psych

MDPI

Tutorial

Flexible Item Response Modeling in R with the flexmet Package

Leah Feuerstahler

Department of Psychology, Fordham University, 441 E Fordham Road, Bronx, NY 10458, USA; lfeuerstahler@fordham.edu

Abstract: The filtered monotonic polynomial (FMP) model is a semi-parametric item response model that allows flexible response function shapes but also includes traditional item response models as special cases. The `flexmet` package for `R` facilitates the routine use of the FMP model in real data analysis and simulation studies. This tutorial provides several code examples illustrating how the `flexmet` package may be used to simulate FMP model parameters and data (both for dichotomous and polytomously scored items), estimate FMP model parameters, transform traditional item response models to different metrics, and more. This tutorial serves as both an introduction to the unique features of the FMP model and as a practical guide to its implementation in `R` via the `flexmet` package.

Keywords: item response theory; psychometric scaling; software tutorial

Citation: Feuerstahler, L. Flexible Item Response Modeling in R with the flexmet Package. *Psych* **2021**, *3*, 447–478. https://doi.org/10.3390/psych3030031

Academic Editor: Alexander Robitzsch

Received: 15 July 2021
Accepted: 11 August 2021
Published: 16 August 2021

1. Background

Although many applications of item response theory are in the context of parametric models such as the Rasch, two, and three-parameter logistic models [1], there is also a recognized need for models that allow for more flexible relationships between the latent trait and an item category response [2]. Models and techniques that allow for more flexible item response functions are variously known as non-parametric [2], quasi-parametric [3], or semi-parametric [4] models, and include methods such as Mokken scale analysis [5], kernel smoothing [6], and polynomial splines [4]. Historically, flexible item response models have been used to analyze data sets with small sample sizes, to check the assumptions of parametric item response models, and as an alternative to poorly fitting parametric models [2]. In recent years, flexible item response models have increasingly been used in confirmatory contexts. For example, flexible item response models have recently been applied to computerized adaptive testing [7,8], the creation of item banks for measuring health outcomes [9], and the development of optimal scoring procedures [10].

A compelling recent addition to the family of semi-parametric item response models is the filtered monotonic polynomial (FMP) model [3,11]. The general form of the FMP model [12] used in this paper is a generalization of Muraki's generalized partial credit model [13] (GPCM) that replaces a linear function of the latent trait θ with a polynomial expansion of θ. As clarified below, constraints are placed on the item parameters so that this polynomial function is a monotonically increasing function of θ. For item i with C_i ordered response categories and person j, the item response function (IRF) of the general FMP model gives the probability of a response in category c, $c = 0, \ldots C_i - 1$, as

$$P(X_{ij} = c|\theta_j, \boldsymbol{b}_i) = \frac{\exp\left(\sum_{v=0}^{c}(b_{0_v i} + m_i^\star(\theta_j, b_{1i}, \ldots b_{2k_i+1,i}))\right)}{\sum_{u=0}^{C_i-1}\exp\left(\sum_{v=0}^{u}(b_{0_v i} + m_i^\star(\theta_j, b_{1i}, \ldots b_{2k_i+1,i}))\right)}, \quad (1)$$

where

$$m_i^\star(\theta_j, b_{1i}, \ldots b_{2k_i+1,i}) = b_{1i}\theta_j + b_{2i}\theta_j^2 + \cdots + b_{2k_i-1,i}\theta_j^{2k_i-1}, \quad (2)$$

$\sum_{v=0}^{0}(b_{0_v i} + m_i^\star(\theta_j, b_{1i}, \ldots b_{2k_i+1,i})) \equiv 0$, and $\boldsymbol{b}_i = (b_{0_1 i}, \ldots, b_{0_{C_i-1} i}, b_{1i}, b_{2i}, \ldots, b_{(2k_i+1)i})'$ is a vector of item parameters in a polynomial coefficient parameterization.

In Equations (1) and (2), the k_i parameter is an item-specific non-negative integer that controls the maximum degree of the polynomial function of θ. Specifically, the highest-order polynomial equals $2k_i + 1$, and so $k_i = 0$, 1, and 2 imply linear, cubic, and fifth-degree polynomial functions of θ. Note that this formulation forces an *odd* value for the highest-order polynomial of θ, which is a necessary (but not sufficient) condition for the polynomial to be a monotonic function of θ.

A key feature of the FMP model is that it reduces to familiar parametric item response models when $m^\star$ is set to be a linear function of θ (i.e., when $k_i = 0$). Specifically, if $k_i = 0$, Equation (1) reduces to the GPCM, and if both $k_i = 0$ and $C_i = 2$ (i.e., scored item responses are dichotomous), then Equation (1) reduces to the two-parameter logistic (2PL) model. Another key feature of this model not shared by many flexible item response models is that its model parameters are portable [3], meaning that the FMP model can be used to construct item banks, conduct adaptive testing, and score examinees not included in the original sample. To better acquaint the reader with the relationship between the FMP model with $k = 0$ and other item response models used in popular IRT software packages, several examples of finding FMP parameters from the output of the R packages `ltm` [14], `mirt` [15], and `TAM` [16] are included in Appendix A.

The FMP model requires that $m^\star(\theta)$ be a *monotonically increasing* function of θ. To enforce monotonicity, we may use a transformation of the polynomial coefficient parameters $\boldsymbol{b}$. In general, consider the polynomial function

$$m(\theta|\boldsymbol{b}) = b_0 + b_1\theta + b_2\theta^2 + b_3\theta^3 + \cdots b_{2k+1}\theta^{2k+1}. \tag{3}$$

Equation (3) is a strictly monotonically increasing function of θ if and only if the first derivative of $m(\theta)$,

$$\frac{\partial m(\theta)}{\partial \theta} = b_1 + 2b_2\theta + 3b_3\theta^2 + \cdots + (2k+1)\theta^{2k}, \tag{4}$$

is positive at all values of θ. One way to enforce positivity is through the following reparameterization of Equation (4) [3,12,17]:

$$\frac{\partial m(\theta)}{\partial \theta} = \begin{cases} \exp(\omega)\prod_{h=1}^{k}(1 - 2\alpha_h\theta + (\alpha_h^2 + \exp(\tau_h))\theta^2 & \text{if } k > 0 \\ \exp(\omega) & \text{if } k = 0. \end{cases} \tag{5}$$

In this parameterization, no boundary constraints are required for ω, α_h, or τ_h, $h = 1, \ldots k$. Because $b_{0_c i}$ parameters do not affect the monotonicity of $m^\star$, no transformation of these parameters is necessary, but we use the symbol $\xi_{ci} = b_{0_c i}$ to use notation consistently across the different parameterizations. Therefore, $b_{0_c i} = \xi_{ci}$ for $c = 1, \ldots, C_i - 1$, $b_1 = \exp(\omega_i)$, and the $b_2, \ldots, b_{2k_i+1}$ parameters are functions of ω_i, $\boldsymbol{\alpha}_i$, and $\boldsymbol{\tau}_i$ calculated using matrix operations described elsewhere [3,11,12,18,19]. Thus, the FMP model is equivalently represented by the polynomial coefficient parameters $\boldsymbol{b}_i = (b_{0_1 i}, b_{0_2 i}, \ldots, b_{0_{C_i-1} i}, b_{1i}, b_{2i}, \ldots, b_{(2k_i+1)i})'$ and the Greek-letter parameters $(\xi_{1i}, \xi_{2i}, \ldots, \xi_{(C_i-1)i}, \omega, \alpha_{1i}, \tau_{1i}, \ldots, \alpha_{(2k_i+1)i}, \tau_{(2k_i+1)i})'$. In both parameterizations, an item with C_i response categories and item complexity k_i is described by $2k_i + C_i - 1$ item parameters. To better acquaint the reader with the Greek-letter and polynomial coefficient parameterizations, Appendix B includes formulas to calculate the polynomial coefficients from the Greek-letter parameters up to $k = 2$.

2. Specifying the FMP Model in `flexmet`

The `R` package `flexmet` provides broad functionality for specifying, fitting, and transforming the FMP model, and many of its features (relative to version 1.1) are illustrated in the remainder of this paper. The IRF for the FMP model is specified using the polynomial coefficient parameters $\boldsymbol{b}_i$ as described in the previous section. However, the Greek-letter parameterization is also needed when fitting and generating FMP items to ensure mono-

tonicity. The `flexmet` package includes the `greek2b` and `b2greek` functions to navigate between the two different parameterizations. To illustrate these functions, consider a six-item test. Three items (items 1, 2, and 3) are defined for binary item responses, and three items are defined for four-category responses (items 4, 5, and 6). Additionally, items 1 and 4 have $k_i = 0$, items 2 and 5 have $k_i = 1$, and items 3 and 6 have $k_i = 2$. The parameters used for this illustration (both the Greek-letter and polynomial coefficient parameterizations) are printed in Table 1.

Table 1. Example FMP item parameters.

Greek-Letter Parameterization						
Parameter	**Item 1**	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item 6**
ξ_1	0.49	−0.87	−0.54	0.58	0.97	0.79
ξ_2				0.49	0.38	0.15
ξ_3				−0.05	−0.15	−1.20
ω	0.25	−0.57	−0.24	−0.41	0.58	0.57
α_1		−0.63	1.13		0.45	−0.22
α_2			−0.40			−0.59
τ_1		−0.12	−1.76		−1.42	−1.57
τ_2			−1.14			−2.45
Polynomial Coefficient Parameterization						
Parameter	**Item 1**	**Item 2**	**Item 3**	**Item 4**	**Item 5**	**Item 6**
b_{0_1}	0.49	−0.87	−0.54	0.58	0.97	0.79
b_{0_2}				0.49	0.38	0.15
b_{0_3}				−0.05	−0.15	−1.20
b_1	1.28	0.57	0.79	0.66	1.79	1.77
b_2		0.36	−0.57		−0.80	1.43
b_3		0.24	0.03		0.26	0.71
b_4			0.01			0.22
b_5			0.11			0.04

The `greek2b` function outputs the $\boldsymbol{b}$ vector associated with the inputted `xi`, `omega`, and (optionally) `alpha` and `tau` values. For example, the $\boldsymbol{b}$ vector for item 1 can be found as follows.

```
library(flexmet)
greek2b(xi = 0.49, omega = 0.25)
```

```
##        b0        b1
## 0.490000 1.284025
```

As another example, let us find the $\boldsymbol{b}$ vector for item 6. In this case, the `greek2b` function requires the user also to specify the `alpha` and `tau` arguments, as well as multiple `xi` parameters corresponding to the different response categories. Note that the `xi` parameters should be given in order from $\xi_{1i}, \ldots, \xi_{(C_i-1)i}$, and the `alpha` and `tau` vectors should also be ordered $\alpha_{1i}, \ldots, \alpha_{k_i i}$ and $\tau_{1i}, \ldots, \tau_{k_i i}$.

```
z01b <- greek2b(xi = c(0.79, 0.15, -1.20),
                omega = 0.57,
                alpha = c(-0.22, -0.59),
                tau = c(-1.57, -2.45))
```

```
##    b0_1    b0_2    b0_3      b1      b2      b3      b4      b5
##  0.7900  0.1500 -1.2000  1.7683  1.4323  0.7132  0.2183  0.0394
```

It is also possible to represent FMP item parameters with different k values and different numbers of items in the same matrix. To do this, the `NA` symbol may be used to represent higher-order ξ values for items with $C_i < \max(C_i)$. In addition, specifying $\alpha_h = 0$ and $\tau_h = -\infty$ will set the corresponding polynomial coefficients (i.e., $b_{2h,i}$ and $b_{2h+1,i}$) to be equal to 0. For example, to find the matrix of polynomial–coefficient item parameters for the example six-item test, we can bind together calls to `greek2b` that have `xi`, `alpha`, and `tau` arguments of the same length for each item.

```
bmat <- rbind(greek2b(xi = c(0.49, NA, NA), omega = 0.25,
                      alpha = c(0, 0), tau = c(-Inf, -Inf)),
              greek2b(xi = c(-0.87, NA, NA), omega = -0.57,
                      alpha = c(-0.63, 0), tau = c(-0.12, -Inf)),
              greek2b(xi = c(-0.54, NA, NA), omega = -0.24,
                      alpha = c(1.13, -0.40), tau = c(-1.76, -1.14)),
              greek2b(xi = c(0.58, 0.49, -0.05), omega = -0.41,
                      alpha = c(0, 0), tau = c(-Inf, -Inf)),
              greek2b(xi = c(0.97, 0.38, -0.15), omega = 0.58,
                      alpha = c(0.45, 0), tau = c(-1.42, -Inf)),
              greek2b(xi = c(0.79, 0.15, -1.20), omega = 0.57,
                      alpha = c(-0.22, -0.59), tau = c(-1.57, -2.45)))
bmat
```

```
##       b0_1 b0_2  b0_3     b1      b2     b3     b4     b5
## [1,]  0.49   NA    NA 1.2840  0.0000 0.0000 0.0000 0.0000
## [2,] -0.87   NA    NA 0.5655  0.3563 0.2420 0.0000 0.0000
## [3,] -0.54   NA    NA 0.7866 -0.5742 0.0317 0.0147 0.1094
## [4,]  0.58 0.49 -0.05 0.6637  0.0000 0.0000 0.0000 0.0000
## [5,]  0.97 0.38 -0.15 1.7860 -0.8037 0.2645 0.0000 0.0000
## [6,]  0.79 0.15 -1.20 1.7683  1.4323 0.7132 0.2183 0.0394
```

