# IRT Tutorial: 2PL, Graded Response, and Bifactor Models

PSYC 6136 Final Project

Sarah Campbell

## Introduction

Item response theory (IRT) models can be understood as a logistic regression model with a latent variable, where instead of observed predictors, latent traits ($\theta$) are used to predict the probability of an item response. It is usually introduced from a psychometric perspective, presented as an alternative and often superior way of analyzing test data compared to classical test theory (CTT) approaches. This is no surprise, after all, IRT models have their grounding in testing and assessment.

However, from a modelling perspective, IRT still falls within categorical data analysis (CDA). Specifically, a close extension of logistic regression. For example, a standard logistic regression can be expressed as

$$P(y = 1 \mid X_i) = logit^{-1}(\beta_0 + \beta_1 X)$$

where $y$ is the binary outcome (i.e., 1/0), $X$ is the observed predictor, $\beta_0$ is the intercept, and $\beta_1$ is the slope.

On the other hand, the two-parameter logistic (2PL) model for dichotomous data can be expressed as

$$P(y = 1 \mid \theta) = logit^{-1}(a(\theta - b))$$

where here, $\theta$ is the latent predictor that governs the relationship between the item parameters: the discriminant (or intercept; $\alpha$) and the difficulty (or slope) parameter ($b$).

The main purpose of this tutorial is to demonstrate how to conduct IRT analyses using the R package `mirt` (v.1.46.1; Chalmers, 2012) for commonly used models: 2PL for dichotomous items, graded response (GRM) for ordered categorical items, and bifactor for multidimensional structures. This tutorial will walk the readers through the main functions that are used for fitting the IRT model, assessing model and item fit, extracting item parameters, obtaining factor scores, and plotting item functions.

## Conducting IRT analysis with `mirt`

As mentioned previously, IRT assumes that an unobserved latent trait (i.e., ability or $\theta$) governs the relationship between observed item responses (e.g., yes/no) and the item parameters (e.g., difficulty, discriminant, and guessing). In other words, IRT models calculate the likelihood of an individual, with a specific latent trait, answering a question or item correctly based on how difficult the item is among the other item parameters.

While multiple IRT models exist (e.g., 1PL/2PL/3PL for dichotomous item response with increasing levels of item parameter flexibility, Rasch model that's conceptually similar to 1PL models except fixing discrimination to 1.0, GRM for Likert-type data, multidimensional models, bi-factor models, multiple-group models, etc.), the following tutorial will focus on unidimensional 2PL, GRM and bifactor models for its relative conceptual simplicity and its wide-use in applied settings.

## Mathematical Expression for 2PL, GRM and Bifactor Models

### 2PL Model Expression

The 2PL model suitable for dichotomous data (e.g., yes/no) can be expressed as

$$P_{ij}(y = correct \mid \theta_i) = \frac{1}{1 + exp[-\alpha_j(\theta_i - b_j)]}$$

where $P_{ij}(y = correct|\theta_i)$ is the probability of a correct response $y$ to item $j$ for individual $i$, $\theta$ is the latent trait, $\alpha$ is the discriminant parameter for item $j$, and $b$ is the difficulty parameter. When $b_j = \theta$, the denominator reduces to $1 + exp(0) = 1 + 1 = 2$, and the right side of the equation becomes $\frac{1}{2} = 0.5$, representing the ability needed for a 50% probability of getting a correct response on that item. Intuitively, the higher the $\theta$, the higher the probability of getting a correct response, and vice versa.

Notice how only two item parameters are present: discriminant and difficulty, hence the name "2PL". Subsequently, 1PL (or one-parameter logistic) models can be understood as having one item parameter (i.e., the difficulty parameter), where equal discrimination among items is assumed, usually $\alpha_j = \alpha$ and $\alpha = 1.0$ specifically for Rasch models (Rasch, 1960; though they are often used interchangeably). 3PL (or three-parameter logistic) models has another parameter, the guessing parameter $c$.

### GRM Expression

The GRM (Samejima, 1969) is suitable for polytomous, Likert-type data (e.g., 5-point scale from "strongly disagree" to "strongly agree") and can be expressed similarly as

$$P^*_{ij}(y \geq k \mid \theta_i) = \frac{1}{1 + exp[-\alpha_j(\theta_i - b_{jk})]}$$

where $P^*_{ij}(y \geq k|\theta_i)$ is the cumulative probability of choosing a response in category $k$ or higher, and $b_{jk}$ is the threshold parameter (or difficulty parameter for each category threshold) for item $j$ category $k$ (de Ayala, 2022; Thissen et al., 2001). The threshold parameter represents the ability needed for a 50% probability of responding to category k or higher. Since there are multiple categories per item, the threshold per item is *k-1* (e.g., for Likert-type data with 5 ordered categories, there will be 4 thresholds). In sum, the GRM is essentially a 2PL extension to accommodate ordered response categories.

### Bifactor Model Expression

The bifactor structure can be modeled with either dichotomous or polytomous data. For dichotomous data, the model is essentially an extension of the 2PL model, and can be expressed as

$$P_j(y = correct \mid \theta_G, \theta_S) = \frac{1}{1 + exp[-(\alpha_{jG}(\theta_G - b_{jG}) + \alpha_{jS}(\theta_S - b_{jS}))]}$$

where now the probability of a correct response for item $j$ depends on both the general factor ($\theta_G$) and the specific factor ($\theta_S$).

Notice how each factor has its own difficulty parameter $b_{jG}$ and $b_{jS}$, which is impossible to solve for a single difficulty threshold (i.e., there are multiple combinations of $b_{jG}$ and $b_{jS}$ to get the same results). Therefore, in practice, it is usually expressed in a slope-intercept form with a single intercept ($d_j$), where $b = -d/\alpha$, resulting in the following:

$$P_j(y = correct \mid \theta_G, \theta_S) = \frac{1}{1 + exp[-(\alpha_{jG}\theta_G + \alpha_{jS}\theta_S + d_j)]}$$

The conversion from a slope-difficulty form to a slope-intercept form is usually done in practice, as the parameters are easier to estimate (this conversion can be easily done in `mirt` through the `traditional2mirt()` function or manually with $b = -d/\alpha$).

## Packages

The `mirt` package (v.1.46.1; Chalmers, 2012) in R (R Core Team, 2026) will be used to fit the IRT models in this tutorial.

```
library(mirt)
library(psych)
```

## Model Fitting: 2PL Dichotomous

There are multiple built-in datasets in `mirt`, some of which will be used today. For the 2PL model with dichotomous data, I decided to go with the `LSAT7` dataset. `LSTA7` consist of 5 items with dichotomous responses (i.e., 0/1) with 1000 individuals.

Since the `LSAT7` dataset is in frequency form, we need to first expand it to raw responses with each row representing an individual's response and the columns as the items. The `expand.table()` function does just that.

```
data(LSAT7)
dat <- expand.table(LSAT7)
head(dat)
```

```
##   Item.1 Item.2 Item.3 Item.4 Item.5
## 1      0      0      0      0      0
## 2      0      0      0      0      0
## 3      0      0      0      0      0
## 4      0      0      0      0      0
## 5      0      0      0      0      0
## 6      0      0      0      0      0
```

The `itemstats()` function provides an overview of the item distributions. Here we see that the mean total score is 3.7 out of a maximum score of 5, which implies that most people score high on this set of items. Under `$itemstats`, we can check for missing data and decide ways to account for missingness. IRT generally handles missing data well, since its focus is on the item level probability instead of total scores as CTT does. Furthermore, due to its local independence assumption (i.e., each item response contributes information to the ability estimate independently), missing responses will simply contribute no information to the overall estimate. So, say a person did not respond to a few items, the person's estimated ability would just have lower precision (i.e., high standard error (SE)), but can still produce an estimated latent ability.

```
itemstats(dat)
```

```
## $overall
##     N mean_total.score sd_total.score ave.r  sd.r alpha SEM.alpha
##  1000             3.707          1.199 0.143 0.052 0.453     0.886
##
## $itemstats
##           N K  mean    sd total.r total.r_if_rm alpha_if_rm
## Item.1 1000 2 0.828 0.378   0.530         0.246       0.396
## Item.2 1000 2 0.658 0.475   0.600         0.247       0.394
## Item.3 1000 2 0.772 0.420   0.611         0.313       0.345
## Item.4 1000 2 0.606 0.489   0.592         0.223       0.415
## Item.5 1000 2 0.843 0.364   0.461         0.175       0.438
##
## $proportions
##            0     1
## Item.1 0.172 0.828
## Item.2 0.342 0.658
```

```
## Item.3 0.228 0.772
## Item.4 0.394 0.606
## Item.5 0.157 0.843
```

```
# Fit 2PL model
mod_2PL <- mirt(dat, 1, itemtype = '2PL')
```

### Model and Item Fit

As a preliminary check, we can first look at the model and item fit before examining the item parameters more closely. The M2 statistic provides a measure for how well the model fits the data. The non-significant p-value based on 0.01 indicates no significant difference between the model and the data (i.e., good fit). There are also loosely defined model-fit cut-offs for a good fit of RMSEA <.05-.10, SRMSR <.08-.10, TLI >.90-.95, CFI >.90-.95. However, there has not been agreed upon cut-offs, as acceptable fit indices vary greatly between sample sizes, model complexity, data distribution, item response types, factor loadings and more. Simulation-based cut-offs, such as the dynamic fit indicies (DFI) by McNeish and Wolf (2023), is generally considered an ideal choice for gauging model fit indices. The authors also developed a Shiny Application that allows users to easily generate custom cut-offs without deep knowledge of simulation methods (Wolf & McNeish, 2020).

It is also important to note that these cut-offs are only one part of understanding how well the model fits the data. In practice, we often examine multiple pieces of evidence and consider the theoretical understanding of the measure to form a judgement on whether such model is appropriate.

```
M2(mod_2PL)
```

```
##           M2 df     p RMSEA RMSEA_5 RMSEA_95 SRMSR   TLI   CFI
## stats 11.938  5 0.036 0.037   0.009    0.065 0.032 0.937 0.968
```

In IRT, items are the unit of measure and is often of more interest. We can asses how well each item fits the model by the `itemfit()` function, which provides a few different fit indices that assess fit. The `S_X2` is the default chi-squared test, it compares the observed and the expected item response distributions. A significant p-value suggests misfit. Though one would usually company the `RMSEA.S_X2` value of >.06 to assess the misfit. Here we see that item 2 is significant and the `RMSEA.S_X2` value is >.60 (RMSEA = .079), suggesting that this item may be driving the slight misfit of the overall model above, and perhaps the item is measuring the effects beyond the one factor.

```
itemfit(mod_2PL)
```

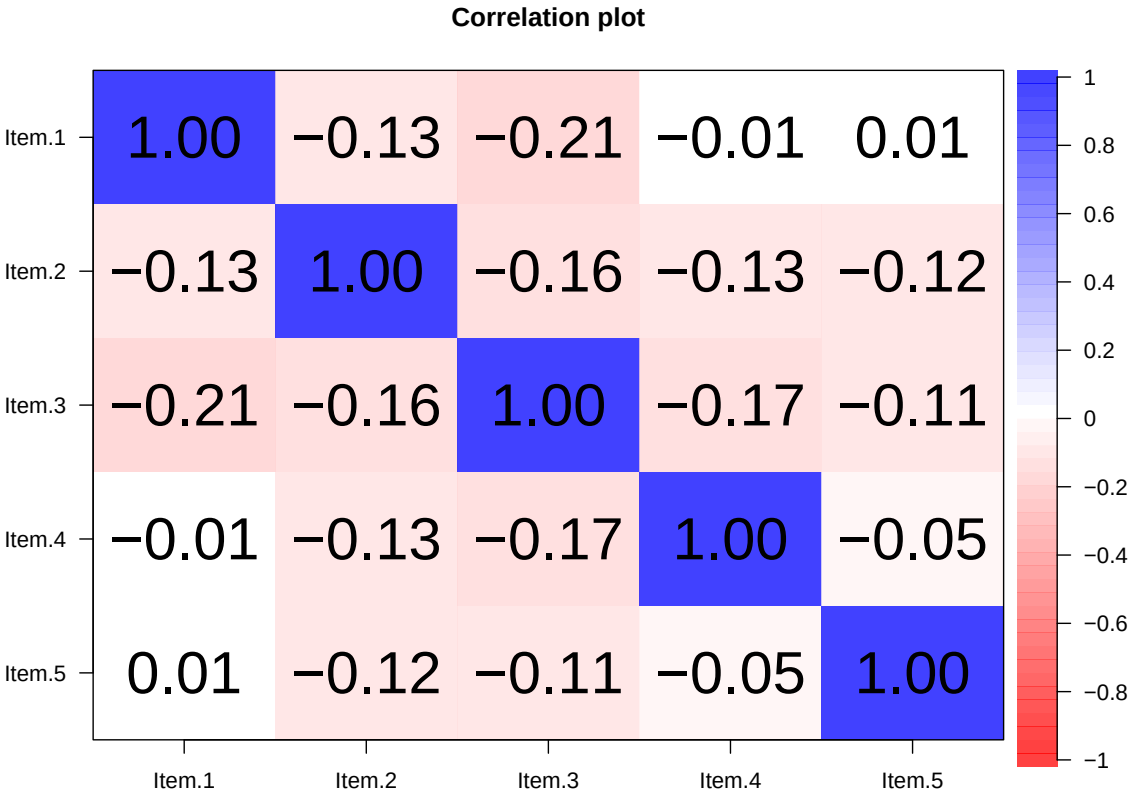
```
##     item   S_X2 df.S_X2 RMSEA.S_X2 p.S_X2
## 1 Item.1  4.749       2      0.037  0.093
## 2 Item.2 14.453       2      0.079  0.001
## 3 Item.3  1.270       2      0.000  0.530
## 4 Item.4  5.238       2      0.040  0.073
## 5 Item.5  0.941       2      0.000  0.625
```

Here we often also look at the residuals to assess the local independence assumption. Large residuals correlations, usually >.30 suggest a violation of the local independence assumption. IRT has a strict local independence assumption, where each item's response is independent of another response, conditioned on the latent trait. So high residual correlations among items suggest that the pair of items are still related (e.g., perhaps answering one item would influence the response of another item, beyond the latent trait) and should be additionally modeled. This is usually the case in bifactor or testlet type structures, where the items are conceptually unidimensional, but often have item bundles that measure a sub-trait that needs to be accounted for.

