

Appendix: Practical Implementation and Extended Models - A Comprehensive Guide to the exametrika Package

Koji Kosugi*

Kojiro Shojima†

Introduction

The **exametrika** package implements statistical models for test data engineering introduced by Shojima (2022), providing a comprehensive framework for analyzing educational test data. This document focuses on two key approaches: Latent Rank Analysis (LRA) for ordinal ability assessment and Biclustering for simultaneous clustering of examinees and items. LRA extends traditional Latent Class Analysis by incorporating ordered structures, while Biclustering enables the exploration of complex response patterns through two-mode clustering. The package implements efficient EM algorithms and rich visualization tools.

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

Data Preparation and Conversion

The most important step in using the **exametrika** package is converting your data into a format that can be processed by the package (**exametrika** class). This conversion is done using the **dataFormat** function.

The exametrika Class

The **exametrika** class is a data structure that includes the following information:

- **U**: Response data matrix (binary 0/1 data by default)
- **Z**: Missing value indicator matrix (1 for observed values, 0 for missing values)
- **ItemLabel**: List of item names
- **ID**: List of examinee IDs
- **response.type**: Data type (“binary”, “nominal”, “ordinal”, or “rated”)

Using Sample Datasets

The **exametrika** package includes several sample datasets that have already been converted to the **exametrika** class format.

```
# List of sample datasets included in the package  
data(package = "exametrika")  
  
# Example: Loading J15S500 (15 items, 500 samples)  
data(J15S500)  
  
# Checking the data structure  
head(J15S500$U, 3) # First 3 rows of the response data matrix
```

*Senshu University, School of Human Sciences, kosugi@psy.senshu-u.ac.jp

†University Entrance Examination Center, Research and Development Department, shojima@rd.dnc.ac.jp

```
##      Item01 Item02 Item03 Item04 Item05 Item06 Item07 Item08 Item09 Item10
## [1,]      0      1      1      0      1      1      0      0      0      1
## [2,]      1      1      1      1      1      1      0      1      0      1
## [3,]      1      1      1      1      1      1      0      0      0      1
##      Item11 Item12 Item13 Item14 Item15
## [1,]      1      0      1      0      1
## [2,]      0      0      1      0      1
## [3,]      0      0      1      1      1
```

```
head(J15S500$Z, 3) # First 3 rows of the missing value indicator matrix
```

```
##      Item01 Item02 Item03 Item04 Item05 Item06 Item07 Item08 Item09 Item10
## [1,]      1      1      1      1      1      1      1      1      1      1
## [2,]      1      1      1      1      1      1      1      1      1      1
## [3,]      1      1      1      1      1      1      1      1      1      1
##      Item11 Item12 Item13 Item14 Item15
## [1,]      1      1      1      1      1
## [2,]      1      1      1      1      1
## [3,]      1      1      1      1      1
```

```
J15S500$response.type # Data type
```

```
## [1] "binary"
```

These sample datasets are useful for testing the package's functionality before analyzing your own data.

Creating and Converting Data

Converting from a Data Frame

```
# Numeric ID column
data1 <- data.frame(
  id = 1:5,
  item1 = c(0, 1, 1, 0, 1),
  item2 = c(1, 0, 1, 0, 0)
)
result1 <- dataFormat(data1)
print(result1)
```

```
## Response Type: binary
## Binary Response Pattern
##      item1 item2
## [1,]      0      1
## [2,]      1      0
## [3,]      1      1
## [4,]      0      0
## [5,]      1      0
##
## Missing Pattern
##      item1 item2
## [1,]      1      1
## [2,]      1      1
## [3,]      1      1
## [4,]      1      1
## [5,]      1      1
##
```

```
## Weight
## [1] 1 1
```

In this code, the first column is automatically recognized as the ID column, and the remaining columns are treated as items.

Specifying the ID Column

If the ID column is not the first column, you can specify its position using the `id` parameter.

```
# String ID column in the third position
data2 <- data.frame(
  item1 = c(0, 1, 1, 0, 1),
  item2 = c(1, 0, 1, 0, 99),
  id = paste0("student", 1:5)
)
result2 <- dataFormat(data2, id = 3, na = 99)
print(result2)
```

```
## Response Type: binary
## Binary Response Pattern
##      item1 item2
## [1,]     0     1
## [2,]     1     0
## [3,]     1     1
## [4,]     0     0
## [5,]     1    -1
##
## Missing Pattern
##      item1 item2
## [1,]     1     1
## [2,]     1     1
## [3,]     1     1
## [4,]     1     1
## [5,]     1     0
##
## Weight
## [1] 1 1
```

Handling Missing Values

Missing values are specified with the `na` parameter. By default, R's standard missing value (`NA`) is used.

```
# Data with NA values
data3 <- data.frame(
  id = c(1, 2, 3, 4, 5),
  item1 = c(1, NA, 2, 4, 1),
  item2 = c(1, 3, 2, 4, 5)
)
result3 <- dataFormat(data3)
print(result3)
```

```
## Response Type: nominal
## Polytomous Response Pattern ( nominal )
##      item1 item2
## [1,]     1     1
## [2,]    -1     3
```

```
## [3,]      2      2
## [4,]      4      4
## [5,]      1      5
##
## Missing Pattern
##      item1 item2
## [1,]      1      1
## [2,]      0      1
## [3,]      1      1
## [4,]      1      1
## [5,]      1      1
##
## Weight
## [1] 1 1
```

```
# Data with custom missing value (-999)
data4 <- data.frame(
  id = c(1, 2, 3, 4, 5),
  item1 = c(1, -999, 2, 4, 1),
  item2 = c(1, 3, 2, 4, -999)
)
result4 <- dataFormat(data4, na = -999)
print(result4)
```

```
## Response Type: nominal
## Polytomous Response Pattern ( nominal )
##      item1 item2
## [1,]      1      1
## [2,]     -1      3
## [3,]      2      2
## [4,]      4      4
## [5,]      1     -1
##
## Missing Pattern
##      item1 item2
## [1,]      1      1
## [2,]      0      1
## [3,]      1      1
## [4,]      1      1
## [5,]      1      0
##
## Weight
## [1] 1 1
```

Loading from CSV Files

When loading data from a CSV file, first use the `read.csv` function to load the data, then convert it using the `dataFormat` function.

```
# Loading from a CSV file
test_data <- read.csv("test_data.csv")
result <- dataFormat(test_data, na = -99) # Specifying -99 as missing value
```