Following from Equation (1), these item parameters can be used to find the probability of responding in each response category as a function of the latent trait θ; that is, the IRF. The `irf_fmp` function calculates item response probabilities for the FMP model given a scalar or vector of latent trait value(s) `theta` and a matrix or vector of item parameters `bmat` in the polynomial coefficient parameterization. It may also be necessary to specify the `maxncat` function, which gives the maximum number of response categories represented in the matrix (such that the first `maxncat`-1 columns are interpreted as ξ parameters). The default value of `maxncat = 2`, so this argument should be specified if `bmat` includes at least one polytomous item. Calls to `irf_fmp` will result in a three-dimensional array with θ values in the first dimension, items in the second dimension, and response categories in the third dimension. For example,

```
theta <- c(-1, 1)
irf_fmp(theta = theta, bmat = bmat, maxncat = 4)
```

```
## , , c = 0
##
##             item 1 item 2 item 3 item 4 item 5 item 6
## theta = -1 0.6887 0.7894 0.8836 0.3256 0.8581 0.4118
## theta = 1  0.1450 0.4271 0.5429 0.0280 0.0051 0.0000
```

```
##
## , , c = 1
##
##            item 1 item 2 item 3 item 4 item 5 item 6
## theta = -1 0.3113 0.2106 0.1164 0.2994 0.1304 0.3800
## theta = 1  0.8550 0.5729 0.4571 0.0970 0.0467 0.0006
##
## , , c = 2
##
##            item 1 item 2 item 3 item 4 item 5 item 6
## theta = -1     NA     NA     NA 0.2517 0.0110 0.1849
## theta = 1      NA     NA     NA 0.3074 0.2374 0.0487
##
## , , c = 3
##
##            item 1 item 2 item 3 item 4 item 5 item 6
## theta = -1     NA     NA     NA 0.1233 0.0005 0.0233
## theta = 1      NA     NA     NA 0.5677 0.7109 0.9506
```

In the above output (and elsewhere in this tutorial), informative dimension labels have been added to ease readability. This output shows, for example, that the probabilities of responding in categories 0, 1, 2, and 3 to item 6 for a person with $\theta = -1$ equal 0.4118, 0.3800, 0.1849, and 0.0233. Notice that the output also includes some `NA` values. This is because items 1, 2, and 3 have only 2 response categories, and therefore these subjects can only respond in categories 0 and 1 and categories 2 and 3 are `NA`.

For dichotomous items, it is common to only find the probability of responding in the higher response category (because the sum of response category probabilities must sum to 1 for each θ, probabilities for category 0 are 1 minus those for category 1). In `flexmet`, specific category response probabilities can be found by adding the `returncat` argument. This is illustrated below for item 1:

```
irf_fmp(theta = theta, bmat = bmat[1, ], maxncat = 4, returncat = 1)
```

```
##               item 1
## theta = -1 0.3113050
## theta = 1  0.8549576
```

Notice that the `maxncat` argument is set equal to 4 in this example, even though the item is dichotomous. This is because the item parameters are taken from the first row of `bmat`, and `bmat` includes columns corresponding to four-category items. In other words, `maxncat` should be set to one greater than than the number of `bmat` columns (or one greater than the number of `bmat` entries, if `bmat` is a vector) that correspond to ξ parameters, even if no item with `maxncat` categories is included in the call to `bmat`. Note that the `returncat` argument represents the *value* of the category to output, where categories are labeled starting at 0. Therefore, returning category "1" for dichotomous item 1 returns the probability of a positive item response to item 1. By default, if `maxncat = 2`, only probabilities for category "1" are returned, and if `maxncat > 2`, all response category probabilities are returned.

Calls to `irf_fmp` also can be used to plot IRFs. Figure 1 displays the IRFs for the six example items, and tge code to produce a version of this figure is included in Appendix C. In this figure, response probabilities for category 1 (a positive response) are shown for the 3 dichotomous items, and all response category probabilities are shown for the 3 polytomous items.

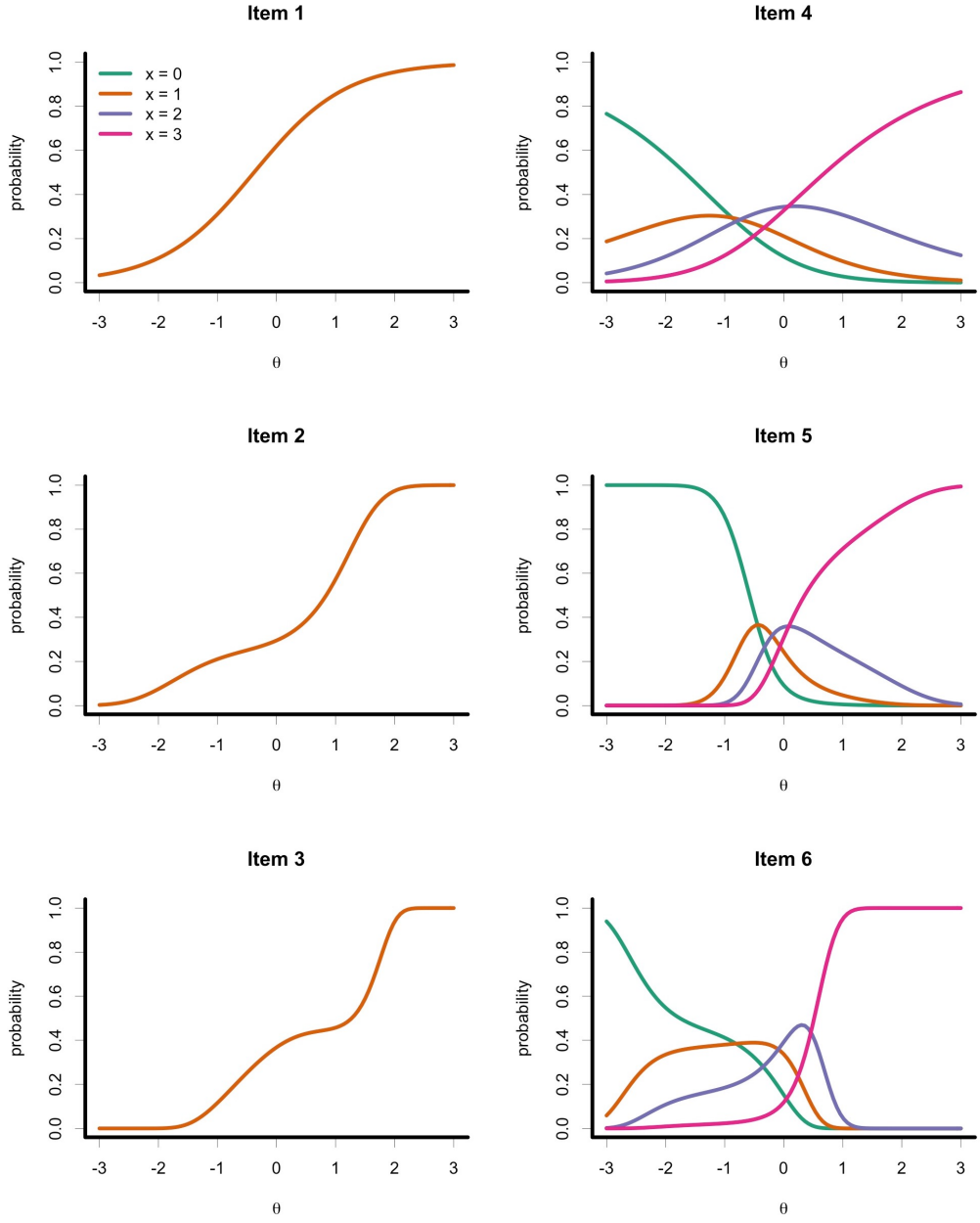

Figure 1. Item response functions for six example FMP items. Items 1, 2, and 3 are dichotomous, and items 4, 5, and 6 have four response categories. Items 1 and 4 have $k_i = 0$, items 2 and 5 have $k_i = 1$, and items 3 and 6 have $k_i = 2$.

3. Fitting the FMP Model in `flexmet`

The `flexmet` package includes functionality for specifying, fitting, and manipulating the general FMP model. Notably, `flexmet` is not the only `R` package available for fitting the FMP model. For example, the `mirt` package [15] includes functionality to estimate FMP item parameters using marginal maximum likelihood estimation. However, if any items have $k = 0$, the `mirt` package currently (version 1.34 of `mirt` is current at the time of writing) requires the user to estimate either 2PL or GPCM parameters rather than FMP parameters. Because `mirt` parameterizes the 2PL and GPCM differently than the Greek-letter FMP parameterization, this may make the `mirt` package less than ideal for tests with a variety of k_i values and for advanced applications of the FMP model (such as scale transformations, as described in a later section). Therefore, the `flexmet` package includes several ways to estimate FMP model parameters, including built-in methods for fixed-effects and random-effects estimation, as well as a wrapper that uses the `mirt`

package to estimate parameters and return parameter estimates in a standardized format. As illustrated below, `flexmet` facilitates item parameter estimation for items with any combination of k_i values and numbers of response categories, with or without the use of Bayesian priors.

Early applications of the FMP model treated the latent trait as a fixed effect to estimate item parameters [3,11]. In this fixed-effects approach, initial estimates of the θ parameters are treated as known quantities when calculating maximum likelihood estimates of the Greek-letter parameters for an individual item. These fixed θ values are called θ surrogates [3,11] and are calculated as the first principal components scores of the full data matrix. In the following code chunk, 1000 true θ values are simulated from a standard normal distribution. Then, the `sim_data` function in `flexmet` is used in conjunction with the six-item `bmat` matrix defined earlier to randomly generate item response data. Finally, θ surrogates are calculated by passing the simulated data to `flexmet`'s `get_surrogates` function.

```
set.seed(234)
theta <- rnorm(1000)
dat <- sim_data(bmat = bmat, theta = theta, maxncat = 4)
tsur <- get_surrogates(dat)
```

The `tsur` object now includes 1000 θ surrogate values calculated from the simulated data. To estimate item parameters for a single item using fixed-effects estimation with θ surrogates, we can use the `fmp_1` function in `flexmet`. As illustrated below, this function requires the user to specify the data vector, the desired `k` value, and a vector of θ surrogates `tsur`. Below, this is illustrated for k values of 0, 1, and 2.

```
i <- 2 # choose an item
fe0i <- fmp_1(dat[, i], k = 0, tsur = tsur)
fe1i <- fmp_1(dat[, i], k = 1, tsur = tsur)
fe2i <- fmp_1(dat[, i], k = 2, tsur = tsur)
```

The best choice of k for a given item is typically unknown, and so authors have suggested comparing the item-level Akaike Information Criteria (AIC) value to select the optimal k value [3,12]. We can perform this comparison by extracting the `AIC` list element from each call to `fmp_1`.

```
c(fe0i$AIC, fe1i$AIC, fe2i$AIC)
```

```
## [1] 1167.464 1157.760 1159.476
```

In this example, `k = 1` leads to the lowest AIC value. Incidentally, $k = 1$ is also the data-generating k value for this item (item 2), though this will not always be the case. Note that it is not always desirable to seek the "correct" k_i value, but instead it may be preferable to approximate the population curve as closely as possible without overfitting the data. One measure of the similarity of item response functions is the root integrated mean squared error (RIMSE; [6,12]), which is defined here as

$$\text{RIMSE} = \sqrt{\int \left(\sum_{c=0}^{C_i-1} xP_1(X_i = c|\theta) - \sum_{c=0}^{C_i-1} xP_2(X_i = c|\theta) \right)^2 g(\theta) d\theta} \quad (6)$$

where P_1 and P_2 represent the two item response functions to compare (not necessarily from the FMP model), and $g(\theta)$ indicates a θ distribution to integrate over. Smaller values of RIMSE indicate greater similarity between the two curves. In `flexmet`, the `rimse` function can calculate the RIMSE for any combination of $\boldsymbol{b}$-vectors that represent the same number of response categories (though not illustrated here, `rimse` can also be used with non-FMP item

response functions). In `flexmet`, $g(\theta)$ is standard normal by default but can be modified using the `int` argument, which expects a matrix with two columns. The first column should include a sequence of quadrature points, and the second column should include the densities of each quadrature point, scaled so that the densities sum to 1. The `int_mat` function in `flexmet` facilitates the creation of this matrix. The `int_mat` function takes a distribution function `distr` (such as `dnorm` or `dunif`), a named list `args` of the parameters of that distribution, the lower and upper bounds of the quadrature points, `lb` and `ub`, and the number of quadrature points, `npts`. An example of modifying these arguments of the `int_mat` function is included later in this paper (Section 4.2).