Below we see that the residuals look reasonable, with a slight negative correlation. You can also plot it for a more visually appealing examination. The darker the cells, the more strongly the items are dependent.

```
psych::cor.plot(residuals(mod_2PL, type = "Q3"))
```

```
## Q3 summary statistics:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.214  -0.156  -0.129  -0.111  -0.070   0.009
##
##        Item.1 Item.2 Item.3 Item.4 Item.5
## Item.1  1.000 -0.135 -0.214 -0.012  0.009
## Item.2 -0.135  1.000 -0.163 -0.134 -0.125
## Item.3 -0.214 -0.163  1.000 -0.170 -0.115
## Item.4 -0.012 -0.134 -0.170  1.000 -0.055
## Item.5  0.009 -0.125 -0.115 -0.055  1.000
```




### Extract and Interpret Item Parameters

We can understand IRT parameters through a factor analysis lens. F1 are the factor loadings and h2 are the commonalities, which is the proportion of variance in the item explained by the latent factor.

```
summary(mod_2PL)    # viewed through a factor analysis/loading lens
```

```
##           F1    h2
## Item.1 0.502 0.252
## Item.2 0.536 0.287
## Item.3 0.708 0.501
## Item.4 0.410 0.168
## Item.5 0.397 0.157
##
## SS loadings:  1.366
```

```
## Proportion Var:  0.273
##
## Factor correlations:
##
##    F1
## F1  1
```

However, in IRT, we usually look at the item parameters directly (i.e., discriminant and difficulty). Below we see that the estimated discriminant parameters (a) is between 0.7 and 1.7. Recall that the discriminant parameter measures how well an item differentiates individuals with similar abilities, the higher the discrimination, the better the item is at differentiating people. Though an extremely high discrimination (usually above 4) is also problematic. Here, item 3 has the highest level of discrimination.

Since 2PL models estimate both the difficulty and the discrimination parameter, we see varying estimates for *a*. On the other hand, the coefficients for 1PL or Rasch models would present either equal $a$ parameter estimates or equal to 1 parameters among items.

The difficulty parameters (b) reflects value of the ability needed for a 50% chance of getting a correct response on that item. The higher the difficulty parameter value, the higher the person's ability needs to be to obtain a 50% chance of getting the answer correct. For example, for item 5, b = -2.520, which suggest that an ability that is 2.52 standard deviation (SD) below the sample's latent variable mean (which is 0, as IRT identifies $\theta \sim N(0, 1)$) has a 50% chance of getting the item correct. This is a pretty easy item. So are all the other items in this dataset. We can visually inspect this relationship through the test information plot that would allow us to see where on the latent trait the test is the most precise (see plots under the "Plotting" section).

A coding note here is to set `IRTpars=T` if you would like the slope-intercept form that was used to estimate the model parameters to be converted back to a slope-difficulty form. The slope-difficulty form is much more interpretable, as the $b$ parameter directly reflects the ability needed for a 50% chance of correct response.

```
coef(mod_2PL, simplify=TRUE, IRTpars=TRUE)    # IRT item coefiicients (a1, d1, d2...)
```

```
## $items
##            a      b g u
## Item.1 0.988 -1.879 0 1
## Item.2 1.081 -0.748 0 1
## Item.3 1.706 -1.058 0 1
## Item.4 0.765 -0.635 0 1
## Item.5 0.736 -2.520 0 1
##
## $means
## F1
##  0
##
## $cov
##    F1
## F1  1
```

### Obtaining Factor Scores

The `fscores()` produces the latent trait estimates for each individual. There are multiple methods to estimate the latent trait, the default is expected a posteriori (`EAP`), but Bayesian modal maximum a posteriori (`MAP`), weighted maximum likihood (`WME`), and sum score approximations (`EAPsum`), among others, are also available. The `full.scores.SE = T` argument gives you the estimated SE as well.

One advantage of IRT scoring is the ability to estimate the latent traits directly, whereas sum scores would not reflect is nuance. For example, individuals with the same sum scores can have different latent traits as

the items they got correct may differ in difficulty and discrimination. This is why using sum scores is often a crude estimate of one's ability.

```
theta_hat <- fscores(mod_2PL, method = "EAP", full.scores.SE = TRUE)
head(theta_hat)
```

```
##             F1     SE_F1
## [1,] -1.869854 0.6927338
## [2,] -1.869854 0.6927338
## [3,] -1.869854 0.6927338
## [4,] -1.869854 0.6927338
## [5,] -1.869854 0.6927338
## [6,] -1.869854 0.6927338
```

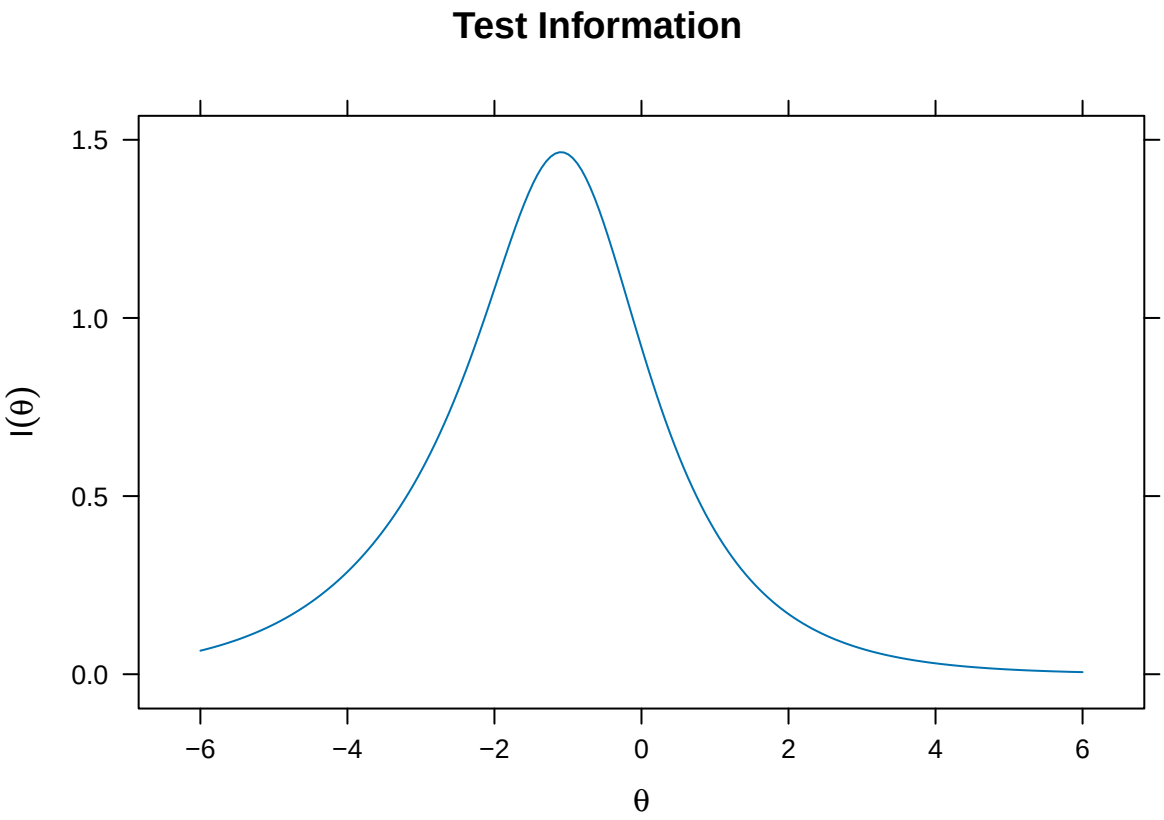
### Plotting (Test-level Plot vs Item-level Plot)

The `plot()` function in `mirt` provides a variety of plots for visualizing the relationship between items and the latent ability. I would start of with test-level plots, then present some item-level plots.

### Test-level Plots

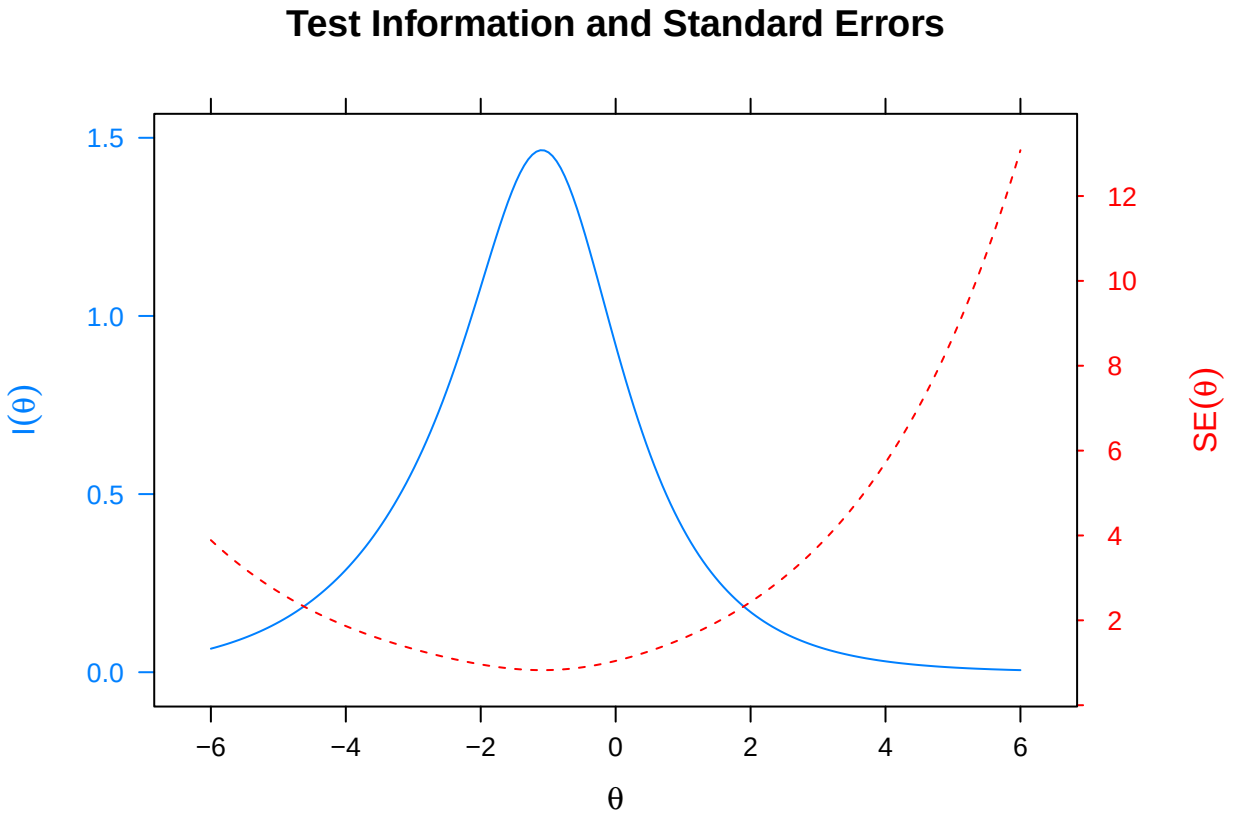
**Test Information Curves** The test information plot below is a very useful plot to see where on the latent trait the test is the most precise, as evaluated by test information. The higher the information, the more precise the test is at the corresponding latent trait and the lower the SE. The desired information for a test differs from measure to measure. For example, an ideal test information distribution for a measure designed for the general population would be one with high information in the middle of the theta range, perhaps $\theta \sim [-2, 2]$. Whereas clinical measures designed to assess high symptom levels may want the test information to peak at the higher end of the theta range.

```
plot(mod_2PL, type = 'info')  # Test Information Curve
```



