007169020
## Item 4  0.7345557 0.075266500
## Item 5  0.2300927 0.013228042
## Item 6  0.7729392 0.149037574
```

```
## Item 7  0.7631689 0.078864496
## Item 8  0.5017296 0.062852480
## Item 9  0.7981833 0.094856933
## Item 10 0.5973468 0.089090611
## Item 11 0.6233342 0.097010968
## Item 12 0.7391540 0.075436894
## Item 13 0.6305169 0.058419944
## Item 14 0.2216938 0.012271173
## Item 15 0.3152027 0.024806115
## Item 16 0.3762542 0.023179748
## Item 17 0.4795749 0.057424330
## Item 18 0.5300879 0.070158272
## Item 19 0.6003730 0.048830216
## Item 20 0.1951189 0.009505554
## Item 21 0.3838858 0.036794866
## Item 22 0.4654126 0.054035772
## Item 23 0.7982554 0.089744008
## Item 24 0.4995229 0.062300843
## Item 25 0.6992954 0.077051576
## Item 26 0.5982611 0.089364419
## Item 27 0.5109845 0.038956578
## Item 28 0.5537079 0.076483272
## Item 29 0.5552175 0.076967129
## Item 30 0.5805042 0.084138307
## Item 31 0.3508279 0.030730322
## Item 32 0.3682517 0.033858884
## Item 33 0.5036740 0.063285495
```

Remove the two items with the lowest discrimination indices. Fit the saturated model and obtain its estimated attribute patterns. Afterwards, compute the classification accuracy. What do you observe when these rates are compared with those in step (7)? Hint: The two items with the lowest discrimination indices are 0_ and 2_.

```
#Deleting two low discriminating items
data_low <- data3[,c(-3,-20)]
Q_low <- Q3[c(-3,-20),]

#Refitting the data without items 3 and 20
mod_low <- GDINA(dat=data_low,Q=Q_low,model=models_wald[c(-3,-20)],
mono.constraint = TRUE,verbose = 0)

#Classification rate
CA_low <- CA(mod_low)
CA_low

## Classification Accuracy
```

```
##
## Test level accuracy =  0.8159
##
## Pattern level accuracy:
##
##   0000   1000   0100   0010   0001   1100   1010   1001   0110   0101
## 0.9028 0.8872 0.7962 0.8337 0.8638 0.7767 0.7343 0.7774 0.7991 0.7610
##   0011   1110   1101   1011   0111   1111
## 0.6845 0.8154 0.7368 0.7693 0.8447 0.8482
##
## Attribute level accuracy:
##
##     A1     A2     A3     A4
## 0.9718 0.9200 0.9313 0.9616
```

Re-do step (7) but this time removing the two most highly discriminating items. What can you observe when these rates are compared with previous results? Hint: The two items with the highest discrimination indices are _1 and _2.

```
#Deleting two highly discriminating items
data_high <- data3[,c(-1,-2)]
Q_high <- Q3[c(-1,-2),]

#Refitting the data without items 1 and 2
mod_high <- GDINA(dat=data_high, Q=Q_high, models=models_wald[c(-1,-2)],
mono.constraint = TRUE, verbose=0)

#classification Rate
CA_high <- CA(mod_high)
CA_high

## Classification Accuracy
##
## Test level accuracy =  0.7909
##
## Pattern level accuracy:
##
##   0000   1000   0100   0010   0001   1100   1010   1001   0110   0101
## 0.8851 0.8502 0.7592 0.7820 0.8016 0.7596 0.7945 0.6755 0.7350 0.7445
##   0011   1110   1101   1011   0111   1111
## 0.7046 0.7903 0.7324 0.7026 0.8226 0.8549
##
## Attribute level accuracy:
##
##     A1     A2     A3     A4
## 0.9717 0.8943 0.9354 0.9625
```

Let's compare the classification rates by attaching them using `rbind`.

```
  #Summary of Classification Accuracy
  #Test-level Classification Accuracy
  round(rbind("all"=CA_all$tau,
              "no low"=CA_low$tau,
              "no high"=CA_high$tau),2)

##          [,1]
## all      0.82
## no low   0.82
## no high  0.79

#Attribute-level Classification Accuracy
round(rbind("all"=CA_all$tau_k,
            "no low"=CA_low$tau_k,
            "no high"=CA_high$tau_k),2)

##            A1   A2   A3   A4
## all      0.97 0.93 0.94 0.96
## no low   0.97 0.92 0.93 0.96
## no high  0.97 0.89 0.94 0.96

#Pattern-level Classification Accuracy
round(rbind("all"=CA_all$tau_l,
            "no low"=CA_low$tau_l,
            "no high"=CA_high$tau_l),2)

##          0000 1000 0100 0010 0001 1100 1010 1001 0110 0101 0011 1110 1101
## all      0.92 0.88 0.79 0.84 0.85 0.81 0.74 0.78 0.79 0.79 0.70 0.82 0.76
## no low   0.90 0.89 0.80 0.83 0.86 0.78 0.73 0.78 0.80 0.76 0.68 0.82 0.74
## no high  0.89 0.85 0.76 0.78 0.80 0.76 0.79 0.68 0.74 0.74 0.70 0.79 0.73
##          1011 0111 1111
## all      0.77 0.84 0.86
## no low   0.77 0.84 0.85
## no high  0.70 0.82 0.85
```