The first two arguments to `rimse` should be two vectors of $\boldsymbol{b}$-parameters, in either order. In the code below, the true $\boldsymbol{b}$-parameters for item 2 are listed first. Note that because columns 2 and 3 of `bmat` represent category intercept parameters for polytomous items (and include `NA` values for item 2), we omit these from the $\boldsymbol{b}$-vector when calling `rimse`. The estimated $\boldsymbol{b}$-parameters are listed second and are found in the `bmat` list element of each call to `fmp_1`. If items are polytomous, the `ncat` argument should also be specified to indicate the number of response categories. Because the default value of `ncat = 2`, it is not necessary to include this argument for dichotomous items.

```
c(rimse(bmat[i, -c(2, 3)], fe0i$bmat),
  rimse(bmat[i, -c(2, 3)], fe1i$bmat),
  rimse(bmat[i, -c(2, 3)], fe2i$bmat))
```

```
## [1] 0.06720749 0.04754961 0.05220503
```

Comparing the RIMSE of the estimated curves versus the data-generating curves for 3 values of `k`, we see that `k = 0` leads to the highest error in estimation, followed by `k = 2`, and `k = 1` most closely traces the population item response function.

It is also possible to estimate fixed-effects item parameters for multiple items in one command using the `fmp` function with the `em = FALSE` argument. This method will automatically calculate θ surrogates based on the provided data matrix. In the `fmp` function, it is possible to specify different k values for different items. Namely, if a scalar is specified for the `k` argument, the same k value will be used for all items. Otherwise, a vector of k values should be specified, one per item.

In the example below, the fixed-effects FMP model is fit several times. First, all items are fit with $k = 0$, $k = 1$, and $k = 2$. Based on these results, a model with differing k values is specified based on the optimal k value, as indicated by comparing item-level AIC values.

```
fe0 <- fmp(dat, k = 0, em = FALSE)
fe1 <- fmp(dat, k = 1, em = FALSE)
fe2 <- fmp(dat, k = 2, em = FALSE)
rbind(fe0$mod$AICs, fe1$mod$AICs, fe2$mod$AICs)
```

```
##              1        2        3        4        5        6
## k = 0 1127.219 1167.464 1129.203 2135.735 1241.019 1783.829
## k = 1 1130.652 1168.813 1131.627 2137.054 1241.437 1710.060
## k = 2 1131.957 1172.118 1126.248 2135.115 1245.171 1764.221
```

The user may notice that the model with `k = 2` produces an error that the (item parameter) information matrix cannot be inverted. This is a common problem with high k values. If this error occurs, standard errors are not available for the estimated item parameters; however, the item parameter estimates themselves may be used without concern.

Based on the above results, we see that the lowest AIC value is observed for $k = 0$, 0, 2, 2, 0, and 1 for the 6 items. We may then choose to fit a new fixed-effects FMP model with these varying k values, as illustrated below.

```
fe_mixed <- fmp(dat, k = c(0, 0, 2, 2, 0, 1), em = FALSE)
fe_mixed$bmat
```

```
##         b0_1   b0_2    b0_3     b1      b2     b3      b4     b5
## [1,]  0.3892     NA      NA 1.2077  0.0000 0.0000  0.0000 0.0000
## [2,] -0.6355     NA      NA 0.9249  0.0000 0.0000  0.0000 0.0000
## [3,] -0.7711     NA      NA 0.8339 -0.0075 0.0866 -0.0006 0.0040
## [4,]  1.1438 0.5142 -0.3266 1.0889  0.0757 0.0270  0.0013 0.0003
## [5,]  1.9865 0.7333 -1.6325 4.0021  0.0000 0.0000  0.0000 0.0000
## [6,]  1.3672 0.2662 -1.2763 1.4312  1.1918 0.9096  0.0000 0.0000
```

In the item parameter matrix printed above, notice how items with different k values and numbers of response categories are represented. Specifically, `NA`'s are used as placeholders for items 1–3 that include less than the maximum number of response categories. In contrast, if k_i is less than the maximum k_i value represented in the parameter matrix (as is the case for items 1, 2, 5, and 6), then the higher-order $\boldsymbol{b}$ parameters are set to 0.

In addition to fixed-effects estimation, `flexmet` also provides functionality for random-effects estimation using marginal maximum likelihood estimation via the expectation-maximization (EM) algorithm [20]. The `flexmet` package includes both an inbuilt algorithm for estimating item parameters (using the `fmp` function with option `em = TRUE`) and a wrapper to the `mirt` [15] package (using the `fmp` function with option `em = "mirt"`). The `mirt` algorithm is currently faster and more reliable than the inbuilt algorithm, and so this option is used for illustration. Note that the `mirt` package must be installed in order to use this option. Using the same pattern of k values found with the fixed-effects model, we find the following results:

```
re_mixed <- fmp(dat, k = c(0, 0, 2, 2, 0, 1), em = "mirt")
re_mixed$bmat
```

```
##         b0_1   b0_2    b0_3     b1      b2     b3      b4     b5
## [1,]  0.3974     NA      NA 1.2866  0.0000 0.0000  0.0000 0.0000
## [2,] -0.6441     NA      NA 0.9516  0.0000 0.0000  0.0000 0.0000
## [3,] -0.4041     NA      NA 0.3588 -0.5703 0.7080 -0.4838 0.2066
## [4,]  0.9198 0.3836 -0.0562 0.4188 -0.0197 0.1213 -0.0043 0.0157
## [5,]  0.4667 0.4789 -0.4709 1.7170  0.0000 0.0000  0.0000 0.0000
## [6,]  0.7189 0.1984 -0.9902 1.4832  0.8613 0.1684  0.0000 0.0000
```

At a glance, comparing the `bmat` outputs of `fe_mixed` and `re_mixed` indicates that while some parameter estimates are similar across the two estimation methods, others are quite different. Notably, for highly parameterized models such as the FMP model, very different sets of parameters can trace similar curves. For this reason, it is useful to plot curves or to apply overall summary measures such as the RIMSE when comparing estimated curves to the true curve or to each other. These RIMSE values are calculated for `fe_mixed` and `re_mixed` after illustrating one final estimation method.

Another useful estimation option available for both the `fmp_1` and `fmp` functions is the use of Bayesian priors. Particularly, for higher values of k, the FMP model may become computationally unstable, which can be somewhat alleviated through the use of priors [12]. Because the Greek-letter parameterization of the FMP is used for parameter estimation, priors are placed on the Greek-letter parameters rather than on the more readily interpretable $\boldsymbol{b}$ parameters. When choosing priors, it may be helpful to note that $(\xi_{1i}, \ldots \xi_{(C_i-1)i}) = (b_{0_1 i}, \ldots b_{0_{C_i-1} i})$ and that $\omega = \ln b_1$. In addition, higher-order $\boldsymbol{b}$ parameters will equal zero if the corresponding $\alpha = 0$ *and* $\tau = -\infty$. Prior predictive simulation may help the user to select appropriate priors. For example, the following

code produces a prior predictive simulation of 20 item response curves generated from the following distributions: $\xi \sim N(0,1)$, $\omega \sim N(-0.5,1)$, $\alpha \sim N(0,0.5)$, and $\tau \sim N(-3,0.5)$. The plot produced by this code is shown in Figure 2. This choice of priors appears to allow for a variety of shapes and locations of the FMP curves, and so we continue to illustrate Bayesian estimation with these priors.

```
par(mfrow = c(1, 1)) # display only one figure in the plotting window
set.seed(234)
k <- rep(c(1, 2), each = 10)
pp_bmat <- sim_bmat(n_items = 20, k = k, ncat = 2,
                    xi_dist = list(rnorm, mean = 0, sd = 1),
                    omega_dist = list(rnorm, mean = -0.5, sd = 1),
                    alpha_dist = list(rnorm, mean = 0, sd = 0.5),
                    tau_dist = list(rnorm, mean = -3, sd = 0.5))$bmat
curve(irf_fmp(x, pp_bmat[1, ]), xlim = c(-3, 3), ylim = c(0, 1),
      xlab = expression(theta), ylab = "probability", col = k[1])
for(i in 2:20)
  curve(irf_fmp(x, pp_bmat[i, ]), add = TRUE, col = k[i])
```

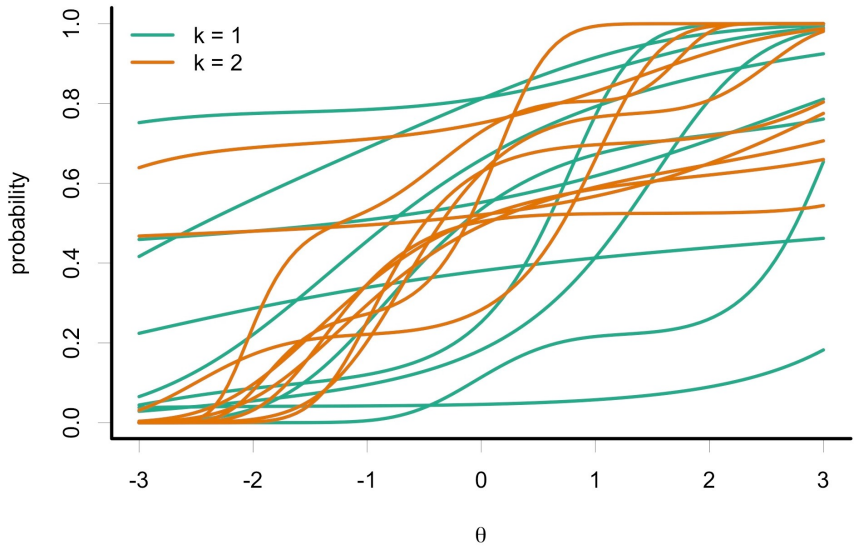

Figure 2. Prior predictive simulation of FMP item response curves with $k = 1$ and $k = 2$ and $\xi \sim N(0,1)$, $\omega \sim N(-0.5,1)$, $\alpha \sim N(0,0.5)$, and $\tau \sim N(-3,0.5)$.

Priors can be added to models fit using `flexmet` by specifying the `prior` argument to the `fmp` or `fmp_1` function. As of version 1.1 of `flexmet`, only normally distributed priors are available, and the same priors are applied to all model parameters of a given class (i.e., for all items in the data set and for all instances of that parameter type within an item). To specify priors, a list with named elements `xi`, `omega`, `alpha`, and `tau` should be passed to the `prior` argument. For example, to specify a standard normal prior for all ξs, we may write `prior = list(xi = c("norm", 0, 1))`. The code below illustrates Bayesian estimation with the `em = "mirt"` option.

```
re_priors <- fmp(dat, k = c(0, 1, 2, 0, 1, 2), em = "mirt",
                 prior = list(xi = c("norm", 0, 1),
                              omega = c("norm", -0.5, 1),
                              alpha = c("norm", 0, 0.5),
                              tau = c("norm",  -3, 0.5)))
```

Finally, we may compare the accuracy of each of the three illustrated estimation methods using the `rimse` function. The code below does this for each of the six simulated items. For each item, random-effects estimation tends to lead to somewhat more accurate curves than fixed-effects estimation, and the use of priors improves parameter estimation accuracy in some, but not all, cases.

```
rimse_res <- sapply(1:6, function(i){
  c(rimse(fe_mixed$bmat[i, ], bmat[i, ], ncat = 4),
    rimse(re_mixed$bmat[i, ], bmat[i, ], ncat = 4),
    rimse(re_priors$bmat[i, ], bmat[i, ], ncat = 4))
})
```

```
##           Item 1 Item 2 Item 3 Item 4 Item 5 Item 6
## fe_mixed  0.0194 0.0672 0.0520 0.1454 0.1446 0.1944
## re_mixed  0.0181 0.0659 0.0298 0.1281 0.1149 0.1093
## re_priors 0.0180 0.0378 0.0595 0.0285 0.0254 0.1147
```

Several other estimation options are available with the `fmp` and `fmp_1` functions that are not illustrated here. A vector of starting values may be specified with the `start_vals` argument, where parameters should be listed in the same order as the estimated parameters given in the `parmat` list element of the fitted model object. For fixed-effects estimation, the `method` argument indicates the optimization algorithm (passed to the `optim` function in `R`'s `stats` package). For random-effects estimation, the `max_em` option indicates the maximum number of EM cycles (default 500), and the `n_quad` function indicates the number of quadrature points (default 49). Additional named arguments may be passed to the `optim` function (if `em = TRUE`) or the `mirt` function (if `em = "mirt"`).

Finally, `flexmet` includes the `th_est_ml` and `th_est_eap` functions for maximum likelihood (ML) and *expected a posteriori* (EAP, [21]) person parameter estimation. Both of these functions require a data matrix `dat` and a matrix of $\boldsymbol{b}$-parameters `bmat`. If at least one item is polytomous, the `maxncat` argument should also be specified. By default, the `th_est_eap` function uses a standard normal prior with 33 quadrature points, and this may be modified using the `int` argument in conjunction with `int_mat`. A call to either trait estimation function will result in a matrix with two columns: one for the θ parameter estimates and one for the standard errors (for ML) or posterior standard deviations (for EAP). The code below illustrates this procedure for the example data and the `re_priors` estimated item parameters.

```
mle_ests <- th_est_ml(dat = dat, bmat = re_priors$bmat, maxncat = 4)
eap_ests <- th_est_eap(dat = dat, bmat = re_priors$bmat, maxncat = 4)
head(cbind(mle_ests, eap_ests))
```

```
##       ML_est ML_sem EAP_est EAP_psd
## [1,] -0.1834 0.3758 -0.1162  0.3628
## [2,] -1.9319 1.1834 -1.2733  0.5576
## [3,] -1.0507 0.5395 -0.9304  0.4786
## [4,]  0.9762 0.4800  0.8133  0.4591
## [5,]  1.5290 0.7820  1.1532  0.5422
## [6,]  0.6364 0.4342  0.5429  0.3581
```

4. Transforming the FMP Model with `flexmet`

Aside from the increased flexibility in IRF shapes afforded by the FMP model, another potential use of the FMP model is to transform item response models linearly or nonlinearly [18,19]. Specifically, suppose that two scalings of the latent trait, θ and $\theta^\star$, are related by a monotonic polynomial function of degree $2k_\theta + 1$ such that

$$\theta = t_0 + t_1\theta^\star + t_2\theta^{\star 2} + \cdots t_{2k_\theta+1}\theta^{\star 2k_\theta+1}, \quad (7)$$

where k_θ is a non-negative integer. Then, an FMP model defined on the scale of θ may be transformed to the scale of $\theta^\star$ using matrix operations described elsewhere [18,19]. For a given k_θ and an item with a given k_i (i.e., item complexity defined on the scale of θ), the item complexity on the scale of $\theta^\star$, $k_i^\star$, equals

$$k_i^\star = 2k_ik_\theta + k_i + k_\theta. \quad (8)$$

Therefore, item response functions on the scale of $\theta^\star$ will necessarily have larger item complexities than response functions on the scale of θ.