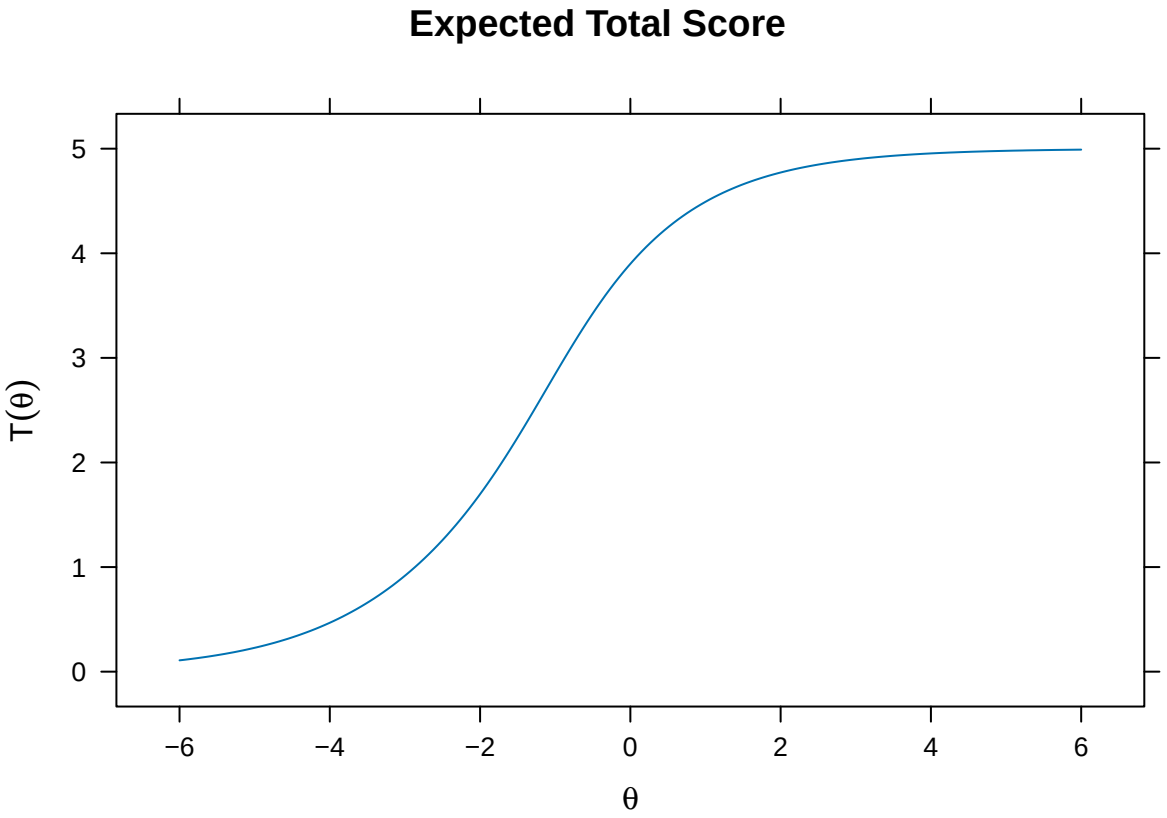

Here the plot combined the test information curve with the corresponding SE.

```
plot(mod_2PL, type = "infoSE")  # Test Information + SE
```




**Expected Total Score Plot** The expected total score plot looks at the relation between $\theta$ and the number of correct scores. The 5-item `LSAT7` measure is a fairy easy test. This is evident here with the middle of the "S" curve corresponding to a $\theta$ of around -1.5, and average students, as indicated by $\theta = 0$, obtain a high score of 4/5.

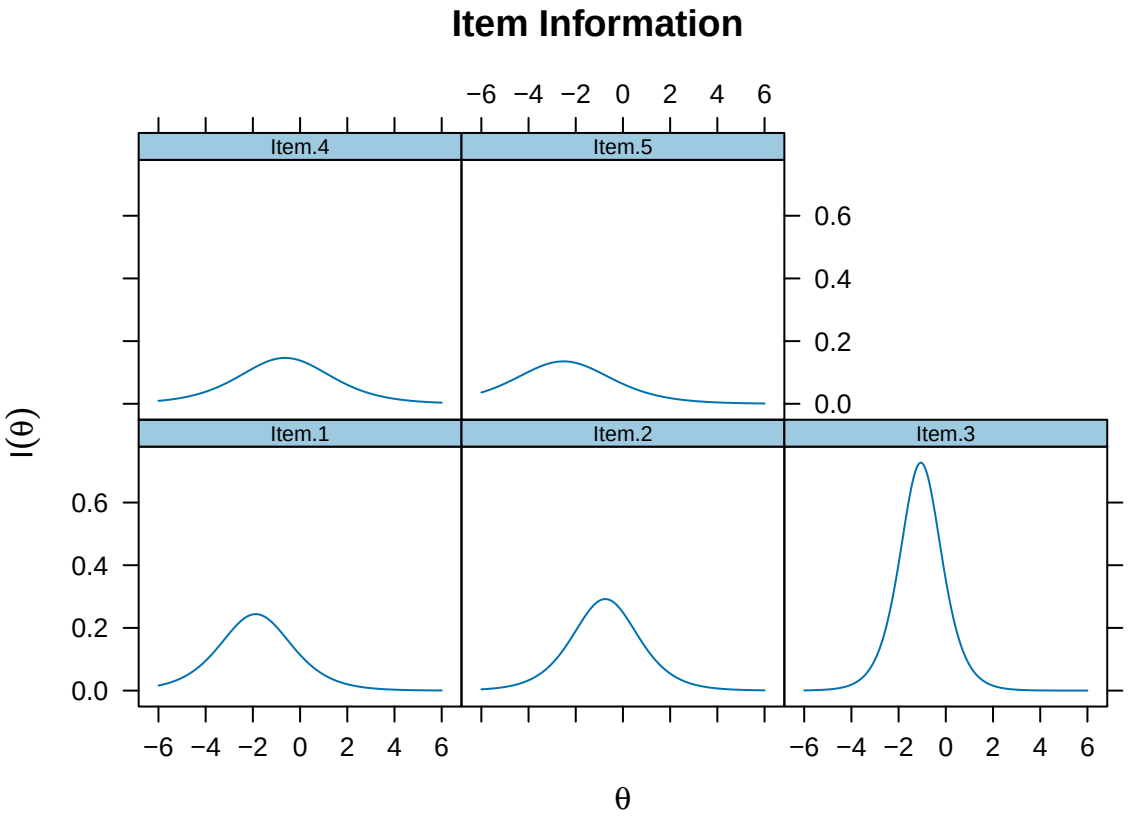
```
plot(mod_2PL) # Expected Total Score (S shape)
```




**Item-level Plots** We can look further into which specific items have the most or least information, and which items are relatively hard or easy compare to the rest by looking at item-level plots.

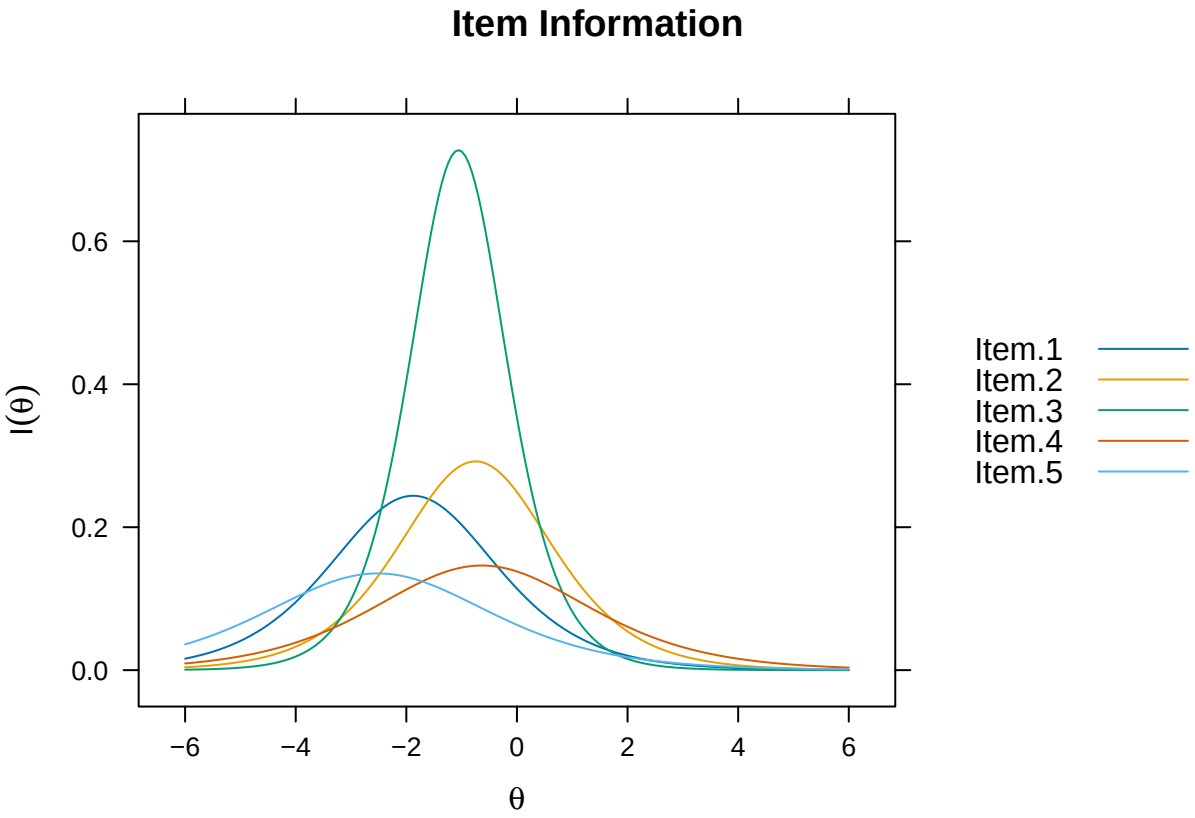
**Item Information Curves** Similar to the above test information curve for the entire measure, we can isolate the information each item contributes from the `infotrace` plot below. Here we see that item 3 is driving the high test information, followed by item 2 and item 1, whereas item 4 and 5 seem to contribute relatively little information.

```
plot(mod_2PL, type = "infotrace")
```



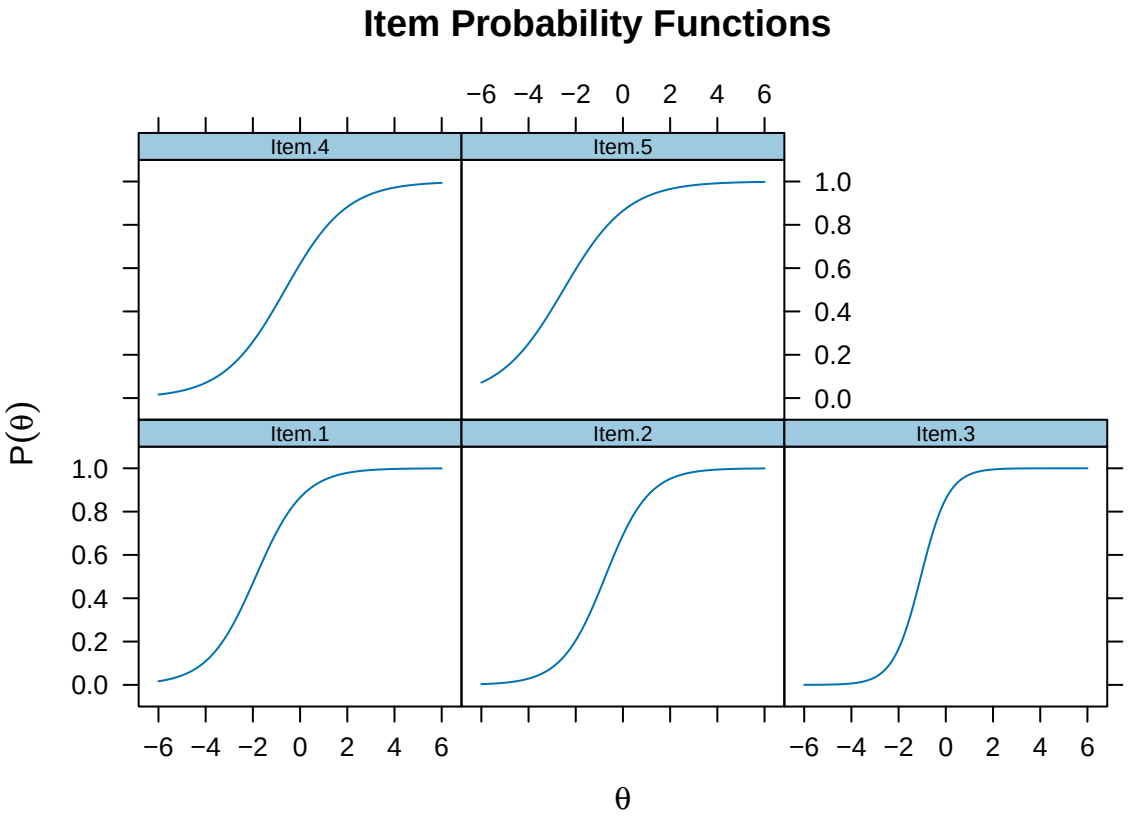

It can be useful to visualize the item information all in one plot, which can be done by passing the `facet_items = FALSE` argument.

```
plot(mod_2PL, type = "infotrace", facet_items = FALSE)
```