A Complete Example with Real Data

Let's examine a practical example where we create test data for 5 examinees, save it as a CSV file, then load and analyze the data.

```
# Create sample test data
sample_test_data <- data.frame(
  student_id = c("S001", "S002", "S003", "S004", "S005"),
  Q1 = c(1, 0, 1, 1, 0),
  Q2 = c(0, 1, 1, 0, 0),
  Q3 = c(1, 1, 0, 1, -99), # -99 represents missing value
  Q4 = c(0, 0, 1, 1, 1),
  Q5 = c(1, 0, 0, -99, 1) # -99 represents missing value
)

# View the raw data
head(sample_test_data)

##   student_id Q1 Q2  Q3 Q4  Q5
## 1      S001  1  0   1  0   1
## 2      S002  0  1   1  0   0
## 3      S003  1  1   0  1   0
## 4      S004  1  0   1  1 -99
## 5      S005  0  0 -99  1   1

# Save data to a CSV file
write.csv(sample_test_data, "sample_test.csv", row.names = FALSE)

# Now read it back as if it was an external file
read_test_data <- read.csv("sample_test.csv")

# Convert to exametrika format
test_data_exam <- dataFormat(read_test_data, id = 1, na = -99)

# Display the converted data
print(test_data_exam)

## Response Type: binary
## Binary Response Pattern
##      Q1 Q2 Q3 Q4 Q5
## [1,]  1  0  1  0  1
## [2,]  0  1  1  0  0
## [3,]  1  1  0  1  0
## [4,]  1  0  1  1 -1
## [5,]  0  0 -1  1  1
##
## Missing Pattern
##      Q1 Q2 Q3 Q4 Q5
## [1,]  1  1  1  1  1
## [2,]  1  1  1  1  1
## [3,]  1  1  1  1  1
## [4,]  1  1  1  1  0
## [5,]  1  1  0  1  1
##
## Weight
## [1] 1 1 1 1 1
```

```

# Remove created csv file
file.remove("sample_test.csv")

## [1] TRUE

# Simulated example (creating a virtual CSV-like dataset)
test_data <- data.frame(
  ID = paste0("S", 1:10),
  Q1 = sample(c(0, 1, -99), 10, replace = TRUE),
  Q2 = sample(c(0, 1, -99), 10, replace = TRUE),
  Q3 = sample(c(0, 1, -99), 10, replace = TRUE)
)
result <- dataFormat(test_data, na = -99)
print(result)

## Response Type: binary
## Binary Response Pattern
##      Q1 Q2 Q3
## [1,]  1 -1  0
## [2,]  0  0  1
## [3,]  1  0  0
## [4,] -1  1 -1
## [5,] -1  1 -1
## [6,] -1  1  1
## [7,]  0  0  0
## [8,] -1  1 -1
## [9,]  1 -1  0
## [10,] 1  0 -1
##
## Missing Pattern
##      Q1 Q2 Q3
## [1,]  1  0  1
## [2,]  1  1  1
## [3,]  1  1  1
## [4,]  0  1  0
## [5,]  0  1  0
## [6,]  0  1  1
## [7,]  1  1  1
## [8,]  0  1  0
## [9,]  1  0  1
## [10,] 1  1  0
##
## Weight
## [1] 1 1 1

```

Specifying Data Types

The `exametrika` package can handle various types of data. The data type is specified using the `response.type` parameter.

- **binary**: Binary data (0/1) - default
- **nominal**: Nominal scale data (categorical)
- **ordinal**: Ordinal scale data (Likert scale, etc.)
- **rated**: Nominal scale data with correct answers (multiple-choice items, etc.)

```

# Ordinal scale data example
ordinal_data <- data.frame(
  id = 1:5,
  Q1 = sample(1:5, 5, replace = TRUE),
  Q2 = sample(1:5, 5, replace = TRUE)
)
ordinal_result <- dataFormat(ordinal_data, response.type = "ordinal")
print(ordinal_result)

## Response Type: ordinal
## Polytomous Response Pattern ( ordinal )
##      Q1 Q2
## [1,]  4  5
## [2,]  5  3
## [3,]  2  3
## [4,]  5  1
## [5,]  1  5
##
## Missing Pattern
##      Q1 Q2
## [1,]  1  1
## [2,]  1  1
## [3,]  1  1
## [4,]  1  1
## [5,]  1  1
##
## Weight
## [1] 1 1

# Multiple-choice data with correct answers
rated_data <- data.frame(
  id = 1:5,
  Q1 = sample(1:4, 5, replace = TRUE),
  Q2 = sample(1:4, 5, replace = TRUE)
)
# Correct answers: Q1 = option 4, Q2 = option 2
rated_result <- dataFormat(rated_data, response.type = "rated", CA = c(4, 2))
print(rated_result)

## Response Type: rated
## Polytomous Response Pattern ( rated )
##      Q1 Q2
## [1,]  4  3
## [2,]  1  2
## [3,]  3  4
## [4,]  2  1
## [5,]  2  2
##
## Correct Answers
## [1] 4 2
##
## Missing Pattern
##      Q1 Q2
## [1,]  1  1
## [2,]  1  1

```

```
## [3,] 1 1
## [4,] 1 1
## [5,] 1 1
##
## Weight
## [1] 1 1
```

Using the Weight (w) Parameter

The `w` parameter is used to specify item weights. It is primarily used in Classical Test Theory (CTT) analysis to adjust the contribution of each item to the total score. However, in modern test theories such as Latent Rank Analysis (LRA) and Biclustering, item characteristics (such as difficulty) are estimated separately, so this parameter is typically not used.

```
# Example of specifying weights (used in CTT)
weighted_data <- data.frame(
  id = 1:5,
  Q1 = c(1, 0, 1, 0, 1),
  Q2 = c(0, 1, 1, 0, 0)
)
# Weight for Q1 = 2, weight for Q2 = 1
weighted_result <- dataFormat(weighted_data, w = c(3, 2))
weighted_result
```

```
## Response Type: binary
## Binary Response Pattern
##      Q1 Q2
## [1,] 1 0
## [2,] 0 1
## [3,] 1 1
## [4,] 0 0
## [5,] 1 0
##
## Missing Pattern
##      Q1 Q2
## [1,] 1 1
## [2,] 1 1
## [3,] 1 1
## [4,] 1 1
## [5,] 1 1
##
## Weight
## [1] 3 2
```

```
TestStatistics(weighted_result)
```

```
## Test Statistics
##              value
## TestLength    2.0000000
## SampleSize    5.0000000
## Mean          2.6000000
## SEofMean      0.8124038
## Variance      3.3000000
## SD            1.8165902
## Skewness     -0.1790422
## Kurtosis     -0.7314050
```

```
## Min      0.0000000
## Max      5.0000000
## Range    5.0000000
## Q1.25%   2.0000000
## Median.50% 3.0000000
## Q3.75%   3.0000000
## IQR       1.0000000
## Stanine.4% 0.3200000
## Stanine.11% 0.8800000
## Stanine.23% 1.8400000
## Stanine.40% 2.6000000
## Stanine.60% 3.0000000
## Stanine.77% 3.1600000
## Stanine.89% 4.1200000
## Stanine.96% 4.6800000
```

Checking Data

To check the contents of the converted data, simply enter the variable name or use the `summary` function.

```
# Basic overview
```

```
summary(result1)
```

```
##           Length Class  Mode
## ID           5    -none- numeric
## ItemLabel     2    -none- character
## Z            10    -none- numeric
## w             2    -none- numeric
## response.type  1    -none- character
## categories     2    -none- numeric
## U            10    -none- numeric
```

```
# Checking individual components
```

```
head(result1$U, 3) # First 3 rows of the response data matrix
```

```
##      item1 item2
## [1,]     0     1
## [2,]     1     0
## [3,]     1     1
```

```
head(result1$Z, 3) # First 3 rows of the missing value indicator matrix
```

```
##      item1 item2
## [1,]     1     1
## [2,]     1     1
## [3,]     1     1
```

```
result1$ItemLabel # Item names
```

```
## [1] "item1" "item2"
```

```
result1$ID # Examinee IDs
```

```
## [1] 1 2 3 4 5
```

```
result1$response.type # Data type
```

```
## [1] "binary"
```

Analysis Examples

Once you have converted your data to the `exametrika` class format, you can perform various analyses. In this section, we'll demonstrate the major analytical methods using sample datasets included in the `exametrika` package.