Two applications of nonlinearly transforming the FMP model will be illustrated below. In the first application, item parameters are estimated separately for two groups with different latent trait distributions. Because the latent trait is often assumed to follow a standard normal distribution for model identification, the fitted item response models will be on different, nonlinearly related, scales. Linear and nonlinear linking transformations will be estimated and compared. In the second application, an item response model will be transformed such that the latent trait scale is on a more interpretable percentage-correct score metric.

4.1. Item Parameter Linking with the FMP Model

If item parameters for the same items are estimated from two different samples, item parameter linking may be used to put the parameters on the same scale. The need for item parameter linking arises from differences in model identification constraints [22]. For the most common parametric item response models, such as the 2PL and GPCM, the chosen functional form of the item response model identifies the scale of the latent trait up to a linear transformation. In addition, models may be identified by assuming that the latent trait is standardized, or that the latent trait follows a standard normal distribution (as is the case for the fixed-effects FMP estimation illustrated above). Therefore, if there are any differences in the location, variance, or shape of the latent trait distribution between groups, linking may be necessary to put the two sets of item parameters on more similar scales. To illustrate how item parameters on different scales may be linked with the FMP model, an illustrative simulation is presented next.

To simulate data, a population set of FMP item parameters was first generated for 20 items each with two response categories and $k = 0$ using the `sim_bmat` function in `flexmet`. With the `sim_bmat` function, the user can generate items with varying k values and numbers of response categories by specifying vectors for the `k` and `ncat` functions. The population item parameters were then used to generate data for two groups of 5000 examinees. The population latent trait values for the first group were first drawn from a standard normal distribution, then transformed by taking -2 plus e to the power of the standard normal scores divided by 2. The population latent trait values for the second group were drawn from a standard normal distribution. From this design, because the second group's θ values were transformed to the scale of the first group's scores (which were identified by a combination of the shape of the fitted model and the assumption of a standard normal distribution), the population relationship between the two scales equals

$$\theta = -2 + \exp(\theta^\star/2). \quad (9)$$

```
set.seed(123)
bmat <- sim_bmat(n_items = 20, k = 0, ncat = 2)$bmat
theta1 <- exp(rnorm(5000) / 2) - 2
theta2 <- rnorm(5000)
```

After generating population θ values, data were then generated separately for each group. To select the k values that best represent these data, each item for each group was first fit to a series of FMP models using the `fmp_1` function based on theta surrogates. Specifically, each item was first fit with $k = 0$ and $k = 1$. If the AIC for $k = 0$ was smaller than for $k = 1$, then the final model was fit with $k = 0$. If not, models with sequentially higher k values were fit until the model with the smaller k value had a smaller AIC than the model with the higher k value. The code to implement this procedure is included in Appendix D and resulted in a mixture of $k = 0$ and $k = 1$ for both groups. Once the empirically selected k values for each item and group were chosen, item parameters were estimated again with `fmp` and the `em = "mirt"` option. For group 1, the argument `technical = list(NCYCLES = 1500)` was passed to `mirt` because more than 500 cycles of the EM algorithm (the default in `mirt`) were required for this model to converge.

```
data1 <- sim_data(bmat = bmat, theta = theta1)
data2 <- sim_data(bmat = bmat, theta = theta2)

k1 <- c(0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1)
k2 <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1)

d1_2pl <- fmp(data1, k = k1, em = "mirt",
              technical = list(NCYCLES = 1500))
d2_2pl <- fmp(data2, k = k2, em = "mirt")
```

The `flexmet` package includes two functions for linking item parameters: `hb_link` for linking based on the Haebara method (HB) [23] and `sl_link` for linking based on the Stocking–Lord method (SL) [24], and both functions take the same arguments. The `bmat1` and `bmat2` arguments give the matrices of item parameters in the polynomial coefficient parameterization. These functions currently assume that both matrices include the same set of common items in the same order. The `bmat2` parameters are on the $\theta^\star$ metric and are transformed to the θ metric of the `bmat1` parameters. The polynomial degree of the estimated transformation is specified by the `k_theta` argument (monotonicity of the estimated linking transformation is ensured using the same strategy as is used for item parameters).

```
hb0 <- hb_link(bmat1 = d1_2pl$bmat, bmat2 = d2_2pl$bmat, k_theta = 0)
hb1 <- hb_link(bmat1 = d1_2pl$bmat, bmat2 = d2_2pl$bmat, k_theta = 1)
hb2 <- hb_link(bmat1 = d1_2pl$bmat, bmat2 = d2_2pl$bmat, k_theta = 2)

sl0 <- sl_link(bmat1 = d1_2pl$bmat, bmat2 = d2_2pl$bmat, k_theta = 0)
sl1 <- sl_link(bmat1 = d1_2pl$bmat, bmat2 = d2_2pl$bmat, k_theta = 1)
sl2 <- sl_link(bmat1 = d1_2pl$bmat, bmat2 = d2_2pl$bmat, k_theta = 2)

c(hb0$value,hb1$value, hb2$value)


## [1] 0.5743 0.4077 1.6079


c(sl0$value, sl1$value, sl2$value)


## [1] 0.1845 0.1203 0.3127
```

From the objects returned by calls to `hb_link` and `sl_link`, the `value` list element gives the function value from minimizing the HB or SL criterion (see [23,24] for details). Comparing these values resulting from the HB or SL method with different k_θ can help

inform which k_θ yields the most appropriate latent trait transformations, where smaller values indicate smaller differences between the estimated item/test response functions on the two scales. Although criteria values from calls to `hb_link` should not be compared to values from calls to `sl_link`, comparing values with different k_θ suggests that $k_\theta = 1$ leads to the closest match between the two sets of item parameters under both the SL and HB methods.

Calls to `hb_link` and `sl_link` also output the estimated vector of polynomial coefficients in the `tvec` list element. As shown below, although these values are similar for the HB and SL methods, there are some differences, particularly for higher-order polynomial coefficients.

```
rbind(hb0$tvec, sl0$tvec)
```

```
##         t0     t1
## HB -0.9472 0.6339
## SL -0.9277 0.6137
```

```
rbind(hb1$tvec, sl1$tvec)
```

```
##         t0     t1     t2     t3
## HB -0.9925 0.4322 0.0732 0.0803
## SL -0.9970 0.5154 0.1189 0.0091
```

```
rbind(hb2$tvec, sl2$tvec)
```

```
##         t0     t1     t2     t3      t4     t5
## HB -0.8978 0.0000 0.0000 0.0002 -0.0073 0.0904
## SL -0.8871 0.0011 0.0017 0.2405  0.0004 0.0003
```

The `tvec` output is useful for understanding the estimated relationship between the θ and $\theta^\star$ scales. Recall from Equation (7) that θ is a polynomial function of $\theta^\star$, and Equation (9) gives the population relationship between θ and $\theta^\star$. In Figure 3, the SL linking transformations are compared to the true linking transformation. The code to reproduce a version of this figure is included in Appendix E and makes use of `flexmet`'s `inv_poly` function to calculate the θ value that corresponds to any particular value of $\theta^\star$. Although in non-simulation applications, the true latent trait transformation will not be known, plotting may still be a useful tool to inspect the similarity of different transformations.

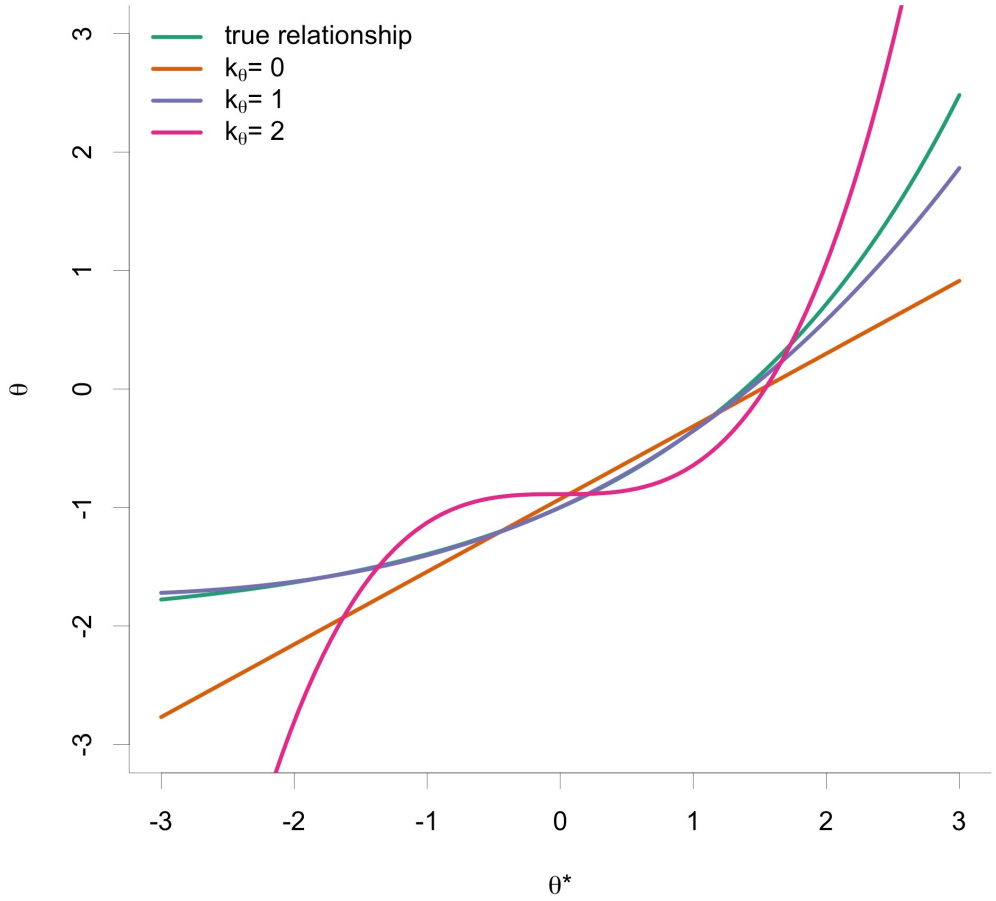

Figure 3. True linking transformation and estimated Stocking–Lord linking transformations with various k_θ values.

To illustrate the meaning of the linking transformation, consider a θ value of 0.5. In the code below, this θ value is transformed to a $\theta^\star$ value using the `inv_poly` function, which takes a scalar value (here, each θ) and returns the $\theta^\star$ value that combines with the polynomial coefficient vector `coefs` (see Equation (7)) to produce θ. As shown below for the SL transformation with $k_\theta = 1$, the item response probabilities for `theta` combined with the originally estimated `bmat` are identical to the transformed `thetastar` values combined with the transformed item parameters (which are found in the `bmat` list element of the linking output). This equality is demonstrated for the first three items of the simulated test.

```
theta <- 0.5
irf_fmp(theta, d2_2pl$bmat[1:3, ])
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.6613631 0.7826067 0.6244098
```

```
thetastar <- inv_poly(theta, coefs = sl1$tvec)
irf_fmp(thetastar, sl1$bmat[1:3, ])
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.6613657 0.7826081 0.6244117
```

4.2. Transforming the FMP Model to a User-Defined Scale

Another potential application of FMP model transformations is to specify an item response model on a latent trait metric other than θ. In the following illustration, `flexmet` functions are used to transform a set of 3 parameter logistic (3PL) model item parameters from the typical θ metric to a percentage-correct (PC) metric where the lowest score is 0, the highest score is 100, and scores roughly correspond to the percentage of items answered correctly. Although this type of model transformation is illustrated using a PC metric, any monotonic transformation is possible, so long as the user can identify a suitable polynomial approximation to the desired transformation. Transforming the entire item response model to a more interpretable metric allows for trait estimates, item information, standard errors, and other quantities to be calculated directly on the reported score metric.

The item parameters used for the following illustration are taken from the `tcals` data set found in version 3.16 of the `catR` package [25] (and also included in Appendix F). Note that these parameters correspond to the following representation of the 3PL model:

$$P(X_{ij} = 1|\theta_j, a_i^\star, b_i^\star, c_i^\star) = c_i^\star + \frac{1 - c_i^\star}{1 + \exp(-a_i^\star(\theta_j - b_i^\star))}, \tag{10}$$

where $a_i^\star$ indicates the item discrimination parameter, $b_i^\star$ indicates the item difficulty parameter, and $c_i^\star$ indicates the guessing, or lower asymptote, parameter. The star notation is used for these item parameters only to avoid confusion with previously defined notation.