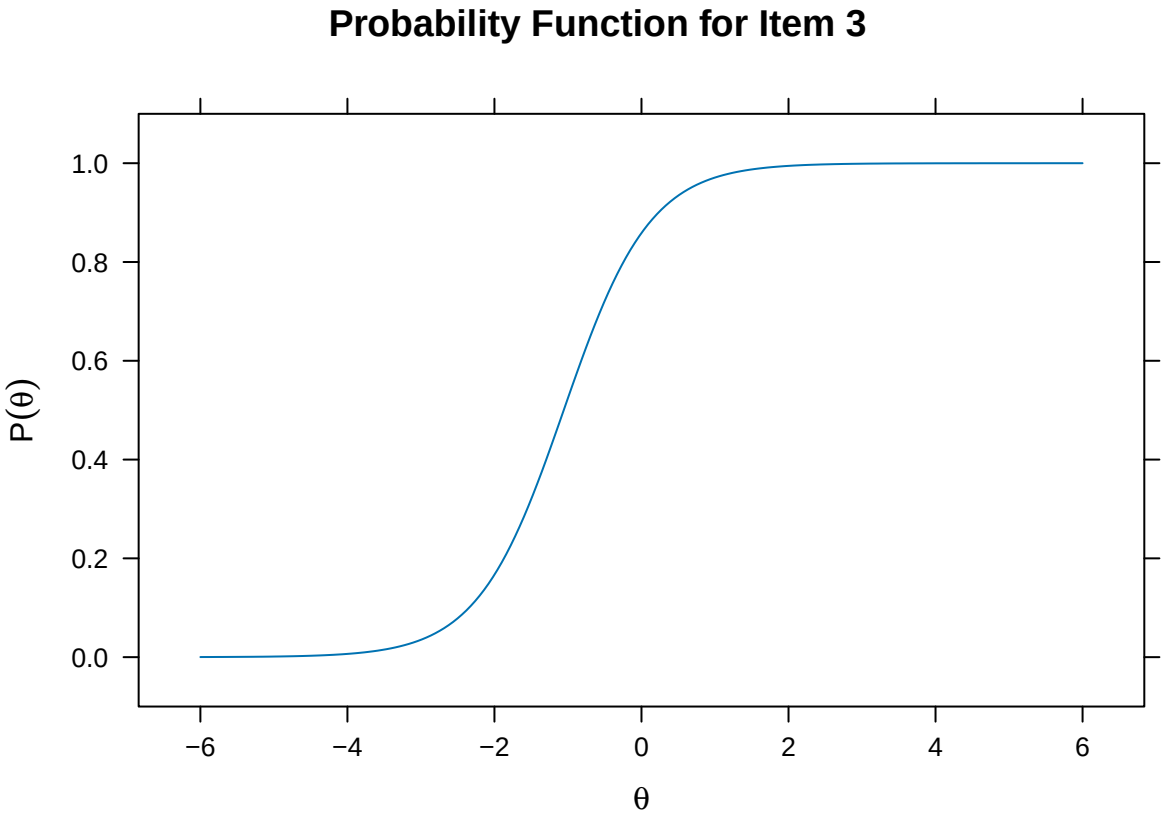

**Item Probability Functions** The item probability plot below shows the relation between the ability level and the corresponding probability of getting the item correct for each item. Here we see that item 3 has a very steep curve, indicating that it has high discrimination around theta of -1. This confirms what we saw earlier when examining the coefficients, where item 3 has the highest discriminant parameter (a = 1.706).

```
plot(mod_2PL, type = "trace")
```



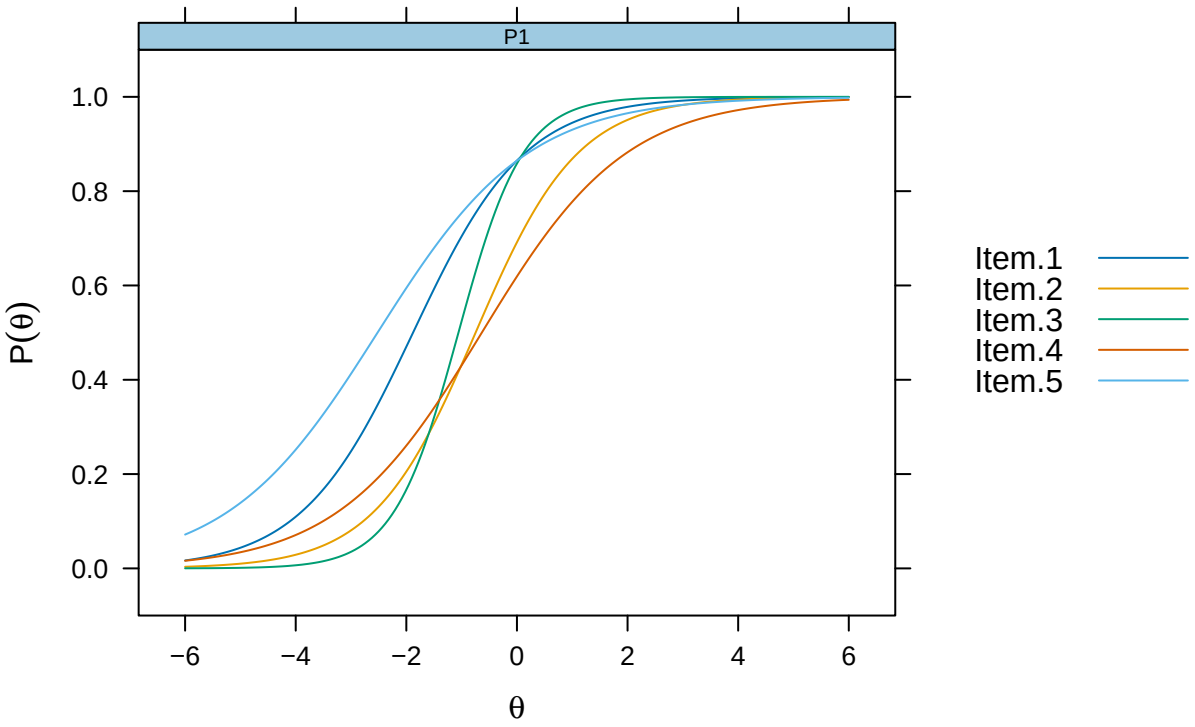

The `itemplot()` function shows the exact same item probaility, but for a single specified item.

```
itemplot(mod_2PL, 3)    # For item 3 only
```




It can also be useful to plot all the item probability functions in one plot, which can be done by passing the argument `facet_items = F`.

```
plot(mod_2PL, type = "trace", facet_items = FALSE)
```




### Model Fitting: GRM Polytomous

Polytomous scoring for ordered categories is common in psychological measures. Hence, here I also provide an example for fitting GRMs. Throughout the next two sections, you would find that most models in `mirt` are similarly coded, so fitting different models is not a steep learning curve.

For GRM, let us generate some data from known parameters. Researchers usually would not know the parameter estimates if they were to conduct IRT analysis on raw scores. The usual route would be from the 2PL example above, where one may only need to clean the data and score it. However, these IRT parameter estimates can be found through published IRT scale validation studies.

Here I decided to go for simulated data with 1000 individual responses with 10 items, each item has 5 response categories (hence k-1, 4 thresholds). The difficulty threshold parameters need to be ordered. In fact, un-ordered categories are usually flagged as problematic items, as the monotonically assumption would otherwise be violated. Instead of re-parameterizing the difficulty parameter from difficulty to intercepts, I decide to just input the intercepts ($d$).

```
set.seed(3959)

# 10 items, each item has 1 discriminant parm
a <- matrix(c(1.5, 1.2, 1.0, 0.8, 0.6, 0.8, 1.0, 1.5, 1.4, 1.0))
d <- matrix(c(
  1.5, 0.5, -0.5, -1.5,      # 10 rows, 4 columns.
  1.2, 0.2, -0.8, -1.0,      # Each column is an category threshold for the row item
  1.0, 0.0, -1.0, -2.0,
  1.3, 0.3, -0.7, -1.5,
  0.8, -0.2, -1.2, -2.2,
  1.6, 0.3, -0.7, -1.5,
  1.1, 0.5, -0.1, -1.5,
```

```
  1.5, 0.4, -0.7, -1.0,
  1.6, 0.8,  -0.1,  -1.8,
  0.6,  0.2,  -1.0,  -1.8
), nrow = 10, byrow = TRUE)

dat2 <- simdata(a = a, d = d, N = 1000, itemtype = "graded")
head(dat2)
```

```
##      Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## [1,]      4      4      2      0      4      4      1      4      2       2
## [2,]      0      0      3      0      0      0      0      0      2       4
## [3,]      0      4      0      1      0      4      4      0      0       0
## [4,]      4      1      4      3      0      2      3      3      4       0
## [5,]      3      4      4      4      2      4      3      4      0       2
## [6,]      0      4      0      1      1      1      0      1      2       2
```

We can also convert the intercepts (d) to a more interpretable difficulty thresholds (b). Here we see that item 1 and 2's difficulty seem to be centred around 0, with an average level of difficulty. Item 3, 4, and 5 have higher difficulty level for the highest category, and item 4 and 5 also seem to have a wider spread (see item 5 with a range of -1.3 to 3.6).

```
b <- -d / as.vector(a)
b
```

```
##            [,1]       [,2]       [,3]      [,4]
##  [1,] -1.000000 -0.3333333 0.33333333 1.0000000
##  [2,] -1.000000 -0.1666667 0.66666667 0.8333333
##  [3,] -1.000000  0.0000000 1.00000000 2.0000000
##  [4,] -1.625000 -0.3750000 0.87500000 1.8750000
##  [5,] -1.333333  0.3333333 2.00000000 3.6666667
##  [6,] -2.000000 -0.3750000 0.87500000 1.8750000
##  [7,] -1.100000 -0.5000000 0.10000000 1.5000000
##  [8,] -1.000000 -0.2666667 0.46666667 0.6666667
##  [9,] -1.142857 -0.5714286 0.07142857 1.2857143
## [10,] -0.600000 -0.2000000 1.00000000 1.8000000
```

The `$proportions` section under `itemstats()` shows the proportion of responses to each category. Low response categories can result in convergence problems and high SEs, and considerations should be made as to whether to collapse the item categories (e.g., 7-point scale combined to 5-point scale, in which category 1 and 2 are combined and category 6 and 7 are combined). There are no agreed upon cut-offs for what is considered "low", with some suggesting fewer than 10 response in each category and others suggest 2% to 4% (Liu et al., 2025). However, here, item categories sparseness is not a problem.

```
itemstats(dat2)
```

```
## $overall
##     N mean_total.score sd_total.score ave.r  sd.r alpha SEM.alpha
##  1000            18.93          8.439 0.236 0.077 0.759     4.144
##
## $itemstats
##            N K  mean    sd total.r total.r_if_rm alpha_if_rm
## Item_1  1000 5 1.984 1.551   0.658         0.533       0.723
## Item_2  1000 5 1.979 1.593   0.615         0.476       0.731
## Item_3  1000 5 1.688 1.490   0.546         0.403       0.741
## Item_4  1000 5 1.903 1.442   0.488         0.341       0.750
```

```
## Item_5  1000 5 1.540 1.344   0.383         0.235       0.762
## Item_6  1000 5 1.961 1.427   0.506         0.364       0.746
## Item_7  1000 5 2.056 1.570   0.559         0.410       0.741
## Item_8  1000 5 2.064 1.591   0.670         0.544       0.721
## Item_9  1000 5 2.112 1.471   0.626         0.501       0.728
## Item_10 1000 5 1.643 1.527   0.538         0.390       0.743
##
## $proportions
##             0     1     2     3     4
## Item_1  0.272 0.137 0.183 0.151 0.257
## Item_2  0.255 0.195 0.185 0.046 0.319
## Item_3  0.321 0.173 0.180 0.149 0.177
## Item_4  0.231 0.194 0.221 0.149 0.205
## Item_5  0.303 0.219 0.223 0.145 0.110
## Item_6  0.189 0.249 0.196 0.144 0.222
## Item_7  0.283 0.112 0.115 0.246 0.244
## Item_8  0.246 0.164 0.202 0.056 0.332
## Item_9  0.234 0.117 0.164 0.273 0.212
## Item_10 0.384 0.075 0.235 0.126 0.180
```

The GRM is fitted by passing through the `itemtype = 'graded'` argument. The number "1" implies that we are fitting the default unidimensional model.

```
# Fit GRM model
mod_GRM <- mirt(dat2, 1, itemtype = "graded")
```

### Model and Item Fit

Similar to the 2PL example above, model fit can be obtained via the `M2()` function. Here, model fit looks good, $\chi^2(5) = 11.536, p = 0.04$.

```
M2(mod_GRM)
```

```
##           M2 df     p RMSEA RMSEA_5 RMSEA_95 SRMSR   TLI   CFI
## stats 11.536  5 0.042 0.036   0.007    0.064 0.025 0.787 0.929
```

Item fit also looks decent.

```
itemfit(mod_GRM)
```

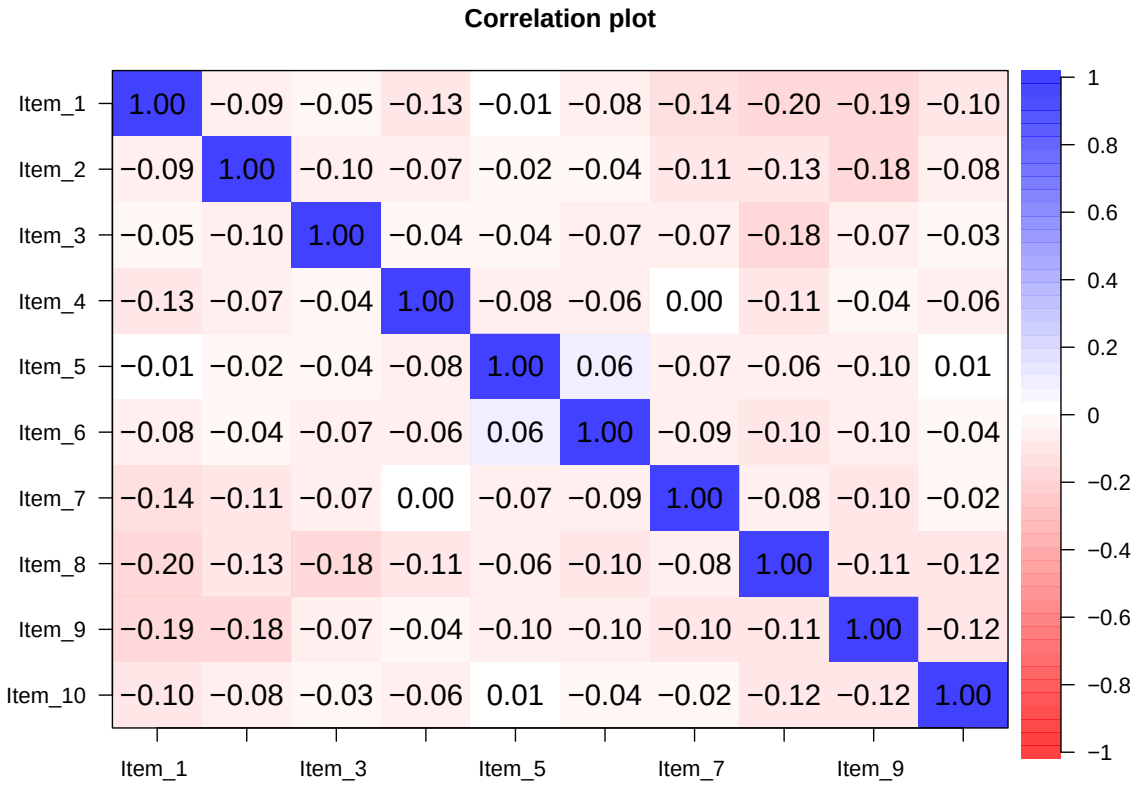
```
##       item    S_X2 df.S_X2 RMSEA.S_X2 p.S_X2
## 1   Item_1  97.659     101      0.000  0.576
## 2   Item_2 100.111      99      0.003  0.450
## 3   Item_3 120.249     110      0.010  0.237
## 4   Item_4 129.576     111      0.013  0.110
## 5   Item_5  94.545     119      0.000  0.952
## 6   Item_6 115.804     111      0.007  0.358
## 7   Item_7  93.369     109      0.000  0.857
## 8   Item_8  97.959      91      0.009  0.290
## 9   Item_9  99.968     100      0.000  0.482
## 10 Item_10 100.050     109      0.000  0.718
```