Latent Rank Analysis (LRA)

Latent Rank Analysis (LRA) is a statistical method that classifies examinees into ordered ranks based on their response patterns. It extends traditional Latent Class Analysis by incorporating an ordinal structure.

```
# Load sample data
data(J15S500) # 15 items, 500 samples binary dataset

# Run LRA with 6 ranks
lra_result <- LRA(J15S500, nrank = 6)

# Examine the Item Reference Profile (IRP)
# IRP shows the probability of correct responses for each item at each rank
head(lra_result$IRP)
```

```
##           IRP1      IRP2      IRP3      IRP4      IRP5      IRP6
## Item01 0.5850570 0.6318654 0.7080067 0.7866549 0.8530561 0.8977529
## Item02 0.5247272 0.6290452 0.7554820 0.8454942 0.8829038 0.8750385
## Item03 0.6134218 0.6095081 0.7082092 0.7726317 0.8009246 0.8386955
## Item04 0.4406175 0.6072562 0.7937016 0.8821678 0.9394150 0.9763276
## Item05 0.6465232 0.7452358 0.8214571 0.8366280 0.8623113 0.9052983
## Item06 0.6470858 0.7747914 0.9109422 0.9674623 0.9633072 0.9154799
```

Model Fit Indices

Model fit indices help evaluate the appropriateness of the specified number of ranks:

```
# Display test fit indices
lra_result$TestFitIndices

##   model_log_like bench_log_like null_log_like model_Chi_sq null_Chi_sq model_df
## 1      -3857.982      -3560.005      -4350.217      595.9544      1580.424 138.4909
##   null_df      NFI      RFI      IFI      TLI      CFI      RMSEA      AIC
## 1      195 0.6229148 0.469051 0.6827429 0.5350704 0.6698025 0.0813612 318.9726
##      CAIC      BIC
## 1 -264.989 -264.7123
```

```
# Display item fit indices
lra_result$ItemFitIndices
```

```
##           model_log_like bench_log_like null_log_like model_Chi_sq null_Chi_sq
## Item01      -264.4952      -240.1896      -283.3432      48.61112      86.30724
## Item02      -253.1410      -235.4364      -278.9486      35.40918      87.02454
## Item03      -282.7855      -260.9064      -293.5981      43.75810      65.38341
## Item04      -207.0823      -192.0718      -265.9618      30.02094     147.77999
## Item05      -234.9021      -206.5372      -247.4032      56.72968      81.73195
## Item06      -168.2177      -153.9397      -198.8174      28.55603      89.75530
## Item07      -250.8635      -228.3788      -298.3455      44.96957     139.93348
## Item08      -312.6207      -293.2252      -338.7888      38.79100      91.12723
## Item09      -317.6005      -300.4923      -327.8422      34.21627      54.69970
## Item10      -309.6535      -288.1984      -319.8497      42.91038      63.30265
```

```
## Item11      -242.8213      -224.0855      -299.2653      37.47153      150.35960
## Item12      -236.5220      -214.7967      -293.5981      43.45053      157.60288
## Item13      -287.7819      -262.0307      -328.3959      51.50235      132.73044
## Item14      -221.7023      -204.9528      -273.2123      33.49910      136.51897
## Item15      -267.7930      -254.7637      -302.8469      26.05866      96.16648
##      model_df null_df      NFI      RFI      IFI      TLI      CFI
## Item01 9.232727      13 0.4367666 0.20694789 0.4890867 0.24364717 0.4628308
## Item02 9.232727      13 0.5931127 0.42708853 0.6635063 0.50209273 0.6463814
## Item03 9.232727      13 0.3307462 0.05766741 0.3851300 0.07197874 0.3409102
## Item04 9.232727      13 0.7968539 0.71396319 0.8499558 0.78282743 0.8457619
## Item05 9.232727      13 0.3059057 0.02269113 0.3448625 0.02698294 0.3089538
## Item06 9.232727      13 0.6818458 0.55202780 0.7600263 0.64552443 0.7482480
## Item07 9.232727      13 0.6786361 0.54750848 0.7265751 0.60358203 0.7184601
## Item08 9.232727      13 0.5743205 0.40062848 0.6390689 0.46729115 0.6216650
## Item09 9.232727      13 0.3744706 0.11923286 0.4505122 0.15640405 0.4008699
## Item10 9.232727      13 0.3221392 0.04554852 0.3771462 0.05731988 0.3304994
## Item11 9.232727      13 0.7507872 0.64909978 0.7999048 0.71053194 0.7944170
## Item12 9.232727      13 0.7243037 0.61181008 0.7693754 0.66681266 0.7633671
## Item13 9.232727      13 0.6119779 0.45365137 0.6577295 0.50290758 0.6469601
## Item14 9.232727      13 0.7546195 0.65449571 0.8093559 0.72337942 0.8035414
## Item15 9.232727      13 0.7290255 0.61845858 0.8064511 0.71513168 0.7976837
##      RMSEA      AIC      CAIC      BIC
## Item01 0.09245145 30.145665 -8.7851095 -8.7666625
## Item02 0.07537722 16.943727 -21.9870470 -21.9686000
## Item03 0.08656731 25.292645 -13.6381295 -13.6196825
## Item04 0.06717277 11.555481 -27.3752933 -27.3568463
## Item05 0.10153543 38.264230 -0.6665442 -0.6480972
## Item06 0.06476277 10.090574 -28.8402004 -28.8217534
## Item07 0.08807300 26.504114 -12.4266601 -12.4082131
## Item08 0.08009847 20.325543 -18.6052309 -18.5867839
## Item09 0.07363966 15.750817 -23.1799572 -23.1615102
## Item10 0.08549794 24.444926 -14.4858478 -14.4674008
## Item11 0.07829029 19.006078 -19.9246960 -19.9062490
## Item12 0.08618085 24.985073 -13.9457014 -13.9272544
## Item13 0.09578531 33.036895 -5.8938789 -5.8754319
## Item14 0.07257501 15.033642 -23.8971320 -23.8786850
## Item15 0.06043302 7.593204 -31.3375698 -31.3191228
```

These indices compare the current model to saturated and null models. Important metrics include: - NFI, RFI, IFI, TLI, CFI: Values close to 1 indicate good fit - RMSEA: Values less than 0.08 suggest reasonable fit - AIC, BIC, CAIC: Lower values indicate better models

Examining Individual Results

We can also examine rank membership profiles for individual examinees:

```
# Display rank membership for the first few examinees
head(lra_result$Students)
```

```
##      Membership 1 Membership 2 Membership 3 Membership 4 Membership 5
## Student001 0.2704649921 0.357479353 0.27632327 0.084988078 0.010069050
## Student002 0.0276546965 0.157616072 0.47438958 0.279914853 0.053715813
## Student003 0.0228189795 0.138860955 0.37884545 0.284817610 0.120794858
## Student004 0.0020140858 0.015608542 0.09629429 0.216973334 0.362406292
## Student005 0.5582996437 0.397431414 0.03841668 0.003365601 0.001443909
## Student006 0.0003866603 0.003168853 0.04801344 0.248329964 0.428747502
```