Note that the 3PL presented in Equation (10) is *not* a special case of the FMP model as presented in Equation (1). However, for dichotomous item responses, an FMP model may be specified that includes upper ($d_i^\star$) and/or lower ($c_i^\star$) asymptotes on the item response functions [26] as follows:

$$P(X_{ij} = 1|\theta_j \boldsymbol{b}_i, c_i^\star, d_i^\star) = c_i^\star + \frac{d_i^\star - c_i^\star}{1 + \exp(-b_0 - m^\star(\theta_j, b_{1i}, \ldots, b_{2k_i+1,i}))}. \tag{11}$$

If not otherwise specified, we may assume that all $d_i^\star = 1$ and all $c_i^\star = 0$. In many `flexmet` functions, upper and lower asymptote parameters may be specified using the `cvec` and `dvec` arguments (note that `flexmet` currently does not allow these asymptote parameters to be estimated with the `fmp` or `fmp_1` functions; however, most other `flexmet` functions accommodate asymptote parameters). These asymptote parameters are unaffected by transformations of the latent trait metric.

In the code below, the `tcals` parameters are read in first. Then, an object `fmp_pars` is defined that transforms the 3PL $a_i^\star$ and $b_i^\star$ parameters to the polynomial coefficients required by the FMP model. Then, a sequence of θ values is generated between -4 and 4, and item response probabilities are calculated for each item and θ value. The expected percentage-correct (PC) scores are then calculated by summing the item response probabilities for each θ value and multiplying them by $100/85$ (where 85 is the number of test items). We then regress the θ values on the PC scores using monotonic polynomial regression [27] as implemented in version 0.3–10 of the `Monopoly` package [28] for `R`. Several regressions are fit using k values (which are used as k_θ values) from 0 to 5, where, as before, $2k + 1$ indicates the maximum polynomial degree. Finally, the average regression residual is printed for each k value.

```
data(tcals, package = "catR")

fmp_pars <- cbind(-tcals$a * tcals$b, tcals$a)

theta <- seq(-4, 4, length = 5000)
probs <- irf_fmp(theta = theta, b = fmp_pars, cvec = tcals$c)
PC <- rowSums(probs) * 100 / 85

mp_res <- lapply(0:5, function(k){
```

```
  MonoPoly::monpol(theta ~ PC, K = k, weights = dnorm(theta))
})

sapply(mp_res, function(x) mean(x$residuals))
```

```
## [1] 0.12065489 0.09256206 0.06798434 0.05051354 0.03738312 0.02756218
```

The average regression residuals indicate that higher k values lead to better approximations of the desired curve. However, choosing a higher k value will lead to many higher-order coefficients for the transformed item parameters, so we may choose to use the smallest k value that yields an acceptable level of accuracy. For instance, if we decide that an average regression residual of less than 0.1 is acceptable, then we may choose to move forward with the $k = 1$ transformation. The estimated regression coefficients then become the polynomial coefficients with which to transform `fmp_pars` (i.e., the t coefficients for Equation (7) with $k_\theta = 1$). We can implement this transformation by applying the `transform_b` function in `flexmet` to each row of `fmp_pars`, specifying the θ transformation coefficients in the `tvec` argument, as illustrated below.

```
tvec <- coef(mp_res[[2]])
transformed_bmat <- t(apply(fmp_pars, 1, transform_b, tvec = tvec))

head(transformed_bmat)
```

```
##            b0     b1      b2     b3
## [1,] -19.1331 0.9053 -0.0136 0.0001
## [2,]  -9.4778 0.4777 -0.0072 0.0000
## [3,] -16.9269 0.8561 -0.0128 0.0001
## [4,] -24.7629 1.0949 -0.0164 0.0001
## [5,] -10.7440 0.5110 -0.0077 0.0000
## [6,] -12.9219 0.5887 -0.0088 0.0000
```

Note again that the $c_i^\star$ parameters remain unchanged during FMP parameter transformations, and so the argument `cvec = tcals$c` should be passed to any `flexmet` function that makes use of these parameters. For example, we may compare two methods of calculating trait estimates on the transformed $\theta^\star$/PC metric: first, estimating person parameters on the θ metric and then applying the polynomial transformation; and estimating person parameters directly on the $\theta^\star$/PC metric. In this example, we use a standard normal prior to estimate EAPs on the θ metric, and a $N(80, 10)$ prior to estimate EAPs directly on the $\theta^\star$/PC metric. This prior is specified by feeding the output of a call to the `int_mat` function to the `int` argument. In the example code shown below, both priors are specified with 89 quadrature points, the prior on θ is bounded between 0 and 4, and the prior on $\theta^\star$ is bounded between 0 and 100. Finally, the EAPs that were estimated on the θ metric are transformed to the $\theta^\star$/PC metric using the `inv_poly` function.

```
set.seed(987)
dat <- sim_data(bmat = fmp_pars, theta = rep(0, 1000),
                maxncat = 2, cvec = tcals$c)
theta_ests <- th_est_eap(dat = dat, bmat = fmp_pars,
                         maxncat = 2, cvec = tcals$c,
                         int = int_mat(dnorm, list(mean = 0, sd = 1),
                                       lb = -4, ub = 4, npts = 89))
tstar_ests <- th_est_eap(dat = dat, bmat = transformed_bmat,
                         maxncat = 2, cvec = tcals$c,
                         int = int_mat(dnorm, list(mean = 80, sd = 10),
                                       lb = 0, ub = 100, npts = 89))

theta_ests_transformed <- sapply(theta_ests[, 1], inv_poly, coefs = tvec)

summary(tstar_ests[, 1])


##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   76.83   82.11   83.71   83.70   85.20   92.80


summary(theta_ests_transformed)


##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   77.59   82.90   84.45   84.44   85.91   93.36
```

Comparing the distribution of trait estimates calculated with both approaches, we see that the scores are relatively similar (the small difference in estimated distributions may be due to the use of different prior distributions). However, a major advantage of estimating parameters directly on the $\theta^\star$/PC metric is that the estimated standard errors are on the appropriate metric. That is, nonlinear monotonic transformations of the latent trait metric can lead to unexpected distortions of the information matrices [19,29]. An illustration of the effects of this transformation is illustrated in Figure 4, and the code to reproduce a version of this figure is included in Appendix F. This code makes use of the `iif_fmp` function in `flexmet`, which has similar arguments to the `irf_fmp` function and yields item information for any FMP items and θ values. In the upper panels of Figure 4, three example IRFs (taken from the 85 `tcals` items) are displayed on the θ metric and after being transformed to the $\theta^\star$/PC metric. Notice that although the shape of each IRF changes, the relative relationships among the three curves do not change. In the bottom panels, the test information function (i.e., the sum of item-level information functions) is illustrated for both the θ and $\theta^\star$ metrics. Whereas test information on the θ metric is unimodal and centered just below $\theta = 0$, the $\theta^\star$ metric is bimodal with peaks around $\theta^\star = 35$ and $\theta^\star = 90$. This result implies that the items that the trait regions measured most accurately on the θ metric are not necessarily those measured most accurately on the $\theta^\star$ metric. This result also highlights the importance of calculating information directly on the metric on which scores are reported.

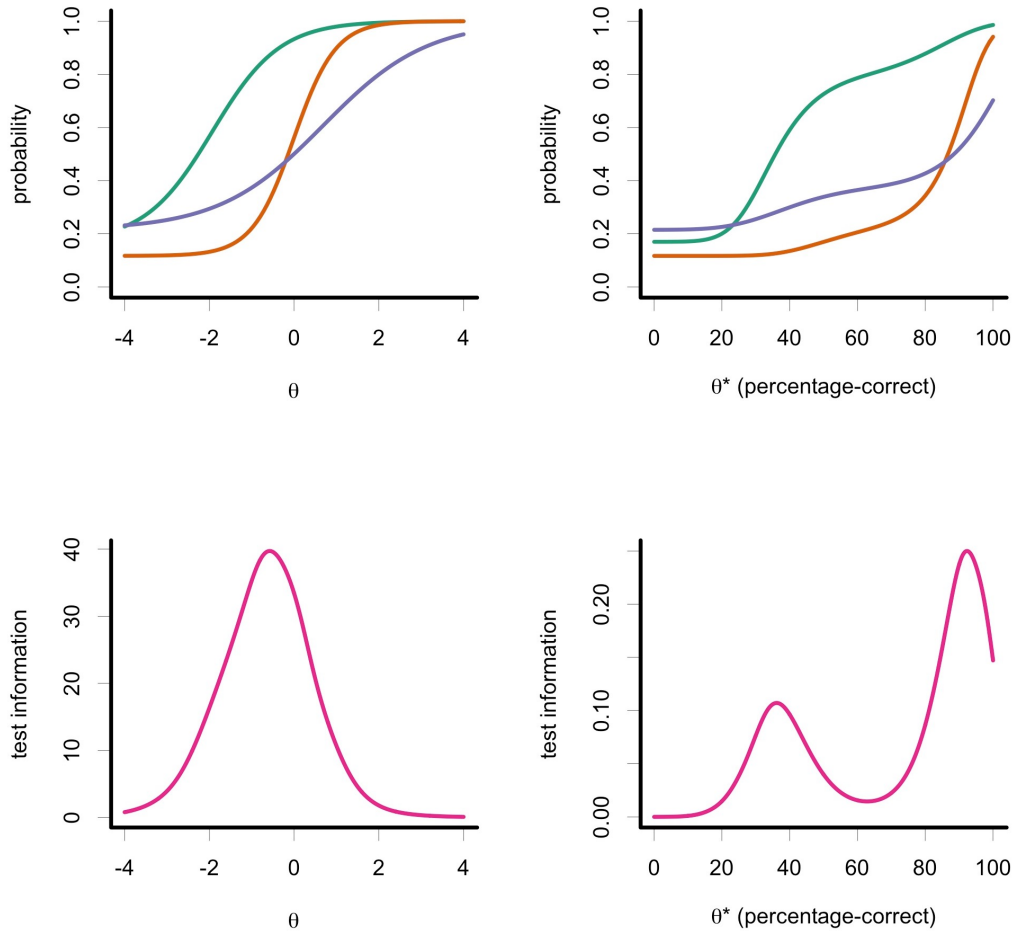

Figure 4. Item response functions (upper row) and test information functions (bottom row) on the original θ metric and on the transformed percentage-correct metric $\theta^{\star}$.

5. Summary

This paper provides an introduction to the specification and use of the FMP model through the `flexmet` package for `R`. Version 1.1 of `flexmet` includes functionality to estimate FMP item parameters for both dichotomous and polytomous items, as well as methods to transform FMP item parameters, generate FMP item parameters and data, and several other methods that are particularly useful when working with the FMP model. The provided examples also illustrate some scenarios in which it may be desirable to use the FMP model, including fitting IRFs that can take on more flexible shapes than allowed by traditional models and transforming item response models to user-defined metrics (e.g., a sum score or proportion-correct metric). Moreover, it is hoped that broader use of the `flexmet` package will lead to further development of the `flexmet` package, including planned features such as more choices of prior distributions for Bayesian estimation and the estimation of upper and lower asymptote parameters. Although not every feature of the current `flexmet` package is illustrated in the examples presented above, we hope that this tutorial provides the reader with an accessible introduction to the FMP model and the tools to use the FMP model with the `flexmet` package.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All information required to fully reproduce the results in this paper is included in text.

Acknowledgments: Many thanks to Xing Chen for his assistance testing code and suggesting improvements and to three anonymous reviewers for their suggestions in improving the readability and accessibility of this tutorial.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

2PL	Two-parameter logistic
3PL	Three-parameter logistic
AIC	Akaike information criterion
EAP	Expected a posteriori
EM	Expectation-maximization
FMP	Filtered monotonic polynomial
GPCM	Generalized partial credit model
HB	Haebara
IRF	Item response function
ML	Maximum likelihood
PC	Percentage-correct
RIMSE	Root integrated mean squared error
SL	Stocking–Lord

Appendix A. Relationship of the FMP Model ($k = 0$) to Equivalent Models in Other Software Packages

The purpose of this appendix is to illustrate how to transform 2PL and GPCM item parameters from various software packages to the item parameters used by the FMP model with $k = 0$. These transformations are useful for fully understanding the FMP model and some of its applications (e.g., see Section 4.2). Although these examples are in the context of specific `R` packages, they are broadly representative of the most common parameterizations of the 2PM and GPCM, and well as variations of these models such as the 3PM (see also Section 4.2) and the partial credit model.

In the following examples, 2PL or GPCM data are generated using the `flexmet` package (item parameter and data generation are described in Section 3). The data are then fit to the 2PL or GPCM using the `ltm` version 1.1-1 [14], `mirt` version 1.34 [15], and `TAM` version 3.7-16 [16] packages, the estimated item parameters are transformed to FMP parameters. The equivalence of the different parameterizations is then verified by comparing category response probabilities using `flexmet`'s `irf_fmp` function (introduced in Section 2) and each package's inbuilt function for calculating response probabilities. Throughout, packages are called explicitly using `R`'s `::` syntax to avoid potential package conflicts and to clarify which commands belong to which packages. Note also that this appendix uses `flexmet` functions and conventions that are more fully described within the main tutorial.