With 10 items, it is getting difficult to examine the residual correlations for each item pair. This is where the plot below helps with seeing the big picture. The darker the cells, the stronger item dependence. Overall, the residual correlations look excellent, suggesting that the GRM structure captures the data well and the local

independence assumption holds.

```
psych::cor.plot(residuals(mod_GRM, type = "Q3"))
```

```
## Q3 summary statistics:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.201  -0.108  -0.081  -0.080  -0.043   0.059
##
##         Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## Item_1   1.000 -0.087 -0.051 -0.128 -0.006 -0.083 -0.138 -0.201 -0.195  -0.099
## Item_2  -0.087  1.000 -0.096 -0.073 -0.021 -0.039 -0.110 -0.128 -0.182  -0.081
## Item_3  -0.051 -0.096  1.000 -0.040 -0.043 -0.070 -0.074 -0.177 -0.069  -0.032
## Item_4  -0.128 -0.073 -0.040  1.000 -0.082 -0.060  0.001 -0.108 -0.038  -0.062
## Item_5  -0.006 -0.021 -0.043 -0.082  1.000  0.059 -0.066 -0.061 -0.096   0.007
## Item_6  -0.083 -0.039 -0.070 -0.060  0.059  1.000 -0.087 -0.105 -0.098  -0.039
## Item_7  -0.138 -0.110 -0.074  0.001 -0.066 -0.087  1.000 -0.081 -0.103  -0.023
## Item_8  -0.201 -0.128 -0.177 -0.108 -0.061 -0.105 -0.081  1.000 -0.113  -0.119
## Item_9  -0.195 -0.182 -0.069 -0.038 -0.096 -0.098 -0.103 -0.113  1.000  -0.118
## Item_10 -0.099 -0.081 -0.032 -0.062  0.007 -0.039 -0.023 -0.119 -0.118   1.000
```




### Extract and Interpret Item Parameters

Here we see the estimated item parameters. Notably, since each item now has k-1 difficult threshold parameters, the "b1" to "b4" represent the threshold parameters.

```
coef(mod_GRM, simplify=TRUE, IRTpars=TRUE)$items    # IRT item coefficients (a1, d1, d2...)
```

```
##                a         b1          b2         b3        b4
## Item_1  1.5166934 -0.9160199 -0.35005984 0.34003123 0.9750856
## Item_2  1.2520086 -1.1025215 -0.21151487 0.57261091 0.7853211
## Item_3  0.9694925 -0.9202297 -0.02102623 0.90681168 1.8726810
```

```
## Item_4  0.7977481 -1.6948919 -0.43173490 0.84725778 1.9058203
## Item_5  0.4813480 -1.8305429  0.17234033 2.31167682 4.5066784
## Item_6  0.8181997 -1.9998437 -0.33029364 0.78525108 1.7503033
## Item_7  1.0075115 -1.1063446 -0.50756397 0.05587462 1.3562410
## Item_8  1.6564978 -0.9885546 -0.31655246 0.42671752 0.6397333
## Item_9  1.4286696 -1.1333434 -0.59917257 0.04753364 1.2388271
## Item_10 0.9178022 -0.6034594 -0.20126945 1.06880186 1.9285705
```

### Obtaining Factor Scores

For the estimated latet ability, we can see that person 1 has an ability of 0.77 SD above the mean latent ability, person 2 has an ability of 1.22 SD below the mean, and so on.

```
theta_hat <- fscores(mod_GRM, method = "EAP", full.scores.SE = TRUE)
head(theta_hat)
```

```
##              F1     SE_F1
## [1,]  0.7764823 0.4781426
## [2,] -1.2202031 0.5109094
## [3,] -0.9186083 0.5268936
## [4,]  0.6732105 0.4508327
## [5,]  0.9776169 0.4799224
## [6,] -0.6349674 0.4403093
```

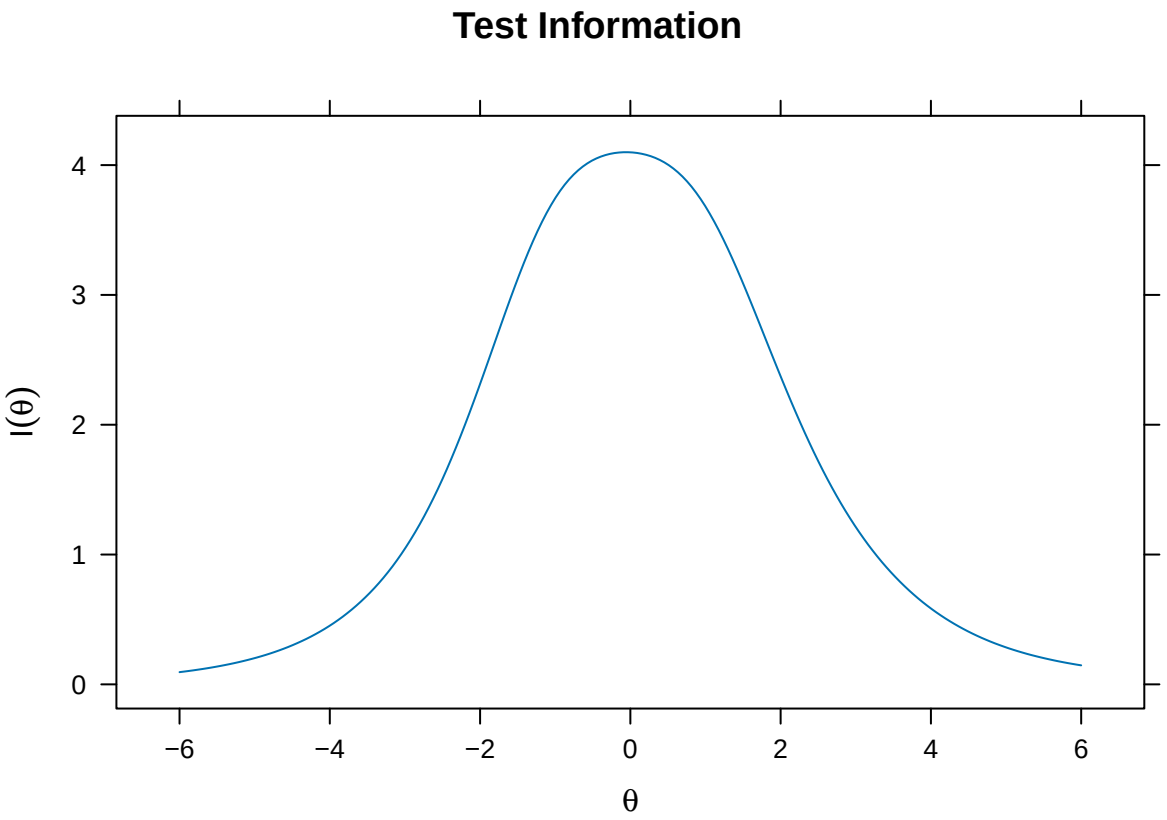
### Plotting (Test-level Plot vs Item-level Plot)

The graphics are very similar to that of 2PL, the biggest difference is that GRM has thresholds for each item, which is evident from the item probability functions.

### Test-level Plots

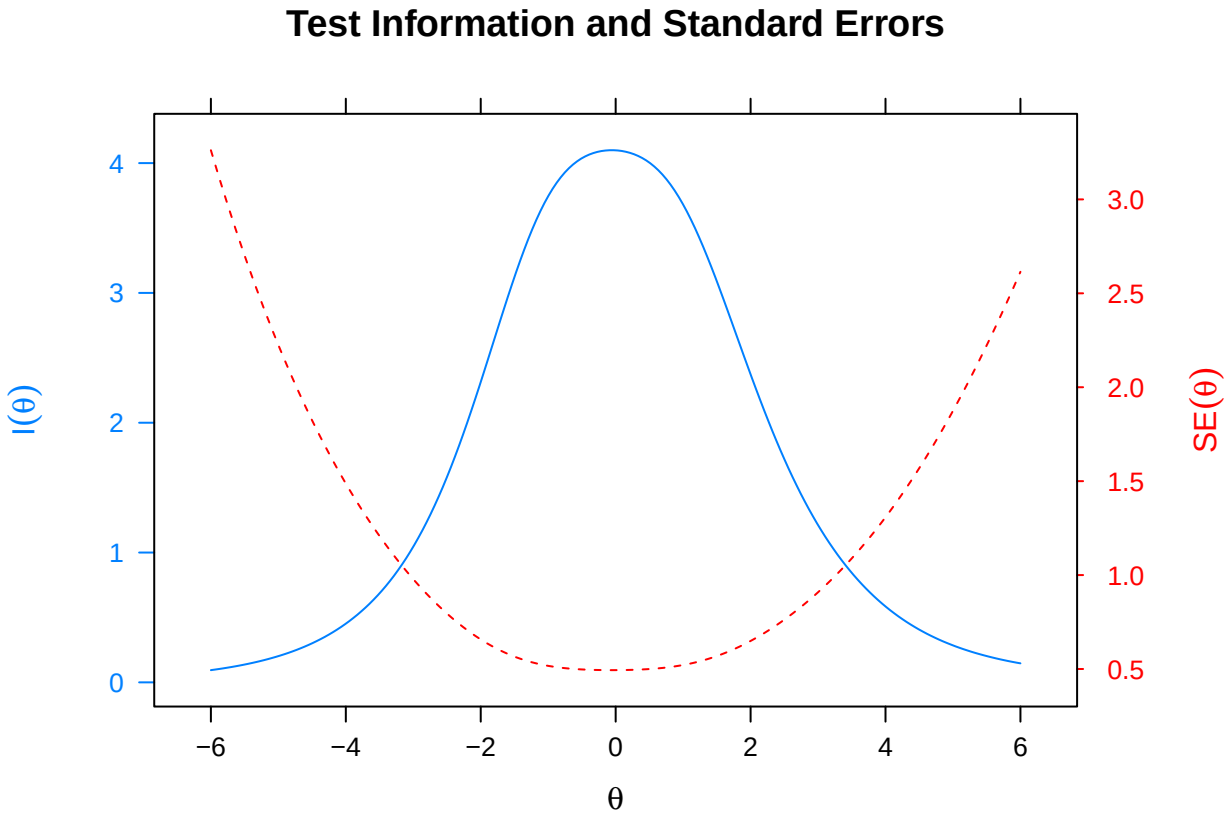
**Test Information Curves** The test information plot shows where on the latent trait the test is the most precise. For this set of simulated data, information is centred around 0, so it is most precise for assessing individuals with average ability.

```
plot(mod_GRM, type = 'info')  # Test Information Curve
```



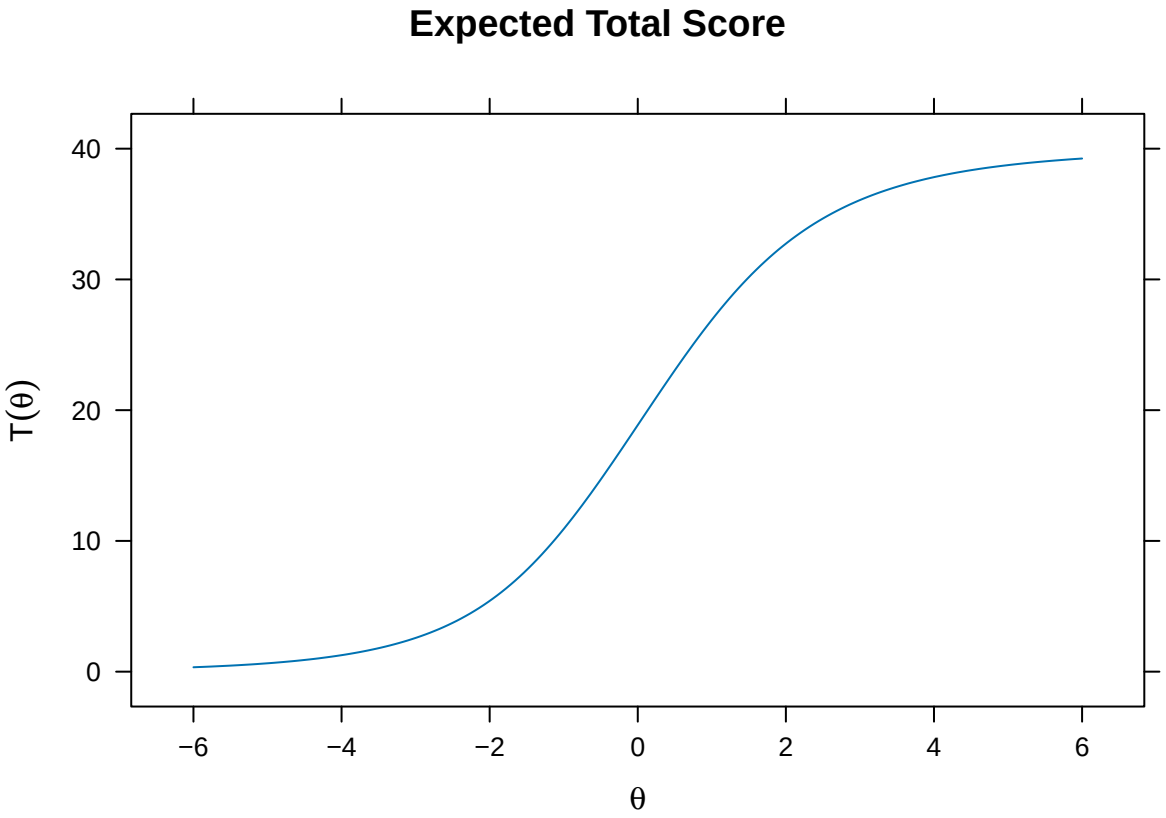

Here, the plot combined the test information curve with the corresponding SE.

```
plot(mod_GRM, type = "infoSE")  # Test Information + SE
```




**Expected Total Score Plot** The expected total score plot looks at the relation between $\theta$ and the number of correct scores.