##	Membership	6 Estimate	Rank-Up Odds	Rank-Down Odds
## Student001	0.0006752546	2	0.7729769	0.7565891
## Student002	0.0067089816	3	0.5900527	0.3322503
## Student003	0.0538621490	3	0.7518042	0.3665372
## Student004	0.3067034562	5	0.8462973	0.5987019
## Student005	0.0010427491	1	0.7118604	NA
## Student006	0.2713535842	5	0.6328983	0.5791986

The results include: - Membership probabilities for each rank - Estimated rank (the rank with highest probability) - Rank-Up and Rank-Down Odds (indicating likelihood of transitioning to adjacent ranks)

Visualizing LRA Results

```
# Plot Item Reference Profiles (IRP) for the first 4 items
plot(lra_result, type = "IRP", items = 1:4, nc = 2, nr = 2)
```

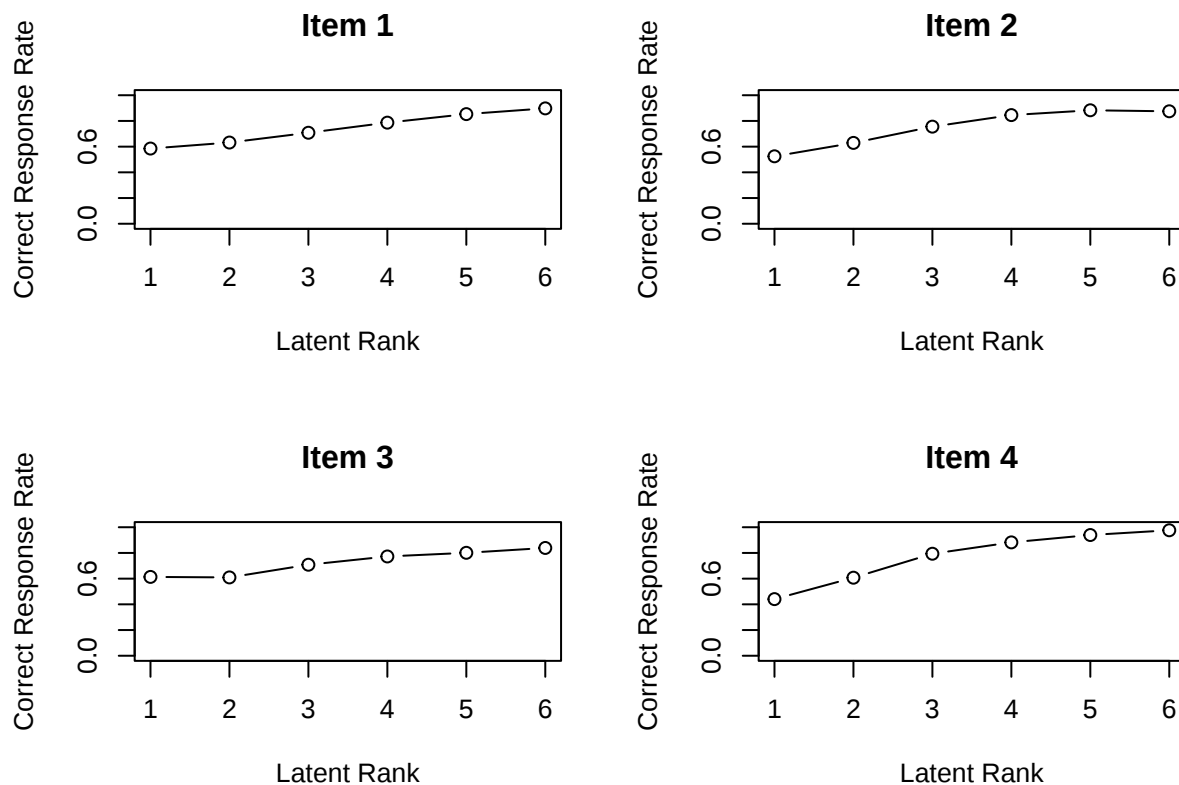


Figure 1: Item Reference Profile (IRP) plots

```
# Plot Rank Membership Profiles (RMP) for the first 4 examinees
plot(lra_result, type = "RMP", students = 1:4, nc = 2, nr = 2)
```

The IRP plots show how the probability of correct response increases with rank for each item. Steeper slopes indicate items with better discrimination between ranks.

Biclustering

Biclustering simultaneously clusters examinees into classes and items into fields, creating a two-mode clustering solution. This allows for a more detailed analysis of response patterns and item groups.