Appendix A.1. Two-Parameter Model

In Equation (1), the portion of the numerator within the $\exp()$ function when $k = 0$ and $C_i = 2$ (i.e., the conditions which make the FMP model equivalent to the 2PL) equals

$$b_{0i} + b_{1i}\theta. \tag{A1}$$

The following code demonstrates how to find the FMP b_{0i} and b_{1i} parameters from other parameterizations of the 2PL. To begin, we first generate 2PL data.

```
set.seed(987)
population_pars <- flexmet::sim_bmat(n_items = 10, k = 0, ncat = 2)$bmat
dat <- flexmet::sim_data(bmat = population_pars, theta = rnorm(1000))
```

The `ltm` package with option `IRT.param = FALSE` and the `mirt` package both parameterize the 2PL in the same way as the FMP model. This is illustrated first for `ltm`.

```
library(ltm) # must be loaded in for the ltm() function to work
ltm_fit1 <- ltm::ltm(dat ~ z1, IRT.param = FALSE)
ltm_fit1_FMP <- ltm::coef.ltm(ltm_fit1)
```

The first column of `ltm_fit1_FMP` is labeled `(Intercept)` and is equivalent to b_{0i}. The second column of `ltm_fit1_FMP` is labeled `z1` and is equivalent to b_{1i}. This is verified by comparing predicted response probabilities.

```
i <- 1 # choose an item
th <- c(-1.35, 0, 1.35) # choose theta values
ltm::plot.ltm(ltm_fit1, type = "ICC", items = i, z = th, plot = FALSE)
flexmet::irf_fmp(theta = th, bmat = ltm_fit1_FMP[i, ])
```

```
## [1] 0.294512 0.484810 0.679617
## [1] 0.294512 0.484810 0.679617
```

When fitting the 2PL in `mirt`, the same parameterization is used as for the FMP model, but `mirt` will print the b_{i1} parameters (labeled a in `mirt`) before the b_{0i} parameters (labeled d in`mirt`), so the columns must be reordered.

```
mirt_fit <- mirt::mirt(as.data.frame(dat), model = 1, itemtype = "2PL")
mirt_fit_FMP <- mirt::coef(mirt_fit, simplify = TRUE)$items
mirt_fit_FMP <- mirt_fit_FMP[, c(2, 1)] # reorder column
i <- 1 # choose an item
th <- c(-1.35, 0, 1.35) # choose theta values
mirt::probtrace(mirt::extract.item(mirt_fit, i), Theta = th)[, 2]
flexmet::irf_fmp(theta = th, bmat = mirt_fit_FMP[i, ])
```

```
## [1] 0.294519 0.484803 0.679598
## [1] 0.294519 0.484803 0.679598
```

A common way for the 2PL to be parameterized is in terms of the difficulty parameter $diff_i$ and a discrimination parameter $disc_i$. In this parameterization, the expression inside the exp() (see Equation (A1)) is

$$disc_i(\theta - diff_i). \tag{A2}$$

Therefore, the FMP $b_{01} = -diff_i disc_i$ and the FMP $b_{1i} = disc_i$. This difficulty-discrimination parameterization is used by the `ltm` package with option `IRT.param = TRUE`, as illustrated next.

```
# library(ltm) # ltm package must be loaded to make the following run
ltm_fit2 <- ltm::ltm(dat ~ z1, IRT.param = TRUE)
head(ltm::coef.ltm(ltm_fit2))
```

```
##           Dffclt   Dscrmn
## Item 1  0.100948 0.602068
## Item 2 -1.667651 0.569169
## Item 3 -0.332753 0.585347
## Item 4 -0.247654 0.723104
## Item 5 -0.981639 0.651483
## Item 6  0.417462 1.614207
```

Here, we see that `ltm` stores the difficulty parameters in the first column and the discrimination parameters in the second column. Next, we transform these parameters to b_{0i} and b_{1i} and verify that these parameters yield the same FMP response probabilities as `ltm`'s inbuilt function.

```
ltm_fit2_FMP <- cbind(-ltm::coef.ltm(ltm_fit2)[, 1] *
                        ltm::coef.ltm(ltm_fit2)[, 2],
                      ltm::coef.ltm(ltm_fit2)[, 2])
i <- 1 # choose an item
th <- c(-1.35, 0, 1.35) # choose theta values
ltm::plot.ltm(ltm_fit2, type = "ICC", items = i, z = th, plot = FALSE)
flexmet::irf_fmp(theta = th, bmat = ltm_fit2_FMP[i, ])
```

```
## [1] 0.294512 0.484810 0.679617
## [1] 0.294512 0.484810 0.679617
```

Finally, the `TAM` package uses a slightly different expression for the 2PL than the previously explored packages. Namely, the expression within the numerator exp() (see Equation (A1)) is

$$B_i\theta - \xi_i. \tag{A3}$$

```
tam_fit <- TAM::tam.mml.2pl(dat)
head(tam_fit$item) # item information, including estimated parameters
```

```
##    item    N     M  xsi.item AXsi_.Cat1 B.Cat1.Dim1
## I1   I1 1000 0.486  0.060796   0.060796    0.602032
## I2   I2 1000 0.708 -0.949163  -0.949163    0.569194
## I3   I3 1000 0.545 -0.194759  -0.194759    0.585384
## I4   I4 1000 0.540 -0.179057  -0.179057    0.723115
## I5   I5 1000 0.642 -0.639498  -0.639498    0.651467
## I6   I6 1000 0.387  0.673979   0.673979    1.614457
```

Therefore, the `TAM` 2PL model output is related to the FMP model in that $b_{0i} = -\xi_i$ and $b_{1i} = B_i$. Again, we can verify that this transformation results in the same predicted response probabilities (up to rounding error). Note that for the `TAM` package, the function that calculates response probabilities is located in the `CDM` package and appears to only be available for a certain grid of θ values. The 9th, 11th, and 13th of these θ values are used below for illustration.

```
tam_fit_FMP <- cbind(-tam_fit$item$xsi.item, tam_fit$item$B.Cat1.Dim1)
i <- 1 # choose an item
th <- attr(CDM::IRT.irfprob(tam_fit), "theta")[c(9, 11, 13)]
CDM::IRT.irfprob(tam_fit)[i , 2, c(9, 11, 13)]
flexmet::irf_fmp(theta = th, bmat = tam_fit_FMP[i, ])
```

```
## [1] 0.313624 0.484806 0.659630
## [1] 0.313623 0.484806 0.659629
```

Appendix A.2. Generalized Partial Credit Model

In Equation (1), the portion of the numerator within the exp() function when $k = 0$ and $C_i > 2$ (i.e., the conditions that make the FMP model equivalent to the GPCM) equals

$$\sum_{v=0}^{c}(b_{0_v i} + b_{1i}\theta). \tag{A4}$$

The following code demonstrates how to find the FMP $b_{0_c i}$ and b_{1i} from other parameterizations of the GPCM. To begin, we first generate GPCM data with five response categories.

```
rm(list = ls()) # clear environment
set.seed(876)
population_pars <- flexmet::sim_bmat(n_items = 10, k = 0, ncat = 5)$bmat
dat <- flexmet::sim_data(bmat = population_pars,
                         theta = rnorm(1000), maxncat = 5)
```

The `ltm` package with the `IRT.param = FALSE` option produces GPCM item parameter estimates that are already in the same format that the `flexmet` package uses for the FMP model.

```
library(ltm) # must be loaded in for the gpcm() function to run
ltm_fit1 <- ltm::gpcm(data = dat, IRT.param = FALSE)
head(ltm::coef.gpcm(ltm_fit1))
```

```
##    Catgr.1 Catgr.2 Catgr.3 Catgr.4 Dscrmn
## V1   1.046   0.865   0.149  -0.525  1.552
## V2   0.956   0.298   0.109  -0.604  2.049
## V3   0.748   0.947   0.731  -0.173  2.136
## V4   1.022   0.241  -0.460  -0.694  1.244
## V5   0.937   0.469   0.181  -0.616  2.289
## V6   0.644   0.326   0.097  -0.396  1.115
```

Here, columns `Catgr.1` through `Catgr.4` include parameters $b_{0_1 i}$ through $b_{0_4 i}$, and the `Dscrmn` column includes the b_{1i} parameters. We can verify that these are equivalent by comparing the response probabilities calculated from each package. Here (and through the rest of this section), any minor differences in predicted probabilities can be attributed to rounding.

```
ltm_fit1_FMP <- ltm::coef.gpcm(ltm_fit1)
i <- 1 # choose an item
th <- c(-1.35, 0, 1.35) # choose theta values
ltm::plot.gpcm(ltm_fit1, type = "ICC", items = i, z = th, plot = FALSE)
flexmet::irf_fmp(theta = th, ltm_fit1_FMP[i, ], maxncat = 5)
```

```
##               Catgr.1  Catgr.2  Catgr.3  Catgr.4
## [1,] 0.681183 0.238462 0.069684 0.009947 0.000724
## [2,] 0.043312 0.123246 0.292752 0.339686 0.201003
## [3,] 0.000040 0.000927 0.017905 0.168873 0.812255
##           [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 0.681086 0.238528 0.069706 0.009955 0.000725
## [2,] 0.043302 0.123249 0.292718 0.339750 0.200981
## [3,] 0.000040 0.000928 0.017909 0.168937 0.812185
```

The `ltm` package with the `IRT.param = TRUE` argument uses a slightly different parameterization of the GPCM. Namely, the numerator expression within the exp() (see Equation (A4)) is $\sum_{v=0}^{c} Dscrm_i(\theta - Catgr_{ci})$ so that $b_{0_c i} = -Catgr_{ci}Dscrm_i$ and $b_{1i} = Dscrm_i$.

```
# library(ltm)  # ltm package must be loaded to make the following run
ltm_fit2 <- ltm::gpcm(data = dat, IRT.param = TRUE)
ltm_pars2 <- ltm::coef.gpcm(ltm_fit2)
head(ltm_pars2)
```

```
##     Catgr.1 Catgr.2 Catgr.3 Catgr.4 Dscrmn
## V1   -0.674  -0.558  -0.096   0.338  1.552
## V2   -0.467  -0.146  -0.054   0.294  2.049
## V3   -0.350  -0.444  -0.342   0.081  2.136
## V4   -0.821  -0.194   0.369   0.558  1.244
## V5   -0.409  -0.205  -0.079   0.269  2.289
## V6   -0.578  -0.293  -0.087   0.354  1.115
## V7    0.251   0.325   0.801   0.622  0.765
## V8   -2.290   0.483   0.504   2.064  0.364
## V9   -0.524  -0.431   0.075  -0.149  0.474
## V10   0.302   0.096   0.481   0.301  0.903
```

We can then transform the outputted `ltm` parameters to FMP parameters and verify that each leads to the same predicted probabilities.

```
ltm_fit2_FMP <- cbind(-ltm_pars2[, 1] * ltm_pars2[, 5],
                      -ltm_pars2[, 2] * ltm_pars2[, 5],
                      -ltm_pars2[, 3] * ltm_pars2[, 5],
                      -ltm_pars2[, 4] * ltm_pars2[, 5],
                      ltm_pars2[, 5])
i <- 1 # choose an item
th <- c(-1.35, 0, 1.35) # choose theta values
ltm::plot.gpcm(ltm_fit2, type = "ICC", items = i, z = th, plot = FALSE)
flexmet::irf_fmp(theta = th, ltm_fit2_FMP[i, ], maxncat = 5)
```

```
##              Catgr.1 Catgr.2 Catgr.3 Catgr.4
## [1,] 0.68114 0.23847 0.06971 0.00996 0.00072
## [2,] 0.04328 0.12318 0.29269 0.33974 0.20110
## [3,] 0.00004 0.00093 0.01789 0.16883 0.81231
##          [,1]    [,2]    [,3]    [,4]    [,5]
## [1,] 0.68102 0.23852 0.06977 0.00996 0.00073
## [2,] 0.04326 0.12314 0.29274 0.33978 0.20108
## [3,] 0.00004 0.00093 0.01790 0.16888 0.81225
```

Next, the `mirt` package with option `itemtype = "gpcmIRT"` is very similar to the `ltm` package with option `IRT.param = TRUE`. The expression within the numerator $\exp()$ (see Equation (A4)) is $\sum_{v=0}^{c} a_i(\theta - b_{ci})$ so that the FMP $b_{0_c i} = -b_{ci}a_i$ and the FMP $b_{1i} = a_i$.

```
mirt_fit1 <- mirt::mirt(as.data.frame(dat), model = 1, itemtype = "gpcmIRT")
mirt_pars1 <- mirt::coef(mirt_fit1, simplify = TRUE)$items
head(mirt_pars1)
```

```
##          a1        b1        b2        b3       b4 c
## V1 1.604088 -0.607911 -0.496352 -0.063637 0.340979 0
## V2 2.307321 -0.434459 -0.114577 -0.011683 0.325999 0
## V3 2.350332 -0.320676 -0.380056 -0.290298 0.116407 0
## V4 1.318478 -0.755949 -0.152858  0.375798 0.559860 0
## V5 2.616824 -0.384260 -0.171619 -0.035806 0.304774 0
## V6 1.213649 -0.537403 -0.248817 -0.047386 0.377680 0
```