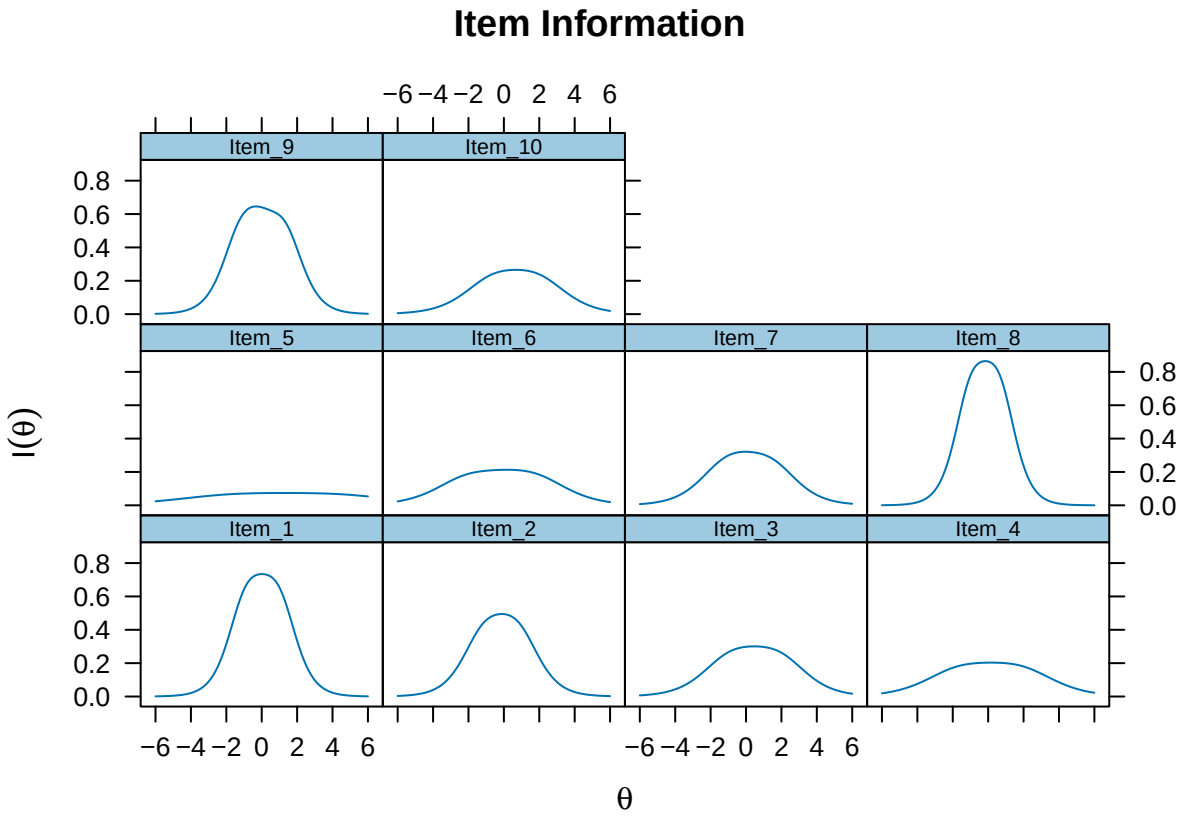
```
plot(mod_GRM) # Expected Total Score (S shape)
```



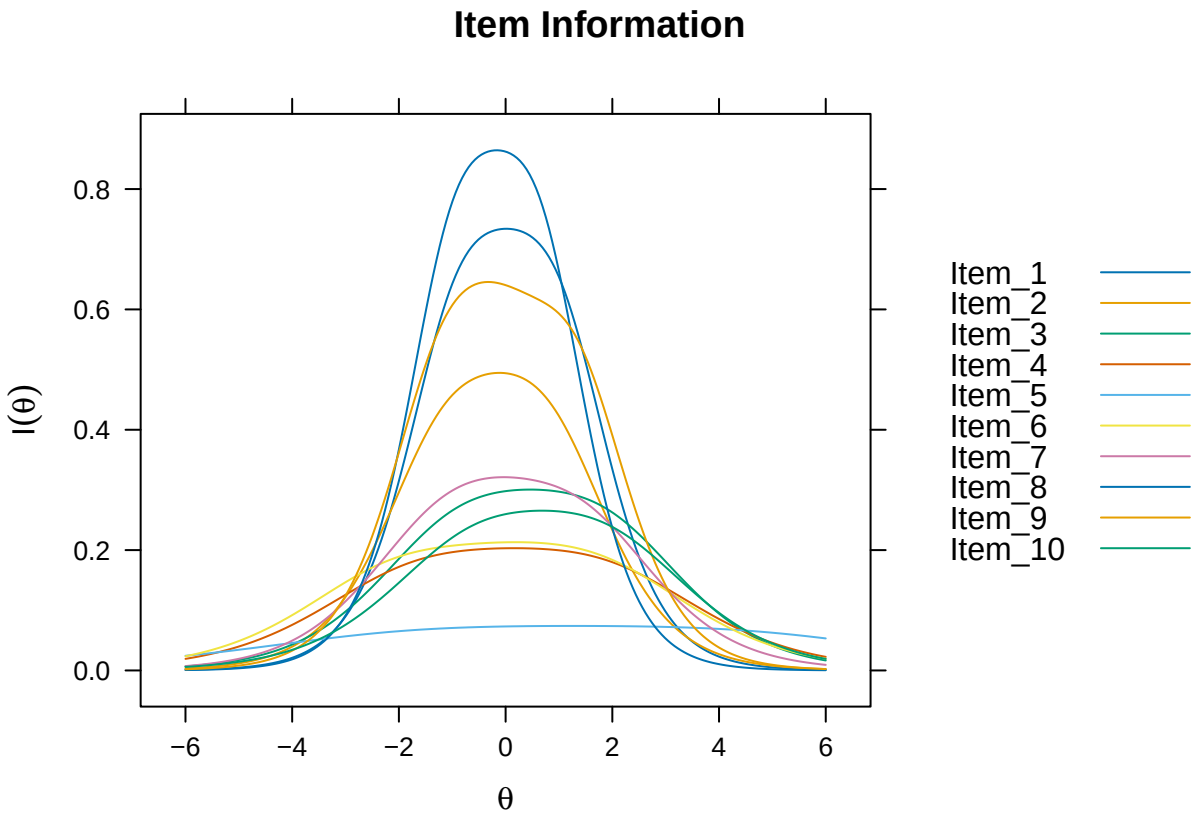

**Item-level Plots** We can look further into which specific items have the most or least information, and which items are relatively hard or easy compare to the rest by looking at item-level plots.

**Item Information Curves** For item information curves, we see that item 5, item 6, and item 4 has the lowest level of information with non-distinguishable peaks, whereas the other items have noticeable peaks centred around 0.

```
plot(mod_GRM, type = "infotrace")
```



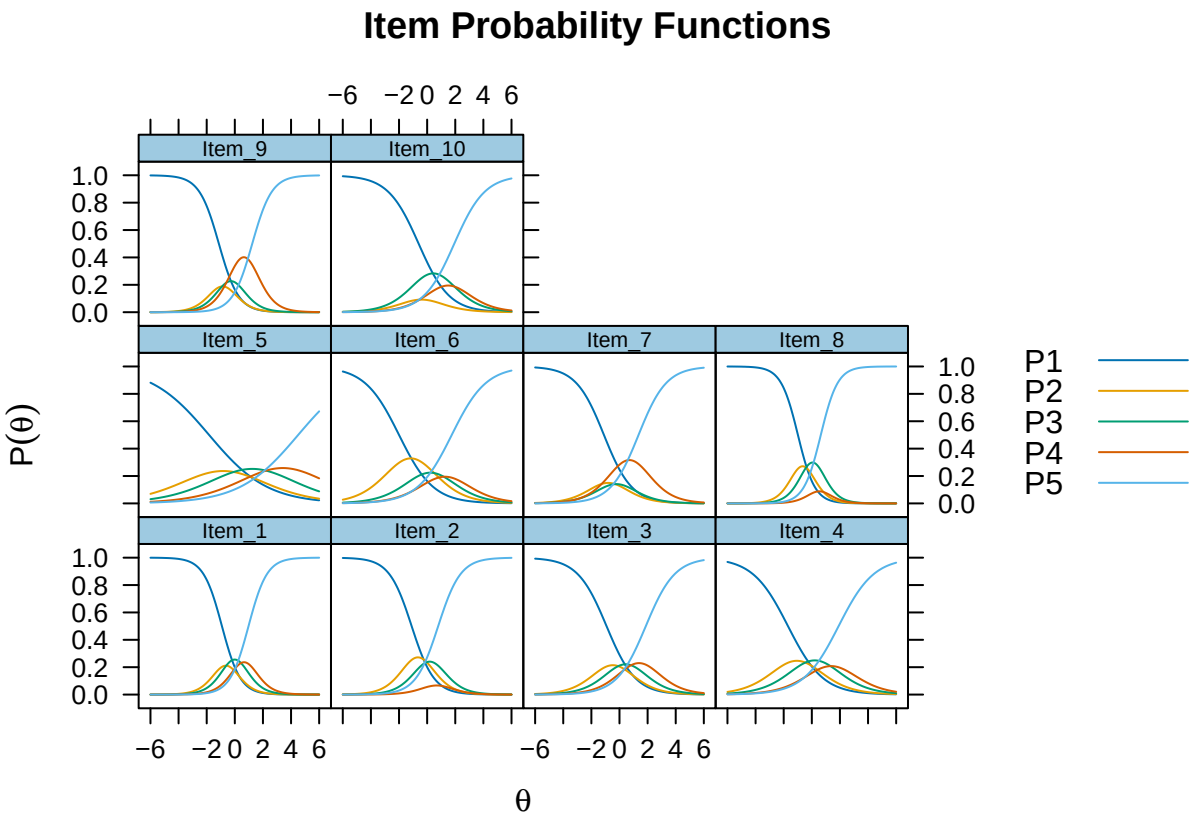

```
plot(mod_GRM, type = "infotrace", facet_items = FALSE)
```



**Item Probability Functions** The item probability functions plot below show the relation between the ability level and the corresponding probability of getting the item correct for each item. Since GRM has threshold categories, we can see the response pattern for each threshold as well.

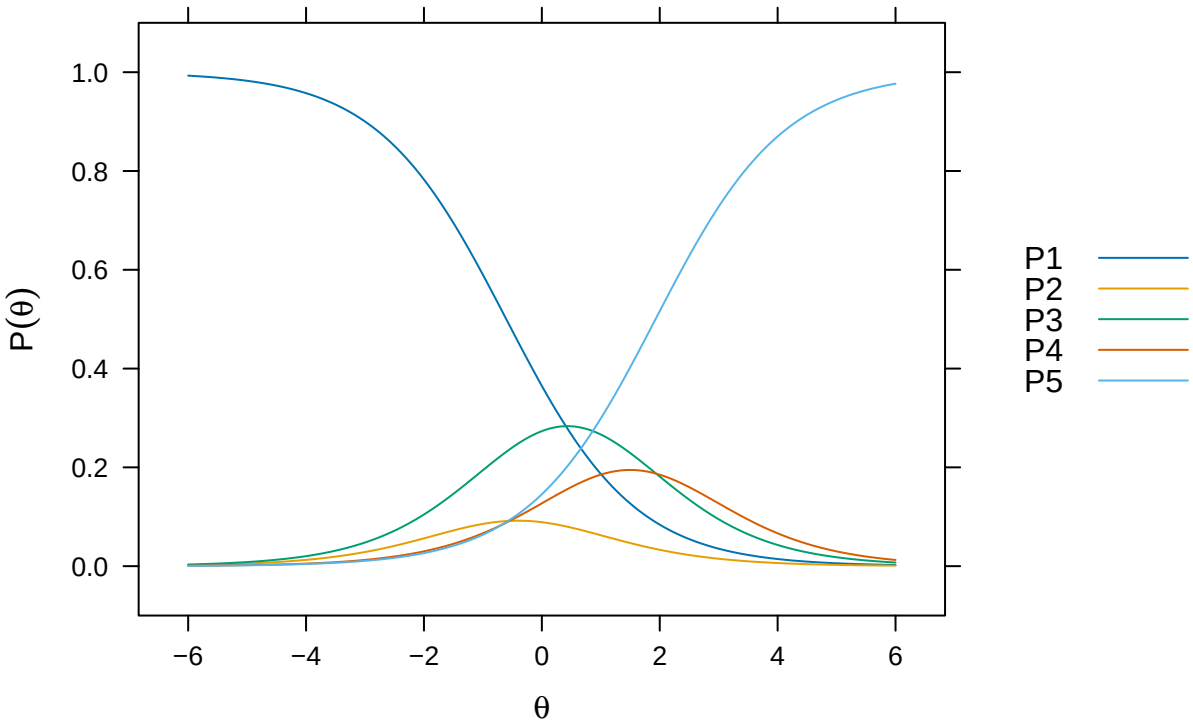
Here we can clearly see that individuals with low ability levels have a high probability choosing category 1 (the dark blue curve) and individuals with high ability levels have a high probability choosing category 5 (the light blue curve). Whereas the other three categories tend to overlap in the middle.

```
plot(mod_GRM, type = "trace")
```




We can also look at a single item, for example item 10.

```
itemplot(mod_GRM, 10)    # For item 10 only
```




## Model Fitting: Bi-factor Model Polytomous

Many psychological inventories are believed to be multidimensional in nature (Reise, 2012; Reise et al., 2013). Therefore, I decided to also include fitting a IRT bifactor model in this tutorial.