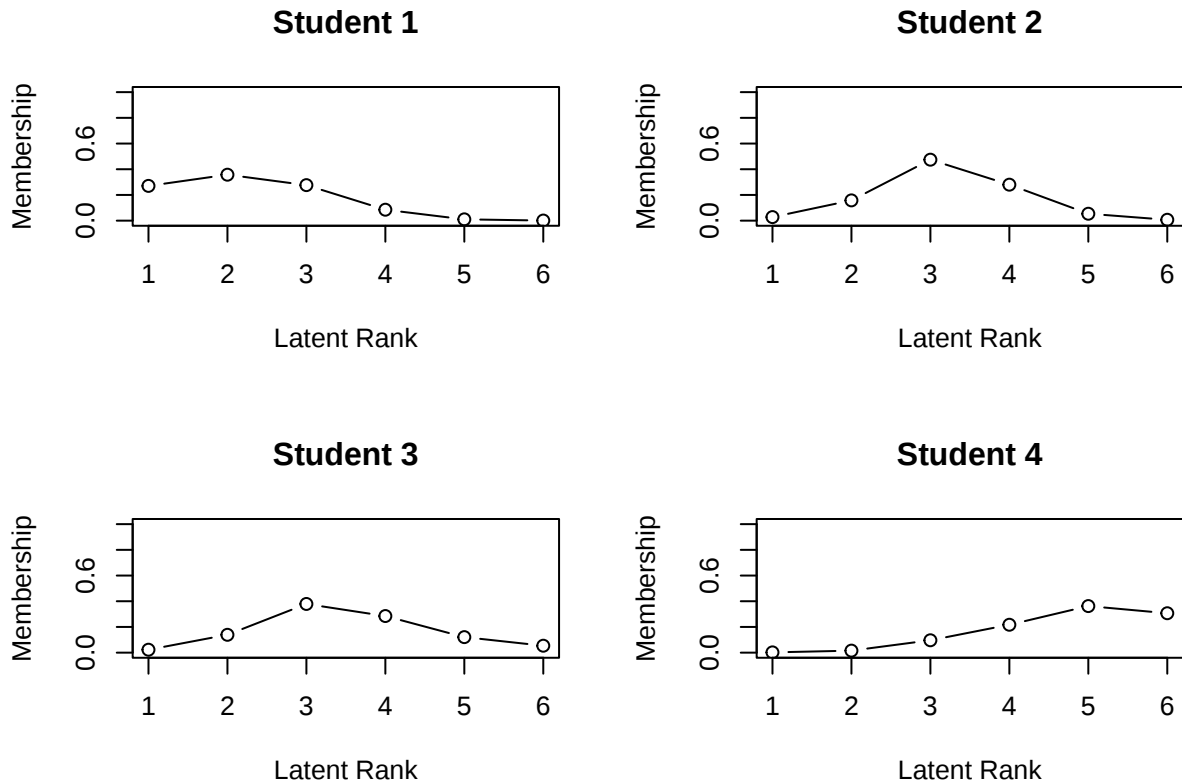


Figure 2: Rank Membership Profile (RMP) plots

```
# Load sample data
data(J35S515) # 35 items, 515 examinees

# Run Biclustering with 5 fields and 6 classes
bi_result <- Biclustering(J35S515, nfld = 5, ncls = 6, method = "B")

# Examine the Field Reference Profile (FRP)
# This matrix shows the probability of correct responses for each field at each class
bi_result$FRP
```

```
##           Class1      Class2      Class3      Class4      Class5      Class6
## Field1 0.62358989 0.86362311 0.87179729 0.89818946 0.9515799 1.0000000
## Field2 0.06270755 0.33323234 0.42554990 0.91879529 0.9904507 1.0000000
## Field3 0.20081642 0.54305218 0.22811379 0.47498694 0.7061406 1.0000000
## Field4 0.04954076 0.24549999 0.07824042 0.23310794 0.6482247 0.9828854
## Field5 0.02254831 0.05446897 0.02836654 0.04301178 0.1603798 0.9834310
```

Biclustering Model Fit

```
# Display test fit indices
bi_result$TestFitIndices

##      model_log_like bench_log_like null_log_like model_Chi_sq null_Chi_sq model_df
## 1      -6884.582      -5891.314      -9862.114      1986.535      7941.601      1160
##      null_df      NFI      RFI      IFI      TLI      CFI      RMSEA      AIC
## 1      1155 0.7498571 0.7509353 0.878121 0.8787357 0.8782108 0.03723232 -333.4651
```

```
##          CAIC          BIC
## 1 -5258.949 -5256.699
```

Field Membership

Item membership in fields shows how items are clustered:

```
# Display field membership for the first few items
head(bi_result$FieldMembership, 10)
```

##		Field1	Field2	Field3	Field4	Field5
##	Item01	1.000000e+00	1.502396e-109	8.724304e-109	7.567244e-250	0.000000e+00
##	Item02	8.593676e-129	4.944244e-43	1.000000e+00	1.004355e-09	6.230609e-111
##	Item03	5.500553e-98	1.268371e-55	1.000000e+00	2.502082e-32	6.205526e-149
##	Item04	3.811147e-184	3.381615e-90	2.532991e-11	1.000000e+00	9.732255e-66
##	Item05	1.287928e-231	3.581828e-133	1.241049e-35	1.000000e+00	7.667684e-45
##	Item06	4.066232e-256	1.339635e-152	5.908853e-43	1.000000e+00	4.082151e-31
##	Item07	5.517345e-36	1.473811e-46	1.000000e+00	2.283322e-67	9.589674e-213
##	Item08	2.250590e-149	1.329222e-120	1.000000e+00	3.015396e-15	1.882520e-79
##	Item09	1.061676e-126	1.035307e-75	1.000000e+00	4.300112e-18	1.398293e-103
##	Item10	3.533670e-146	4.534493e-95	1.000000e+00	4.152073e-11	4.233951e-84

Each row represents an item, with probabilities of belonging to each field. The highest probability indicates the most likely field for that item.

Class Membership

Examinee membership in classes:

```
# Display class membership for the first few examinees
head(bi_result$Students)
```

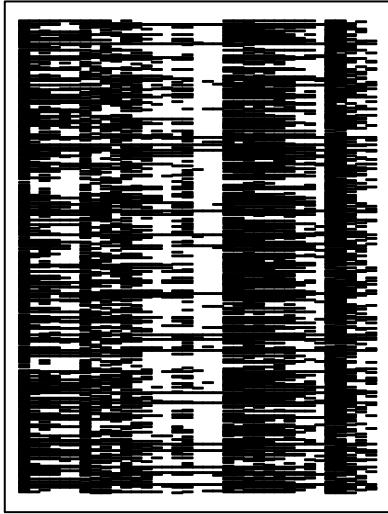
##		Membership 1	Membership 2	Membership 3	Membership 4	Membership 5
##	Student001	9.360799e-01	0.0487483523	1.517174e-02	2.001885e-09	1.341106e-20
##	Student002	3.995253e-13	0.0002472064	3.531118e-05	7.867206e-01	2.129969e-01
##	Student003	9.355415e-01	0.0489323551	1.552613e-02	1.291339e-09	2.627945e-20
##	Student004	2.450861e-06	0.1075442854	1.532487e-01	7.391882e-01	1.629733e-05
##	Student005	9.430185e-01	0.0158021680	4.117937e-02	9.642652e-10	3.870670e-21
##	Student006	2.075971e-02	0.0057891211	9.734434e-01	7.723502e-06	3.299905e-15
##		Membership 6	Estimate			
##	Student001	1.271407e-221		1		
##	Student002	8.756839e-63		4		
##	Student003	7.681856e-218		1		
##	Student004	5.768675e-126		4		
##	Student005	4.477727e-222		1		
##	Student006	3.076774e-206		3		

Visualizing Biclustering Results

```
# Array plot: visualize the raw data and the clustered result
plot(bi_result, type = "Array")
```

The Array plot shows the original data pattern on the left and the rearranged (clustered) pattern on the right. Black cells represent correct responses (1), and white cells represent incorrect responses (0). This visualization helps identify structure in the response patterns.

Original Data



Clusterd Plot

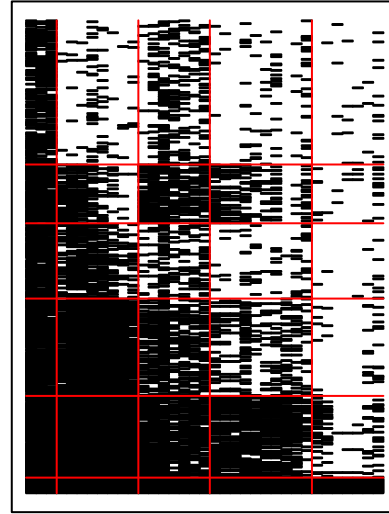


Figure 3: Array plot

```
# Plot of Field Reference Profile
plot(bi_result, type = "FRP", nc = 2, nr = 3)
```

Rankclustering

Rankclustering is an extension of Biclustering that incorporates ordered ranks for examinees instead of unordered classes, making it particularly useful for educational assessment.

```
# Run Rankclustering with 5 fields and 6 ranks
rank_result <- Biclustering(J35S515, nfld = 5, ncls = 6, method = "R")

# Examine the Field-Rank Profile (FRP)
# Shows the probability of correct responses for each field at each rank
rank_result$FRP
```

```
##           Rank1      Rank2      Rank3      Rank4      Rank5      Rank6
## Field1 0.64951141 0.79197342 0.88093148 0.9140316 0.9381373 0.9701273
## Field2 0.09362918 0.27752126 0.61746787 0.9036746 0.9835653 0.9979939
## Field3 0.22472428 0.32079306 0.44027355 0.6163989 0.7507110 0.8988009
## Field4 0.10394873 0.18598451 0.28100007 0.3614656 0.6030432 0.8411943
## Field5 0.03293073 0.05469207 0.09344104 0.1298351 0.2618470 0.5980969
```