Here, the a_i parameters are in the first column, and the b_{1i}–b_{4i} parameters are in columns 2 through 5. The following code implements the appropriate transformation and verifies its equivalence to the FMP model as implemented in `flexmet`.

```
mirt_fit1_FMP <- cbind(-mirt_pars1[, 2] * mirt_pars1[, 1],
                       -mirt_pars1[, 3] * mirt_pars1[, 1],
                       -mirt_pars1[, 4] * mirt_pars1[, 1],
                       -mirt_pars1[, 5] * mirt_pars1[, 1],
                        mirt_pars1[, 1])
i <- 1 # choose an item
th <- c(-1.35, 0, 1.35) # choose theta values
mirt::probtrace(mirt::extract.item(mirt_fit1, i), Theta = th)
flexmet::irf_fmp(theta = th, bmat = mirt_fit1_FMP[i, ], maxncat = 5)
```

```
##           P.1      P.2      P.3      P.4      P.5
## [1,] 0.718438 0.218482 0.055555 0.007056 0.000468
## [2,] 0.050484 0.133860 0.296778 0.328673 0.190205
## [3,] 0.000038 0.000870 0.016826 0.162471 0.819796
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 0.718438 0.218482 0.055555 0.007056 0.000468
## [2,] 0.050484 0.133860 0.296778 0.328673 0.190205
## [3,] 0.000038 0.000870 0.016826 0.162471 0.819796
```

The `mirt` package also includes the `itemtype = "gpcm"` option for fitting the GPCM. Here, the expression in the numerator $\exp()$ (see Equation (A4)) is $ca_i\theta + d_i$ (note the lack of the summation operator), where c is the response category (e.g., 1, 2, 3, or 4). Because this expression is equivalent to the cumulative sum in the FMP model expression, some of the FMP-equivalent parameters must be found through subtraction. Namely, the FMP $b_{1i} = a_i$, $b_{0_1 i} = d_1$, and $b_{0_c i} = d_c - d_{c-1}$ for $c > 1$. Below, we fit the model and inspect the outputted model parameters.

```
mirt_fit2 <- mirt::mirt(as.data.frame(dat), model = 1, itemtype = "gpcm")
mirt_pars2 <- mirt::coef(mirt_fit2, simplify = TRUE)$items
head(mirt_pars2)
```

```
##          a1 ak0 ak1 ak2 ak3 ak4 d0       d1       d2       d3        d4
## V1 1.604585   0   1   2   3   4  0 0.976432 1.773830 1.877017  1.330860
## V2 2.308880   0   1   2   3   4  0 1.005227 1.271728 1.300226  0.548738
## V3 2.353282   0   1   2   3   4  0 0.757869 1.654055 2.338335  2.065157
## V4 1.318610   0   1   2   3   4  0 0.997297 1.199669 0.705070 -0.032312
## V5 2.619433   0   1   2   3   4  0 1.009280 1.460999 1.556443  0.759314
## V6 1.213762   0   1   2   3   4  0 0.652742 0.955466 1.013786  0.556153
```

Here, we only want to use the columns of estimated parameters: `a1`, `d1`, `d2`, `d3`, and `d4`. In the code below, the appropriate FMP parameters is found by manually subtracting the appropriate parameter estimates.

```
mirt_fit2_FMP <- cbind(mirt_pars2[, 8], # d1
                       mirt_pars2[, 9] - mirt_pars2[, 8], # d2 - d1
                       mirt_pars2[, 10] - mirt_pars2[, 9], # d3 - d2
                       mirt_pars2[, 11] - mirt_pars2[, 10], # d4 - d3
                       mirt_pars2[, 1]) # a1
i <- 1 # choose an item
```

```
th <- c(-1.35, 0, 1.35) # choose theta values
mirt::probtrace(mirt::extract.item(mirt_fit2, i), Theta = th)
flexmet::irf_fmp(theta = th, bmat = mirt_fit2_FMP[i, ], maxncat = 5)
```

```
##           P.1      P.2      P.3      P.4      P.5
## [1,] 0.718286 0.218571 0.055608 0.007066 0.000469
## [2,] 0.050336 0.133639 0.296647 0.328893 0.190485
## [3,] 0.000037 0.000866 0.016776 0.162280 0.820041
##           [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 0.718286 0.218571 0.055608 0.007066 0.000469
## [2,] 0.050336 0.133639 0.296647 0.328893 0.190485
## [3,] 0.000037 0.000866 0.016776 0.162280 0.820041
```

Finally, we illustrate the equivalence of the FMP model and the the GPCM as estimated in `TAM`. The parameterization used here is similar to that used by `mirt` with the `itemtype = "gpcm"` option. Here, the expression inside the exp() of Equation (A4) is $B_{ci}\theta + A_{ci}$.

```
tam_fit <- TAM::tam.mml.2pl(dat, irtmodel = "GPCM")
tam_fit$item
```

```
##     item    N     M xsi.item AXsi_.Cat1 AXsi_.Cat2 AXsi_.Cat3 AXsi_.Cat4
## I1    I1 1000 2.258   -0.351     -1.011     -1.839     -1.957     -1.406
## I2    I2 1000 2.083   -0.152     -0.955     -1.226     -1.288     -0.609
## I3    I3 1000 2.311   -0.527     -0.729     -1.638     -2.336     -2.106
## I4    I4 1000 1.974   -0.003     -1.002     -1.218     -0.733     -0.012
## I5    I5 1000 2.106   -0.203     -0.944     -1.389     -1.521     -0.814
## I6    I6 1000 2.134   -0.147     -0.637     -0.945     -1.017     -0.588
## I7    I7 1000 1.498    0.398      0.206      0.470      1.098      1.592
## I8    I8 1000 1.849    0.076     -0.828     -0.645     -0.454      0.306
## I9    I9 1000 2.155   -0.113     -0.240     -0.435     -0.389     -0.451
## I10  I10 1000 1.661    0.284      0.292      0.397      0.847      1.134
##     B.Cat1.Dim1 B.Cat2.Dim1 B.Cat3.Dim1 B.Cat4.Dim1
## I1        1.577       3.153       4.730       6.307
## I2        2.142       4.285       6.427       8.570
## I3        2.213       4.425       6.638       8.851
## I4        1.272       2.544       3.816       5.088
## I5        2.410       4.819       7.229       9.638
## I6        1.151       2.302       3.452       4.603
## I7        0.781       1.561       2.342       3.123
## I8        0.371       0.742       1.113       1.484
## I9        0.483       0.965       1.448       1.931
## I10       0.917       1.835       2.752       3.670
```

Note in the above that the columns beginning with `B.Cat` are integer multiples of each other; that is, the values in `B.Cat2.Dim1` are double those in `B.Cat1.Dim1`, those in `B.Cat3.Dim1` are triple those in `B.Cat1.Dim1`, and so forth. Therefore, we only need the `B.Cat1.Dim1` column, which is equivalent to the FMP b_{1i}. The columns beginning with `AXsi` are equivalent to the negative d_i parameters in the previous `mirt` example (i.e., `mirt` with option `itemtype = "gpcm"`). As such, subtraction could be used to find the FMP parameters in the same manner as the previous `mirt` example. However, it is simpler to use the estimated `xsi` parameters in `TAM`, printed below.

```
head(tam_fit$xsi)
```

```
##               xsi   se.xsi
## I1_Cat1 -1.011297 0.106840
## I1_Cat2 -0.828046 0.101751
## I1_Cat3 -0.117161 0.095900
## I1_Cat4  0.550557 0.092844
## I2_Cat1 -0.955250 0.107489
## I2_Cat2 -0.270634 0.113281
```

Notice that for item 1, the `xsi` for category 1 is the same as `AXsi_.Cat1`, the `xsi` for category 2 equals `AXsi_.Cat2 - AXsi_.Cat1`, and so forth. Therefore, the negative `xsi` parameters are equal to the FMP $b_{0_c i}$ parameters. The following code builds a matrix of FMP parameters from the `TAM` output and verifies that both methods produce the same category response probabilities.

```
tam_fit_FMP <- cbind(-matrix(tam_fit$xsi$xsi, ncol = 4, byrow = TRUE),
                     tam_fit$item$B.Cat1.Dim1)
th <- attr(CDM::IRT.irfprob(tam_fit), "theta")[c(9, 11, 13)]
t(CDM::IRT.irfprob(tam_fit)[i , , c(9, 11, 13)])
flexmet::irf_fmp(theta = th, bmat = tam_fit_FMP[i, ], maxncat = 5)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 0.631367 0.261690 0.090305 0.015307 0.001331
## [2,] 0.047186 0.129715 0.296881 0.333767 0.192451
## [3,] 0.000098 0.001778 0.026997 0.201300 0.769827
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 0.631355 0.261696 0.090310 0.015309 0.001331
## [2,] 0.047180 0.129705 0.296874 0.333776 0.192465
## [3,] 0.000098 0.001778 0.026994 0.201292 0.769839
```

Appendix B. Relationship between Greek-Letter and Polynomial Coefficient Parameterizations of the FMP Model

This appendix presents explicit formulas to transform Greek-letter FMP parameters to FMP polynomial coefficients for items with $k_i \in \{0, 1, 2\}$. Matrix equations to implement these equations more efficiently, as well as extensions to larger k values, are available in other sources [3,11,12,18,19]. These equations are provided for didactic purposes only; the practical implementation of these transformations can always be performed using the `greek2b` and `b2greek` functions in `flexmet`.

Appendix B.1

k = 0

$$b_{0_c i} = \xi_{ci} \tag{A5}$$

$$b_{1i} = \exp(\omega_i) \tag{A6}$$

Appendix B.2

k = 1

$$b_{0_c i} = \xi_{ci} \tag{A7}$$
$$b_{1i} = \exp(\omega_i) \tag{A8}$$
$$b_{2i} = -\alpha_{1i}\exp(\omega_i) \tag{A9}$$
$$b_{3i} = (\alpha_{1i}^2 + \exp(\tau_{1i}))\frac{\exp(\omega_i)}{3} \tag{A10}$$

Appendix B.3

k = 2

$$b_{0_c i} = \xi_{ci} \tag{A11}$$
$$b_{1i} = \exp(\omega_i) \tag{A12}$$
$$b_{2i} = -(\alpha_{1i} + \alpha_{2i})\exp(\omega_i) \tag{A13}$$
$$b_{3i} = \left(\alpha_{1i}^2 + \alpha_{2i}^2 + \exp(\tau_{1i}) + \exp(\tau_{2i}) + 4\alpha_{1i}\alpha_{2i}\right)\frac{\exp(\omega_i)}{3} \tag{A14}$$
$$b_{4i} = -\left(\alpha_{1i}(\alpha_{2i}^2 + \exp(\tau_{2i})) + \alpha_{2i}(\alpha_{1i}^2 + \exp(\tau_{1i}))\right)\frac{\exp(\omega_i)}{2} \tag{A15}$$
$$b_{5i} = (\alpha_{1i}^2 + \exp(\tau_{1i}))(\alpha_{2i}^2 + \exp(\tau_{2i}))\frac{\exp(\omega_i)}{5} \tag{A16}$$

Appendix C

Code to Reproduce Figure 1.

```
library(flexmet)

# find matrix of b-parameters
bmat <- rbind(greek2b(xi = c(0.49, NA, NA), omega = 0.25,
                alpha = c(0, 0), tau = c(-Inf, -Inf)),
              greek2b(xi = c(-0.87, NA, NA), omega = -0.57,
                alpha = c(-0.63, 0), tau = c(-0.12, -Inf)),
              greek2b(xi = c(-0.54, NA, NA), omega = -0.24,
                alpha = c(1.13, -0.40), tau = c(-1.76, -1.14)),
              greek2b(xi = c(0.58, 0.49, -0.05), omega = -0.41,
                alpha = c(0, 0), tau = c(-Inf, -Inf)),
              greek2b(xi = c(0.97, 0.38, -0.15), omega = 0.58,
                alpha = c(0.45, 0), tau = c(-1.42, -Inf)),
              greek2b(xi = c(0.79, 0.15, -1.20), omega = 0.57,
                alpha = c(-0.22, -0.59), tau = c(-1.57, -2.45)))

# set up plotting window
par(mfrow = c(3, 2))

for(i in 1:3){
  # items 1, 2, 3
  curve(irf_fmp(x, bmat[i, ], maxncat = 4, returncat = 1),
        xlim = c(-3, 3), ylim = c(0, 1), main = paste("Item", i),
        xlab = expression(theta), ylab = "probability")

  # items 4, 5, 6
  curve(irf_fmp(x, bmat[i+3, ], maxncat = 4, returncat = 0),
        xlim = c(-3, 3), ylim = c(0, 1), main = paste("Item", i + 3),
        xlab = expression(theta), ylab = "probability")
  curve(irf_fmp(x, bmat[i+3, ], maxncat = 4, returncat = 1), add = TRUE)
  curve(irf_fmp(x, bmat[i+3, ], maxncat = 4, returncat = 2), add = TRUE)
  curve(irf_fmp(x, bmat[i+3, ], maxncat = 4, returncat = 3), add = TRUE)
}
```