Bifactor models generally assume (1) orthogonality, where the general and specific factors are not allowed to correlate, (2) no cross-loadings, that is each item can only load on one general and one specific factor (not multiple specific factors), and (3) local independence, that is responses to one item is independent of another item's response after accounting for the general and specific trait. Notice that for the two unidimensional models above, local independence is conditioned on the one general latent trait, whereas bifactor models' local independence assumption is conditioned on the general and specific traits. In other words, once the general and specific traits are accounted for, the residual correlations should be minimal.

This simulated dataset is a measure with 12 items, the first 6 loads onto the first specific factor and the second 6 loads onto the second specific factor. The item type is ordered with five categories. There are 500 individuals' responses.

```
set.seed(7992)

N <- 500  # N individuals
j <- 12  # N items
cats <- 5  # 5 categories (so 4 thresholds)

# Create discriminant matrix with 0s as placeholder
a <- matrix(0, nrow = j, ncol = 3)
colnames(a) <- c("G", "S1", "S2")

# Fill in the matrix with numbers
a[, "G"] <- rlnorm(j, 0.3, 0.2)    # mean a ~ e^(0.3+0.02) = 1.38 (higher discriminant)
```

```r
a[1:6, "S1"] <- rlnorm(j/2, meanlog = -0.7, sdlog = 0.2)   # mean a ~ e^(-0.7+0.02) = 0.51
a[7:12, "S2"] <- rlnorm(j/2, meanlog = 0.2, sdlog = 0.2)

# Create discriminant matrix with 0s as placeholder
d <- matrix(0, nrow = j, ncol = cats-1)
for(i in 1:j){
  b <- sort(rnorm(4, 0, 1))

  # get a parm [row = i, col = "G"] = vector of c("a", "b1", b2"..."b4")
  x <- c(a[i, "G"], b)

  # traditional2mirt(x, cls = 'graded', ncat = cats)
  pars <- traditional2mirt(x, cls = "graded", ncat = cats)

  d[i, ] <- pars[2:5]
}

# Create latent traits from normal distribution
Theta <- matrix(rnorm(N * 3), ncol = 3)
colnames(Theta) <- c("G", "S1", "S2")

# Actually simulate response data now
dat3 <- simdata(a = a, d = d, N = N, itemtype = "graded", Theta = Theta)
head(dat3)
```

```
##      Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## [1,]      4      3      2      4      2      3      0      2      1       2
## [2,]      0      0      0      0      0      0      1      2      1       0
## [3,]      4      4      4      3      4      4      4      4      1       3
## [4,]      1      0      3      0      4      1      2      4      1       2
## [5,]      4      4      2      4      4      1      1      4      0       2
## [6,]      4      4      4      4      1      0      2      2      1       0
##      Item_11 Item_12
## [1,]       1       2
## [2,]       0       0
## [3,]       2       3
## [4,]       3       0
## [5,]       2       0
## [6,]       3       0
```

Looking at the item stats, we see that for item 7, there seem to be missing a category (NA). This may be due to the threshold generation process. It is not problematic for our demonstration purpose, especially since the model still converged, but in practice, one may wish to look into it more.

```r
itemstats(dat3)
```

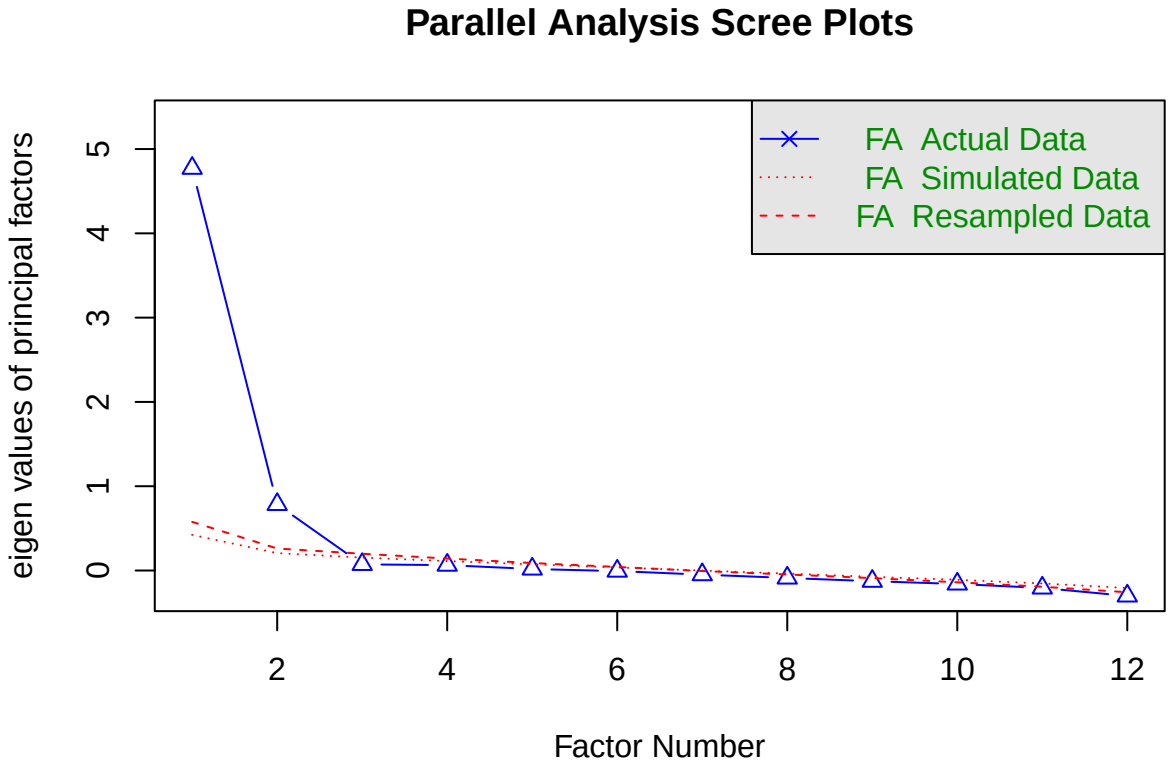
```
## $overall
##    N mean_total.score sd_total.score ave.r  sd.r alpha SEM.alpha
##  500             27.2         11.572 0.316 0.093 0.841     4.609
##
## $itemstats
##           N K  mean    sd total.r total.r_if_rm alpha_if_rm
## Item_1  500 5 2.408 1.620   0.695         0.612       0.821
## Item_2  500 5 2.454 1.720   0.596         0.486       0.831
```

```
## Item_3  500 5 3.002 1.287   0.543         0.457       0.833
## Item_4  500 5 2.546 1.863   0.566         0.441       0.836
## Item_5  500 5 2.510 1.796   0.584         0.468       0.833
## Item_6  500 5 2.092 1.648   0.546         0.434       0.835
## Item_7  500 4 1.936 1.459   0.698         0.625       0.821
## Item_8  500 5 2.376 1.599   0.534         0.424       0.835
## Item_9  500 5 1.938 1.347   0.660         0.586       0.825
## Item_10 500 5 1.994 1.388   0.691         0.620       0.822
## Item_11 500 5 2.280 1.776   0.569         0.451       0.834
## Item_12 500 5 1.664 1.549   0.636         0.546       0.826
##
## $proportions
##             0     1     2     3     4
## Item_1  0.214 0.154 0.034 0.206 0.392
## Item_2  0.276 0.064 0.050 0.150 0.460
## Item_3  0.084 0.036 0.214 0.126 0.540
## Item_4  0.338 0.002 0.028 0.040 0.592
## Item_5  0.278 0.086 0.048 0.024 0.564
## Item_6  0.290 0.144 0.050 0.216 0.300
## Item_7  0.156 0.336 0.216    NA 0.292
## Item_8  0.274 0.016 0.090 0.300 0.320
## Item_9  0.168 0.314 0.050 0.348 0.120
## Item_10 0.228 0.076 0.374 0.118 0.204
## Item_11 0.322 0.062 0.076 0.094 0.446
## Item_12 0.396 0.108 0.058 0.312 0.126
```

The bifactor model requires explicit identifications of which items load onto which specific factor. Often times, theoretical justifications would give researchers a good idea on this. Though it is also common to do some sort of exploratory work, such as conducting parallel analysis to identify the number of factors and running exploratory factor analysis (EFA) to understand the item loading patterns.

Parallel analysis below suggest a 2 factor structure, align with our simulated data structure.

```
fa.parallel(dat3, fa="fa", show.legend=T, fm = 'minres', cor = 'poly', n.iter=100)
```




```
## Parallel analysis suggests that the number of factors =  2  and the number of components =  NA
```

EFA shows strong loadings for item 1 to item 6 on the first factor and item 7 to item 12 on the second factor. Though item 10 seem to have moderate cross loadings with the first factor.

```
twoFmod <- fa(dat3, nfactors=2, SMC=T, cor = 'poly', fm = 'minres', rotate = "oblimin")
print(twoFmod$loadings, cutoff = 0.3)
```

```
##
## Loadings:
##         MR1    MR2
## Item_1   0.787
## Item_2   0.500
## Item_3   0.719
## Item_4   0.589
## Item_5   0.624
## Item_6   0.619
## Item_7          0.730
## Item_8          0.767
## Item_9          0.510
## Item_10  0.371  0.442
## Item_11         0.755
## Item_12         0.484
##
##                  MR1   MR2
## SS loadings    2.823 2.428
## Proportion Var 0.235 0.202
## Cumulative Var 0.235 0.438
```

Here we specified the items that load on each specific factor to fit the bifactor model.

```r
# Fit bifactor model
model <- "S1 = 1-6
          S2 = 7-12"
mod_bfactor <- bfactor(dat3, model, itemtype = "graded")
```

```
## "Item_7" re-mapped to ensure all categories have a distance of 1
```

```r
# Check whether model converged
mod_bfactor@OptimInfo$converged
```

```
## [1] TRUE
```

### Model and Item Fit

Model fit can be obtained via `M2()` function. For more complex bifactor models, `QMC = T` should be supplied for accurate estimation. Model fit looks good for this toy example, $\chi^2(7) = 8.58, p = 0.284$. Again, obtaining DFI from simulated data may give us a better picture of what cut-off is considered "good".

```r
M2(mod_bfactor)
```

```
##          M2 df     p RMSEA RMSEA_5 RMSEA_95 SRMSR   TLI   CFI
## stats 8.583  7 0.284 0.021       0    0.062 0.029 0.974 0.994
```

Item fit also looks good; no items with significant misfit.

```r
itemfit(mod_bfactor)
```

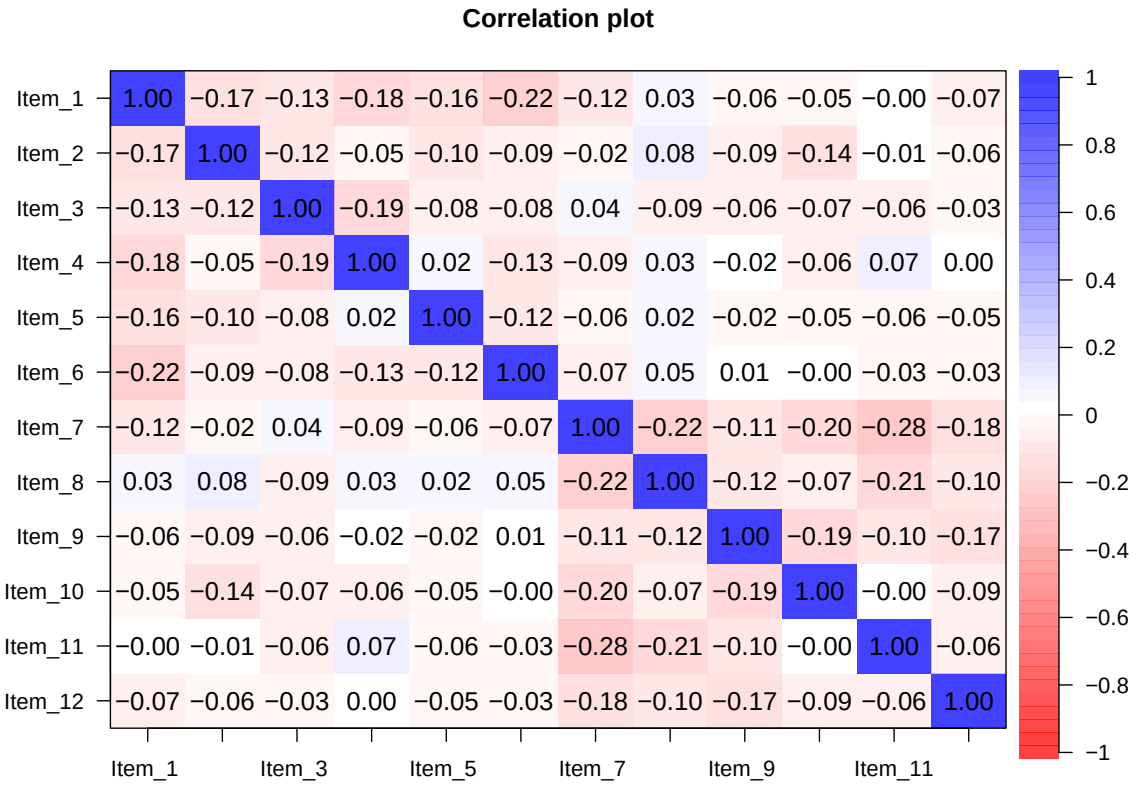
```
##       item    S_X2 df.S_X2 RMSEA.S_X2 p.S_X2
## 1   Item_1  66.340      77      0.000  0.802
## 2   Item_2  98.202      91      0.013  0.284
## 3   Item_3  84.580      85      0.000  0.492
## 4   Item_4  42.098      44      0.000  0.553
## 5   Item_5  62.665      74      0.000  0.823
## 6   Item_6 106.717      97      0.014  0.235
## 7   Item_7  76.057      73      0.009  0.380
## 8   Item_8 107.539      87      0.022  0.067
## 9   Item_9  91.157      72      0.023  0.063
## 10 Item_10  91.699      89      0.008  0.401
## 11 Item_11  93.703      93      0.004  0.460
## 12 Item_12  89.583      86      0.009  0.374
```