Rankclustering Model Fit

```
# Display test fit indices
rank_result$TestFitIndices

##  model_log_like bench_log_like null_log_like model_Chi_sq null_Chi_sq model_df
## 1      -7273.063      -5891.314      -9862.114      2763.498      7941.601 1166.164
##  null_df      NFI      RFI      IFI      TLI      CFI      RMSEA      AIC
## 1      1155 0.6520226 0.6553538 0.7642464 0.7668873 0.7646342 0.05162221 431.1703
##           CAIC           BIC
```

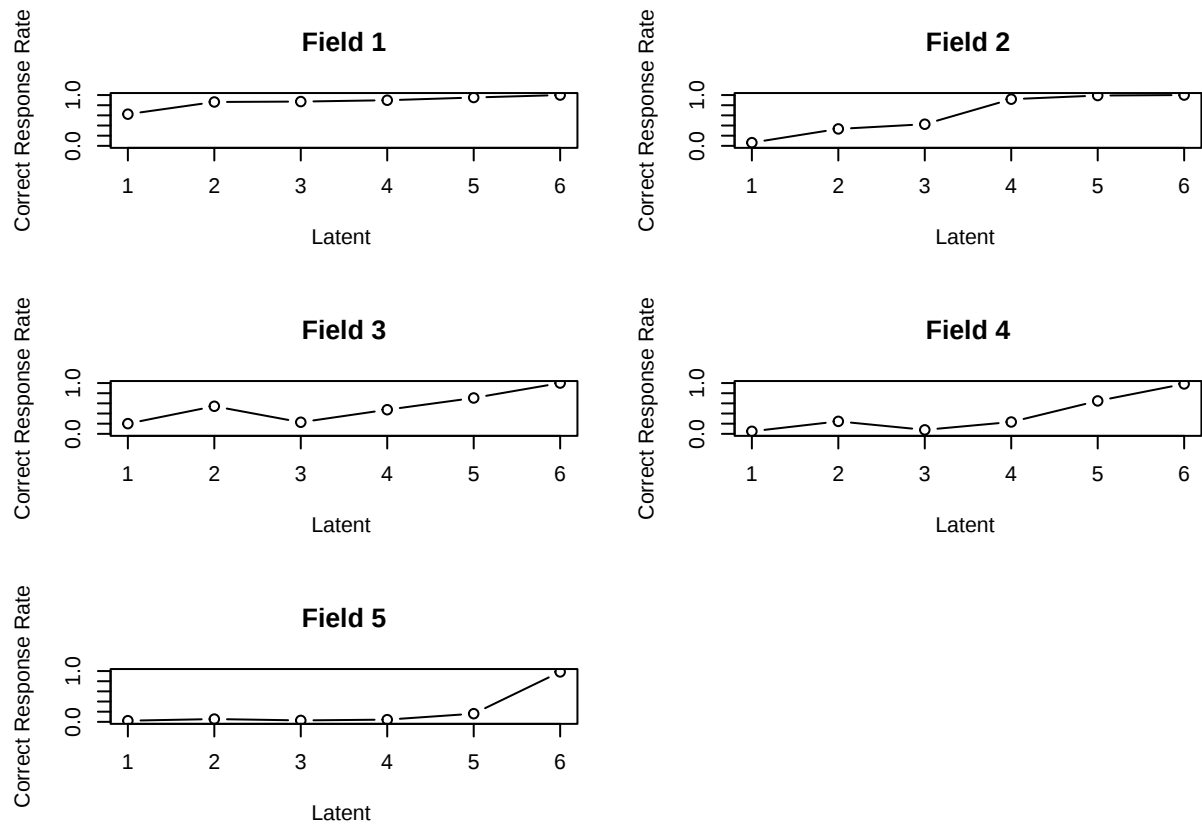


Figure 4: Field Reference Profile(FRP) plots

```
## 1 -4520.485 -4518.223
```

Field Membership in Rankclustering

```
# Display field membership for the first few items  
head(rank_result$FieldMembership, 10)
```

```
##           Field1      Field2      Field3      Field4      Field5  
## Item01  1.000000e+00  2.288832e-86  5.706565e-83  2.091043e-165  0.000000e+00  
## Item02  7.749584e-123  7.678734e-34  4.502569e-03  9.954974e-01  5.965152e-51  
## Item03  1.124141e-90  6.575315e-41  1.000000e+00  4.767645e-11  3.471920e-80  
## Item04  2.959054e-177  2.415474e-81  3.602966e-19  1.000000e+00  1.255540e-23  
## Item05  4.544645e-208  5.303588e-111  1.309392e-28  9.999999e-01  9.276257e-08  
## Item06  2.888383e-232  2.320900e-132  3.903128e-38  2.703468e-04  9.997297e-01  
## Item07  6.756857e-40  2.189049e-40  1.000000e+00  5.920375e-33  2.217949e-135  
## Item08  1.392498e-142  1.206256e-108  5.383242e-09  1.000000e+00  2.034326e-36  
## Item09  1.936847e-121  4.171067e-67  2.413280e-02  9.758672e-01  4.765915e-49  
## Item10  7.597617e-142  1.803765e-85  6.852640e-09  1.000000e+00  4.816190e-38
```

Rank Membership for Examinees

```
# Display rank membership for the first few examinees  
head(rank_result$Students)
```

```
##           Membership 1 Membership 2 Membership 3 Membership 4 Membership 5  
## Student001 7.306275e-01 2.676161e-01 0.001756384 2.530204e-08 8.959856e-16  
## Student002 1.013677e-11 1.822511e-06 0.007688111 3.203504e-01 6.716934e-01  
## Student003 6.132982e-01 3.822317e-01 0.004470062 9.297209e-08 7.848836e-15  
## Student004 4.105535e-05 2.965646e-02 0.756516510 2.134293e-01 3.566324e-04  
## Student005 7.582381e-01 2.400191e-01 0.001742819 3.503843e-08 4.617900e-16  
## Student006 2.567149e-01 6.901288e-01 0.053138690 1.764034e-05 5.055875e-12  
##           Membership 6 Estimate Rank-Up Odds Rank-Down Odds  
## Student001 1.370827e-28          1 0.3662826041          NA  
## Student002 2.662291e-04          5 0.0003963551          0.47692958  
## Student003 5.044221e-27          1 0.6232395801          NA  
## Student004 2.677414e-11          3 0.2821211892          0.03920134  
## Student005 3.675098e-29          1 0.3165484318          NA  
## Student006 9.433808e-24          2 0.0769982206          0.37198109
```

The results include Rank-Up and Rank-Down Odds, which are unique to Rankclustering and indicate the likelihood of moving to a higher or lower rank.

Visualizing Rankclustering Results

```
# Array plot  
plot(rank_result, type = "Array")  
  
# Field Reference Profile  
plot(rank_result, type = "FRP", nr = 2, nc = 3)
```

The key difference in visualization between Biclustering and Rankclustering is that the latter maintains an ordinal structure among examinee clusters, which is evident in the Array plot's pattern.