Appendix D

Code to Find AIC-Based k Values for Simulated Data.

```
library(flexmet)

set.seed(123)
bmat <- sim_bmat(n_items = 20, k = 0, ncat = 2)$bmat
theta1 <- exp(rnorm(5000) / 2) - 2
theta2 <- rnorm(5000)
data1 <- sim_data(bmat = bmat, theta = theta1)
data2 <- sim_data(bmat = bmat, theta = theta2)
tsur1 <- get_surrogates(data1)
tsur2 <- get_surrogates(data2)

# choose k values for data set 1
k1 <- rep(NA, 20)
for(i in 1:20){
  # initialize
  mod_a <- mod_b <- fmp_1(data1[, i], k = 0, tsur = tsur1)
  while(mod_a$AIC >= mod_b$AIC){ # chile
    mod_a <- mod_b
    mod_b <- fmp_1(data1[, i], k = mod_a$k + 1, tsur = tsur1)
  }
  k1[i] <- mod_a$k
}

# choose k values for data set 2
k2 <- rep(NA, 20)
for(i in 1:20){
  # initialize
  mod_a <- mod_b <- fmp_1(data2[, i], k = 0, tsur = tsur2)
  while(mod_a$AIC >= mod_b$AIC){
    mod_a <- mod_b
    mod_b <- fmp_1(data2[, i], k = mod_a$k + 1, tsur = tsur2)
  }
  k2[i] <- mod_a$k
}
```

Appendix E

Code to Reproduce Figure 3.

```
library(flexmet)

# read in the linking coefficients estimated in the main paper
tvec_sl0 <- c(-0.9278, 0.6137) # sl0$tvec
tvec_sl1 <- c(-0.9970, 0.5155, 0.1188, 0.0091) # sl1$tvec
tvec_sl2 <- c(-0.8871, 0.0012, 0.0018, 0.2412, -0.0001, 0.0003) # sl2$tvec

# calculate theta values
tstar <- seq(-3, 3, length = 100)
theta <- exp(tstar / 2) - 2
theta_0 <- sapply(tstar, fw_poly, coefs = tvec_sl0)
theta_1 <- sapply(tstar, fw_poly, coefs = tvec_sl1)
theta_2 <- sapply(tstar, fw_poly, coefs = tvec_sl2)
```

```
# plot
plot(tstar, theta, type = 'l', xlim = c(-3, 3), ylim = c(-3, 3),
     ylab = expression(theta), xlab = expression(paste(theta, "*")))
points(tstar, theta_0, type = 'l', lty = 2)
points(tstar, theta_1, type = 'l', lty = 3)
points(tstar, theta_2, type = 'l', lty = 4)
legend("topleft", lty = 1:4,
       legend = c("true relationship",
                  "k_theta = 0",
                  "k_theta = 1",
                  "k_theta = 2"))
```

Appendix F

Code to Reproduce Figure 4.

```
# read in tcals parameters
tcals <- data.frame(
  a = c(2.225, 1.174, 2.104, 2.691, 1.256, 1.447, 1.947, 2.817, 2.664,
        3.218, 2.572, 2.311, 1.768, 1.615, 2.107, 1.455, 1.722, 1.789,
        3.533, 1.615, 1.176, 2.218, 2.405, 2.029, 1.973, 1.030, 0.814,
        1.770, 1.322, 2.841, 2.310, 0.407, 0.866, 0.912, 1.753, 2.442,
        1.756, 1.728, 0.888, 2.771, 1.469, 1.736, 2.131, 3.826, 3.171,
        2.029, 1.248, 1.244, 2.621, 2.024, 2.194, 0.739, 2.575, 2.682,
        1.751, 1.176, 1.025, 2.093, 2.568, 3.076, 2.866, 2.613, 3.983,
        1.639, 1.899, 1.782, 2.698, 3.162, 2.361, 2.655, 1.901, 1.016,
        0.992, 1.346, 1.385, 1.256, 2.881, 1.166, 1.462, 2.968, 2.058,
        1.935, 1.600, 2.189, 1.933),
b = c(-1.885, -2.411, -2.439, -1.282, -1.930, -1.554, -1.075, -0.551,
      -0.626, -0.246,  0.251, 0.189, -1.534, -3.088, -1.139, -1.673,
      -1.349, -1.481, -0.962, -1.441, -0.549, -0.814, -0.454, -0.008,
      0.805, -1.229,  0.687, -1.104, -1.441, -0.257, -0.282, -1.103,
      -2.224, -3.457, -1.658, -2.070, -1.782, -1.991, -2.815, -1.412,
      -1.531, -1.945, -0.810, -0.729, -0.663, -0.993, -2.392, -1.743,
      -1.867, -1.696, -1.284, -2.356, -1.444, -1.107, -1.352, -2.014,
      -1.364, -1.232, -0.565, -0.265, -0.032, -0.115,  0.120, -2.139,
      -1.889, -1.639, -1.006, -0.621, -0.248, -0.086, -1.107, -1.268,
      -1.115, -0.612, -0.589, -0.182,  0.840, -1.377, -1.103,  0.711,
      -0.265, -0.648, -1.093, -0.664, -0.952),
c = c(0.210, 0.212, 0.192, 0.294, 0.170, 0.205, 0.246, 0.105, 0.095, 0.078,
      0.147, 0.178, 0.185, 0.202, 0.224, 0.182, 0.195, 0.144, 0.168, 0.133,
      0.223, 0.115, 0.076, 0.117, 0.094, 0.183, 0.215, 0.192, 0.134, 0.341,
      0.239, 0.236, 0.195, 0.194, 0.210, 0.168, 0.233, 0.214, 0.201, 0.252,
      0.218, 0.253, 0.295, 0.212, 0.222, 0.224, 0.196, 0.209, 0.161, 0.156,
      0.227, 0.213, 0.175, 0.275, 0.228, 0.222, 0.172, 0.243, 0.166, 0.235,
      0.236, 0.092, 0.063, 0.200, 0.191, 0.187, 0.148, 0.256, 0.273, 0.330,
      0.238, 0.210, 0.203, 0.153, 0.171, 0.196, 0.233, 0.190, 0.283, 0.089,
      0.142, 0.304, 0.384, 0.180, 0.316))

fmp_pars <- cbind(-tcals$a * tcals$b, tcals$a)

tvec <- c(-10.484125, 0.406875, -0.006091, 0.000032) # coef(mp_res[[2]])
transformed_bmat <- t(apply(fmp_pars, 1, transform_b, tvec = tvec))

i1 <- 5
```

```
i2 <- 24
i3 <- 27

par(mfrow = c(2, 2))

# upper-left:
curve(irf_fmp(x, fmp_pars[i1, ], cvec = tcals$c[i1]),
      xlim = c(-4, 4), ylim = c(0, 1),
      xlab = expression(theta),
      ylab = "probability", col = 2)
curve(irf_fmp(x, fmp_pars[i2, ], cvec = tcals$c[i2]),
      add = TRUE, col = 3)
curve(irf_fmp(x, fmp_pars[i3, ], cvec = tcals$c[i3]),
      add = TRUE, col = 4)


# upper-right:
curve(irf_fmp(x, transformed_bmat[i1, ], cvec = tcals$c[i1]),
      xlim = c(0, 100), ylim = c(0, 1),
      xlab = expression(paste(theta, "*")),
      ylab = "probability", col = 2)
curve(irf_fmp(x, transformed_bmat[i2, ], cvec = tcals$c[i2]),
      add = TRUE, col = 3)
curve(irf_fmp(x, transformed_bmat[i3, ], cvec = tcals$c[i3]),
      add = TRUE, col = 4)

# lower-left:
x1 <- seq(-4, 4, length = 1000)
y1 <- rowSums(iif_fmp(theta = x1, bmat = fmp_pars, cvec = tcals$c))
plot(x1, y1, type = 'l', xlab = expression(theta),
     ylab = "test information")

# lower-right:
x2 <- seq(0, 100, length = 1000)
y2 <- rowSums(iif_fmp(theta = x2, bmat = transformed_bmat,
                      maxncat = 2, cvec = tcals$c))
plot(x2, y2, type = 'l', xlab = expression(paste(theta, "*")),
     ylab = "test information")
```

References

1. Birnbaum, A. Some latent trait models and their use in inferring an examinee's ability. In *Statistical Theories of Mental Test Scores*; Lord, F., Novick, M., Eds.; Addison-Wesley: Reading, MA, USA, 1968; pp. 397–479.
2. Molenaar, I.W. Thirty years of nonparametric item response theory. *Appl. Psychol. Meas.* **2001**, *25*, 295–299. [CrossRef]
3. Liang, L.; Browne, M.W. A quasi-parametric method for fitting flexible item response functions. *J. Educ. Behav. Stat.* **2015**, *40*, 5–34. [CrossRef]
4. Ramsay, J.; Winsberg, S. Maximum marginal likelihood estimation for semiparametric item analysis. *Psychometrika* **1991**, *56*, 365–379. [CrossRef]
5. Mokken, R.J. *A Theory and Procedure of Scale Analysis*; De Gruyter Mouton: Berlin, Germany; New York, NY, USA, 2011.
6. Ramsay, J.O. Kernel smoothing approaches to nonparametric item characteristic curve estimation. *Psychometrika* **1991**, *56*, 611–630. [CrossRef]
7. Falk, C.F.; Feuerstahler, L.M. On the Performance of Semi-and Nonparametric Item Response Functions in Computer Adaptive Tests. *Educ. Psychol. Meas.* **2021**. [CrossRef]
8. Xu, X.; Douglas, J. Computerized adaptive testing under nonparametric IRT models. *Psychometrika* **2006**, *71*, 121–137. [CrossRef]
9. Falk, C.F.; Fischer, F. More flexible response functions for the PROMIS physical functioning item bank by application of a monotonic polynomial approach. *Qual. Life Res.* **2021**, 1–11. [CrossRef]

10. Wiberg, M.; Ramsay, J.O.; Li, J. Optimal scores: An alternative to parametric item response theory and sum scores. *Psychometrika* **2019**, *84*, 310–322. [CrossRef] [PubMed]
11. Liang, L. A Semi-Parametric Approach to Estimating Item Response Functions. Ph.D. Thesis, The Ohio State University, Columbus, OH, USA, 2007.
12. Falk, C.F.; Cai, L. Maximum marginal likelihood estimation of a monotonic polynomial generalized partial credit model with applications to multiple group analysis. *Psychometrika* **2016**, *81*, 434–460. [CrossRef] [PubMed]
13. Muraki, E. A Generalized Partial Credit Model: Application of an EM Algorithm. *Appl. Psychol. Meas.* **1992**, *16*, 159–176. [CrossRef]
14. Rizopoulos, D. ltm: An R package for Latent Variable Modelling and Item Response Theory Analyses. *J. Stat. Softw.* **2006**, *17*, 1–25. [CrossRef]
15. Chalmers, R.P. mirt: A multidimensional item response theory package for the R environment. *J. Stat. Softw.* **2012**, *48*, 1–29. [CrossRef]
16. Robitzsch, A.; Kiefer, T.; Wu, M. TAM: Test Analysis Modules; R Package Version 3.7-16. 2021. Available online: https://search.r-project.org/CRAN/refmans/TAM/html/TAM-package.html (accessed on 16 August 2021).
17. Elphinstone, C. A target distribution model for nonparametric density estimation. *Commun. Stat.-Theory Methods* **1983**, *12*, 161–198. [CrossRef]
18. Feuerstahler, L. Exploring Alternate Latent Trait Metrics with the Filtered Monotonic Polynomial IRT Model. Ph.D. Thesis, University of Minnesota, Minneapolis, MN, USA, 2016.
19. Feuerstahler, L.M. Metric transformations and the filtered monotonic polynomial item response model. *Psychometrika* **2019**, *84*, 105–123. [CrossRef]
20. Bock, R.D.; Aitkin, M. Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika* **1981**, *46*, 443–459. [CrossRef]
21. Bock, R.D.; Mislevy, R.J. Adaptive EAP estimation of ability in a microcomputer environment. *Appl. Psychol. Meas.* **1982**, *6*, 431–444. [CrossRef]
22. Van der Linden, W.J.; Barrett, M.D. Linking item response model parameters. *Psychometrika* **2016**, *81*, 650–673. [CrossRef]
23. Haebara, T. Equating logistic ability scales by a weighted least squares method. *Jpn. Psychol. Res.* **1980**, *22*, 144–149. [CrossRef]
24. Stocking, M.L.; Lord, F.M. Developing a common metric in item response theory. *Appl. Psychol. Meas.* **1983**, *7*, 201–210. [CrossRef]
25. Magis, D.; Barrada, J.R. Computerized adaptive testing with R: Recent updates of the package catR. *J. Stat. Softw.* **2017**, *76*, 1–19. [CrossRef]
26. Falk, C.F.; Cai, L. Semiparametric item response functions in the context of guessing. *J. Educ. Meas.* **2016**, *53*, 229–247. [CrossRef]
27. Murray, K.; Müller, S.; Turlach, B.A. Revisiting fitting monotone polynomials to data. *Comput. Stat.* **2013**, *28*, 1989–2005. [CrossRef]
28. Murray, K.; Müller, S.; Turlach, B. Fast and flexible methods for monotone polynomial fitting. *J. Stat. Comput. Simul.* **2016**, *86*, 2946–2966. [CrossRef]
29. Lord, F.M. *Applications of Item Response Theory to Practical Testing Problems*; Erlbaum: New York, NY, USA, 1980.