Residual correlation also show no sign of local dependence violation, except for correlations between item 11 and item 7 (-0.28) is a bit large. Though generally implying that the bifactor model captures the data structure well.

```r
psych::cor.plot(residuals(mod_bfactor, type = "Q3"))
```

```
## Q3 summary statistics:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.283  -0.118  -0.070  -0.076  -0.025   0.079
##
##         Item_1 Item_2 Item_3 Item_4 Item_5 Item_6 Item_7 Item_8 Item_9 Item_10
## Item_1   1.000 -0.169 -0.126 -0.184 -0.159 -0.221 -0.115  0.028 -0.058  -0.052
## Item_2  -0.169  1.000 -0.121 -0.046 -0.100 -0.088 -0.025  0.079 -0.093  -0.138
## Item_3  -0.126 -0.121  1.000 -0.188 -0.084 -0.085  0.036 -0.091 -0.062  -0.068
## Item_4  -0.184 -0.046 -0.188  1.000  0.021 -0.129 -0.090  0.032 -0.016  -0.062
```

```
## Item_5  -0.159 -0.100 -0.084  0.021  1.000 -0.116 -0.058  0.020 -0.025  -0.049
## Item_6  -0.221 -0.088 -0.085 -0.129 -0.116  1.000 -0.071  0.052  0.013  -0.001
## Item_7  -0.115 -0.025  0.036 -0.090 -0.058 -0.071  1.000 -0.221 -0.113  -0.197
## Item_8   0.028  0.079 -0.091  0.032  0.020  0.052 -0.221  1.000 -0.119  -0.074
## Item_9  -0.058 -0.093 -0.062 -0.016 -0.025  0.013 -0.113 -0.119  1.000  -0.188
## Item_10 -0.052 -0.138 -0.068 -0.062 -0.049 -0.001 -0.197 -0.074 -0.188   1.000
## Item_11  0.000 -0.008 -0.060  0.070 -0.058 -0.025 -0.283 -0.207 -0.104  -0.004
## Item_12 -0.072 -0.056 -0.031  0.004 -0.047 -0.035 -0.175 -0.103 -0.170  -0.091
##         Item_11 Item_12
## Item_1    0.000  -0.072
## Item_2   -0.008  -0.056
## Item_3   -0.060  -0.031
## Item_4    0.070   0.004
## Item_5   -0.058  -0.047
## Item_6   -0.025  -0.035
## Item_7   -0.283  -0.175
## Item_8   -0.207  -0.103
## Item_9   -0.104  -0.170
## Item_10  -0.004  -0.091
## Item_11   1.000  -0.063
## Item_12  -0.063   1.000
```




### Extract and Interpret Item Parameters

For the estimated parameters, we see k-1 thresholds for the polytonomous scoring and three discriminant parameters: one for the general factor (a1), one for the first specific factor (a2), and one for the second specific factor (a3). Item 1 and item 7 have the highest general discriminant value (Item 1 = 1.99, Item 7 = 2.02), indicating their ability to differentiate latent abilities well. Most of the items have general factor discrimination values between 1.0 and 1.5, reflecting good ability to differentiate individuals with similar

latent ability.

```
coef(mod_bfactor, simplify=TRUE, IRTpars=TRUE)$items    # IRT item coefficients (a1, d1, d2...)
```

```
## Warning: Traditional parameterization only available for unidimensional models
## or models with simple structure patterns (neither found)
```

```
##                a1        a2        a3        d1          d2          d3
## Item_1  1.9872579 1.1372501 0.0000000 2.3345914  1.01297163  0.74314190
## Item_2  1.3404187 0.4592130 0.0000000 1.3279510  0.93489624  0.64740119
## Item_3  1.3755117 0.7822236 0.0000000 3.2469265  2.75196959  1.02579018
## Item_4  1.0653719 0.7678494 0.0000000 0.8953898  0.88347222  0.71956445
## Item_5  1.2354477 0.4843976 0.0000000 1.2788276  0.75434011  0.47997239
## Item_6  0.9777925 0.9812626 0.0000000 1.2017966  0.33132696  0.05393628
## Item_7  2.0181271 0.0000000 1.3671182 3.0919984  0.11399302 -1.64751365
## Item_8  0.9487845 0.0000000 1.5810161 1.5354489  1.41550552  0.78181505
## Item_9  1.5461014 0.0000000 0.7044853 2.3840989  0.20734901 -0.10512855
## Item_10 1.8411183 0.0000000 0.5003145 1.9463962  1.34468237 -1.17872837
## Item_11 1.0968848 0.0000000 1.2543702 1.1301940  0.73100711  0.26137831
## Item_12 1.4275713 0.0000000 0.6151427 0.6002552 -0.05229442 -0.39035888
##                 d4
## Item_1  -0.8020734
## Item_2  -0.1947808
## Item_3   0.2697679
## Item_4   0.4949995
## Item_5   0.3471420
## Item_6  -1.1627912
## Item_7          NA
## Item_8  -1.1822536
## Item_9  -2.8585549
## Item_10 -2.1296183
## Item_11 -0.3088794
## Item_12 -2.6749776
```

### Obtaining Factor Scores

When it comes to scoring bifactor structures, a natural question is whether the specific factors should be scored independently as stand-alone, interpretable sub-sets or whether a total, general factor score is more appropriate. One piece of evidence that can support a dominant general trait is by looking at the empirical reliability estimates. Here we see that the general factor has an empirical reliability of 0.76 and the specific factors have reliability of 0.34 and 0.48 respectively. This suggest that the there is a general, dominant factor, and the specific factors are not strong enough to stand on its own to score as sub-scores. The specific factors are simply there to model the local dependence among item bundles but have no substantial meaning.

```
theta_hat <- fscores(mod_bfactor, full.scores.SE = TRUE, method = 'EAP')
head(theta_hat)
```

```
##               G         S1         S2      SE_G     SE_S1     SE_S2
## [1,] -0.2375530  0.4793605 -0.6599039 0.4488865 0.7858389 0.6442333
## [2,] -1.6583765 -1.1131750  0.2010464 0.5136105 0.8564876 0.6783254
## [3,]  0.7818964  0.5479247  0.2114639 0.4788680 0.8264422 0.7240948
## [4,] -0.5270972 -0.3377548  0.7201295 0.4510503 0.7631654 0.6825296
## [5,] -0.2215978  0.5115851 -0.1066335 0.4802825 0.8001001 0.7088168
## [6,] -0.1488742  0.5016984 -0.2879946 0.4831368 0.8330056 0.6498959
```

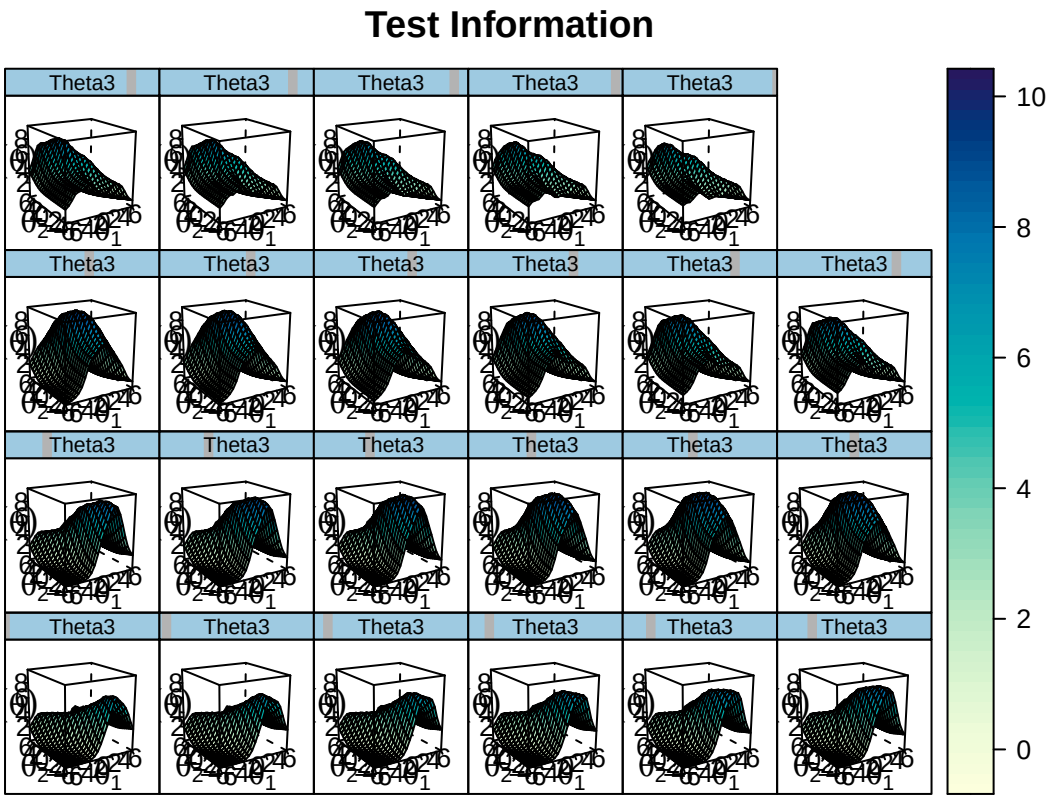
```
empirical_rxx(theta_hat)
```

```
##         G        S1        S2
## 0.7641764 0.3361837 0.4839027
```

### Plotting

Since bifactor models are multidimensional models, the plots get harder to interpret visually. For example, the test information plot for each item does not look intutive.

```
plot(mod_bfactor, type = 'info')  # Test Information Curve for each item
```




One way to still present a two dimensional plot when the general factor is of interest is by collapsing the irrelevant dimensions into a single latent trait through marginalization. That is by obtaining unidimensional parameters from inherit multidimensional data structures. This is the core idea of projective item response theory (PIRT) models (Reise et al., 2025). After obtaining the projected unidimensional parameters, we can proceed to plot and score as usual. The application of PIRT models will soon be available (*shhhh - top secret*) in the `mirt` package (Chalmers, 2012). For now, one can enjoy gazing the multiple "mountain peaks" above.

## Other Notes

Often times, dichotomous data do not come cleanly with correct and incorrect scores (e.g., 1/0), but the raw responses from the individual. In those cases, the `key2binary()` function can assign binary scores to the items based on the key supplied.

```
# Toy example code
newdat <- key2binary(toydat, key = c(2,3,1,2,5))
head(newdat)
```

# References


Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, *48*(6), 1–29. https://doi.org/10.18637/jss.v048.i06

de Ayala, R. J. (2022). *The theory and practice of item response theory.* Second edition. New York, NY: Guilford Publications.

Liu, J., Bao, Y., DiStefano, C., & Jiang, W. (2025). Collapsing Sparse Responses in Likert-Type Scale Data: Advantages and Disadvantages for Model Fit in CFA. *Educational and psychological measurement*, *0*(0). https://doi.org/10.1177/00131644251401097

McNeish, D. & Wolf, M. G. (2023). Dynamic fit index cutoffs for confirmatory factor analysis Models. *Psychological Methods*, *28*(1), 61 - 88. https://doi.org/10.1037/met0000425

R Core Team. (2026). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen, Denmark: Danish Institute for Educational Research.

Reise, S. P., Block, J. M., Mansolf, M., Haviland, M. G., Schalet, B. D., & Kimerling, R. (2025). Using projective irt to evaluate the effects of multidimensionality on a unidimensional IRT model parameters. *Multivariate Behavioral Research*, *60*(2), 345–361. https://doi.org/10.1080/00273171.2024.2430630

Reise, S. P., Bonifay, W., & Haviland, M. G. (2013). Scoring and modeling psychological measures in the presence of multidimensionality. *Journal of Personality Assessment*, *95*(2), 129–140. https://doi.org/10.1080/00223891.2012.725437

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monographs*, *34*(17), 1–97. https://doi.org/10.1007/bf03372160

Thissen, D., Nelson, L., Rosa, K., & McLeod, L. D. (2001). Item Response Theory for Item Scored in More Than Two Categories. In Thissen, D., & Wainer, H. (Eds.), *Test scoring.* (pp.141-184). Lawrence Erlbaum Associates Publishers.

Wolf, M. G. & McNeish, D. (2020). Dynamic Model Fit. R Shiny application version 1.1.0