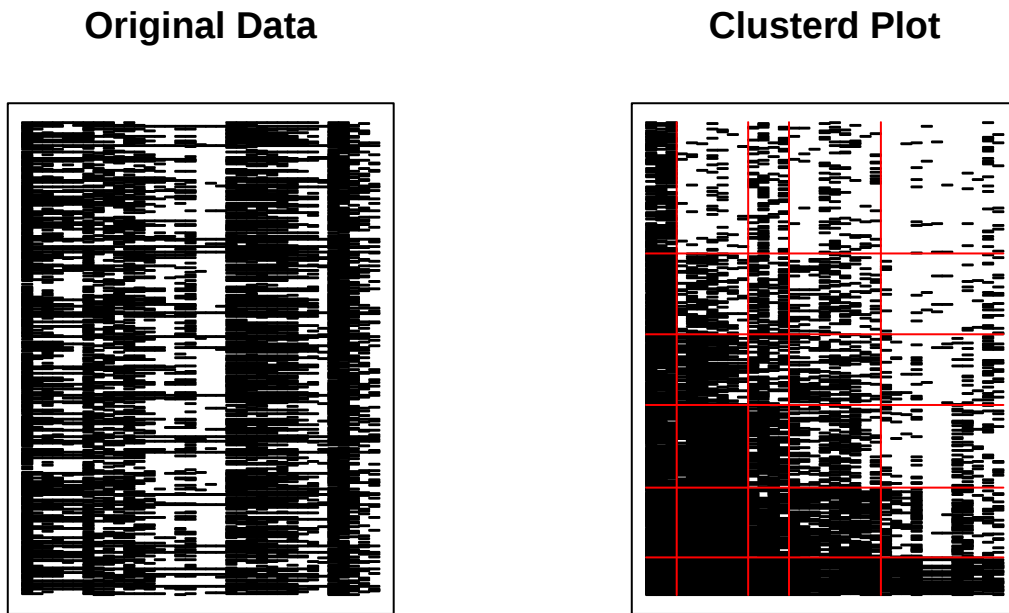


Figure 5: Array plot and Field Reference Profile(FRP) plots

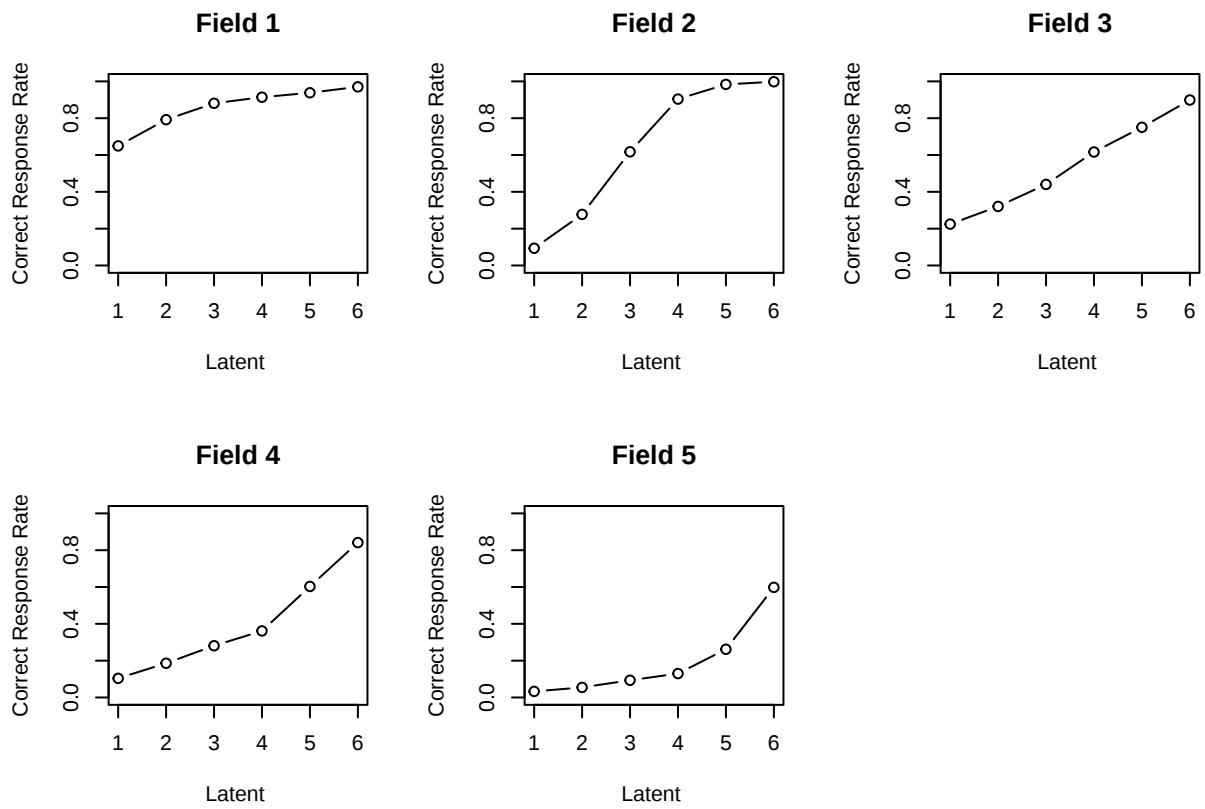


Figure 6: Array plot and Field Reference Profile(FRP) plots

Selecting the Optimal Number of Ranks and Fields

Finding the optimal number of ranks and fields is crucial for meaningful interpretation. While model fit indices provide guidance, domain knowledge about the test content should also inform this decision.

```
fit_indices <- data.frame()
for (r in 3:6) {
  result <- LRA(J15S500, nrank = r)
  fit_indices <- rbind(
    fit_indices,
    data.frame(
      Rank = r,
      AIC = result$TestFitIndices$AIC,
      BIC = result$TestFitIndices$BIC
    )
  )
}
print(fit_indices)
```

##	Rank	AIC	BIC
## 1	3	320.1021	-388.2848
## 2	4	311.4377	-349.9710
## 3	5	319.6186	-301.3337
## 4	6	318.9726	-264.7123

In educational contexts, the number of ranks should reflect meaningful developmental stages rather than being determined solely by statistical criteria.

Bayesian network model

The Bayesian network model provided by the **exametrika** package represents the conditional probabilities between items in a network format based on the pass rates of the items. This approach allows for modeling the hierarchical relationships and dependencies between test items, providing insights into knowledge structures and potential learning pathways.

Theoretical Background

Bayesian networks in educational assessment model the probabilistic relationships between test items, where edges in the network represent prerequisite relationships or cognitive dependencies. The model assumes that mastery of certain concepts (represented by correctly answering specific items) influences the probability of mastering dependent concepts.

Implementation in exametrika

By providing a Directed Acyclic Graph (DAG) between items externally, the package calculates the conditional probabilities based on the specified graph. The **igraph** package is used for the analysis and representation of the network. There are three ways to specify the graph structure:

- Pass a matrix-type DAG to the argument `adj_matrix`
- Pass a DAG described in a CSV file to the argument `adj_file`
- Pass a graph-type object `g` used in the **igraph** package to the argument `g`

Interpretation and Educational Applications

The BNM results provide several valuable insights:

- Conditional Correct Response Rates (CCRR): These indicate how the probability of answering an item correctly changes based on responses to prerequisite items

- **Parameter Learning:** The model estimates the strength of dependencies between items
- **Model Fit Indices:** These help evaluate whether the specified network structure adequately represents the relationships in the data

It is natural to assume directed paths from items with higher pass rates to those with lower pass rates. The model calculates conditional pass rates from parent nodes (higher pass rates) to child nodes (lower pass rates).

Practical Applications

The Bayesian network approach offers several practical benefits for educational assessment:

- **Learning Pathways:** By identifying prerequisite relationships, educators can design more effective learning sequences
- **Diagnostic Assessment:** The model can identify specific knowledge gaps based on response patterns
- **Adaptive Testing:** Results can inform which items to present next based on current performance
- **Curriculum Design:** Understanding item relationships helps in structuring educational content

The model fit can be evaluated by comparing it with both the complete independence model and the saturated model. Furthermore, it can provide a learning roadmap by identifying which items are easier to answer correctly and which concepts should be mastered first. For more advanced applications, the package also offers features to infer graph structures from data using genetic algorithms, as well as integration with biclustering to identify item groups and student classes simultaneously.

Example

There are three ways to specify the graph. You can either pass a matrix-type DAG to the argument `adj_matrix`, pass a DAG described in a CSV file to the argument `adj_file`, or pass a graph-type object `g` used in the `igraph` package to the argument `g`.

The methods to create the matrix-type `adj_matrix` and the graph object `g` are as follows:

```
library(igraph)
DAG <-
  matrix(
    c(
      "Item01", "Item02",
      "Item02", "Item03",
      "Item02", "Item04",
      "Item03", "Item05",
      "Item04", "Item05"
    ),
    ncol = 2, byrow = T
  )
## graph object
g <- igraph::graph_from_data_frame(DAG)
g

## IGRAPH 004652b DN-- 5 5 --
## + attr: name (v/c)
## + edges from 004652b (vertex names):
## [1] Item01->Item02 Item02->Item03 Item02->Item04 Item03->Item05 Item04->Item05
## Adjacency matrix
adj_mat <- as.matrix(igraph::get.adjacency(g))
print(adj_mat)

##           Item01 Item02 Item03 Item04 Item05
## Item01         0      1      0      0      0
```

```
## Item02      0      0      1      1      0
## Item03      0      0      0      0      1
## Item04      0      0      0      0      1
## Item05      0      0      0      0      0
```

A CSV file with the same information as the graph above in the following format. The first line contains column names (headers) and will not be read as data.

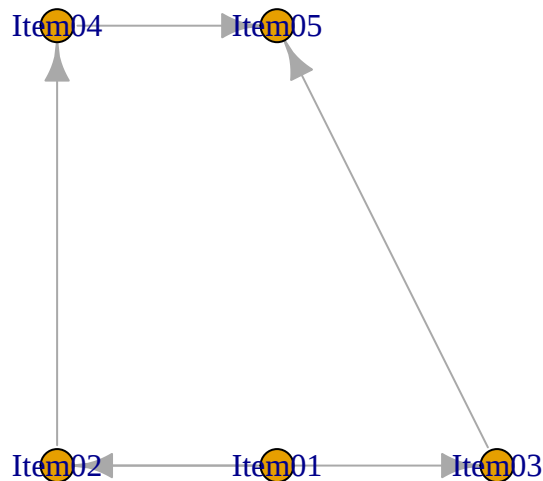
```
## From,To
## Item01,Item02
## Item02,Item03
## Item02,Item04
## Item03,Item05
## Item04,Item05
```

While only one specification is sufficient, if multiple specifications are provided, they will be prioritized in the order of file, matrix, and graph object.

An example of executing BNM by providing a graph structure (DAG) is as follows:

```
result.BNM <- BNM(J5S10, adj_matrix = adj_mat)
result.BNM
```

```
## Adjacency Matrix
##      Item01 Item02 Item03 Item04 Item05
## Item01      0      1      0      0      0
## Item02      0      0      1      1      0
## Item03      0      0      0      0      1
## Item04      0      0      0      0      1
## Item05      0      0      0      0      0
## [1] "Your graph is an acyclic graph."
## [1] "Your graph is connected DAG."
```



```
##
## Parameter Learning
##      PIRP 1 PIRP 2 PIRP 3 PIRP 4
## Item01  0.600
## Item02  0.250    0.5
## Item03  0.833    1.0
## Item04  0.167    0.5
## Item05  0.000   NaN  0.333  0.667
```

```
##
## Conditional Correct Response Rate
##      Child Item N of Parents      Parent Items      PIRP Conditional CRR
## 1      Item01              0      No Parents No Pattern      0.6000000
## 2      Item02              1      Item01      0      0.2500000
## 3      Item02              1      Item01      1      0.5000000
## 4      Item03              1      Item02      0      0.8333333
## 5      Item03              1      Item02      1      1.0000000
## 6      Item04              1      Item02      0      0.1666667
## 7      Item04              1      Item02      1      0.5000000
## 8      Item05              2 Item03, Item04      00      0.0000000
## 9      Item05              2 Item03, Item04      01      NaN(0/0)
## 10     Item05              2 Item03, Item04      10      0.3333333
## 11     Item05              2 Item03, Item04      11      0.6666667
##
## Model Fit Indices
##      value
## model_log_like -27.046
## bench_log_like -8.935
## null_log_like -28.882
## model_Chi_sq   36.222
## null_Chi_sq    39.894
## model_df       20.000
## null_df        25.000
## NFI            0.092
## RFI            0.000
## IFI            0.185
## TLI            0.000
## CFI            0.000
## RMSEA          0.300
## AIC            -3.778
## CAIC           -11.736
## BIC            -9.829
```

The package also offers features to infer graph structures from data using genetic algorithms, as well as a model that integrates with biclustering. For more details, please refer to the package manual or website(Kosugi,2023).

Conclusion

This appendix has provided an overview of the basic usage of the `exametrika` package and detailed the Bayesian Network Model (BNM), which could not be elaborated in the main text of the paper. The overall workflow can be summarized as:

1. Prepare your data (create a data frame or load from a CSV file)
2. Convert it to the `exametrika` class format using the `dataFormat` function
3. Select and run an appropriate analysis method (LRA, Biclustering, Rankclustering, BNM, etc.)
4. Check the results and visualize them as needed

Within this workflow, data conversion using the `dataFormat` function represents the most critical step. Properly specifying the data type, ID column, and missing values in this function ensures smooth analysis in subsequent steps.

The `exametrika` package, based on the theoretical framework proposed by Shojima (2022) in “Test Data Engineering,” provides an integrated approach for educational test data analysis. While the main paper focused primarily on LRA and Biclustering/Rankclustering, the package implements several other models. In

particular, the BNM introduced in this appendix is valuable for discovering prerequisite relationships between items and presenting learning pathways.

The visualization capabilities of the package also play a crucial role in providing educational feedback. Through IRP and FRP plots, array plots, and network diagrams, statistical results can be transformed into practical insights for educators and test-takers. Additionally, this appendix has addressed the package's application range, including handling ordinal data and accommodating polytomous data. These features enable appropriate analysis for a wider variety of test data.

The **exametrika** package goes beyond simple numerical evaluation, enabling a deeper understanding of examinee abilities and providing clear learning goals. By feeding back the results of rank analysis and clustering to educational settings, effective educational support becomes possible. For further details on the exametrika package, readers are encouraged to refer to the package website (Kosugi, 2023) and Shojima's (2022) book. These resources contain abundant information on other models and applications not covered in this appendix.