plink: An R Package for Linking Mixed-Format Tests Using IRT-Based Methods

Jonathan P. Weeks
University of Colorado at Boulder

Abstract

This introduction to the R package **plink** is a (slightly) modified version of Weeks (2010), published in the *Journal of Statistical Software*.

The R package **plink** has been developed to facilitate the linking of mixed-format tests for multiple groups under a common item design using unidimensional and multidimensional IRT-based methods. This paper presents the capabilities of the package in the context of the unidimensional methods. The package supports nine unidimensional item response models (the Rasch model, 1PL, 2PL, 3PL, graded response model, partial credit and generalized partial credit model, nominal response model, and multiple-choice model) and four separate calibration linking methods (mean/sigma, mean/mean, Haebara, and Stocking-Lord). It also includes functions for importing item and/or ability parameters from common IRT software, conducting IRT true-score and observed-score equating, and plotting item response curves and parameter comparison plots.

Keywords: item response theory, separate calibration, chain linking, mixed-format tests, R.

1. Introduction

In many measurement scenarios there is a need to compare results from multiple tests, but depending on the statistical properties of these measures and/or the sample of examinees, scores across tests may not be directly comparable; in most instances they are not. To create a common scale, scores from all tests of interest must be transformed to the metric of a reference test. This process is known generally as *linking*, although other terms like *equating* and *vertical scaling* are often used to refer to specific instantiations (see Linn 1993, for information on the associated terminology). Linking methods were originally developed to equate observed scores for parallel test forms (Hull 1922; Kelley 1923; Gulliksen 1950; Levine 1955). These approaches work well when the forms are similar in terms of difficulty and reliability, but as the statistical specifications of the tests diverge, the comparability of scores across tests becomes increasingly unstable (Petersen, Cook, and Stocking 1983; Yen 1986).

Thurstone (1925, 1938) developed observed score methods for creating vertical scales when the difficulties of the linked tests differ substantively. These methods depend on item p-values or empirical score distributions which are themselves dependent on the sample of examinees and the particular items included on the tests. As such, these approaches can be unreliable. Lord and Novick (1968) argued that in order to maintain a consistent scale, the linking approach must be based on a stable scaling model (i.e., a model with invariant item parameters). With the advent of item response theory (IRT; Lord 1952; Lord and Novick 1968; Lord 1980)

this became possible. Today, IRT-based linking is the most commonly used approach for developing vertical scales, and it is being used increasingly for equating (particularly in the development of calibrated item banks).

The R (R Development Core Team 2010) package **plink**, available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=plink`, was developed to facilitate the linking of mixed-format tests for multiple groups under a common item design (Kolen and Brennan 2004) using unidimensional and multidimensional IRT-based methods. The aim of this paper is to present the package with a specific focus on the unidimensional methods. An explication of the multidimensional methods will be described in a future article. This paper is divided into three main sections and two appendices. Section 2 provides an overview of the item response models and linking methods supported by **plink**. Section 3 describes how to format the various objects needed to run the linking function, and Section 4 illustrates how to link a set of tests using the `plink` function. Appendix A provides a brief description of additional features, and Appendix B.1 presents a comparison and critique of available linking software.

2. Models and methods

plink supports nine[1] unidimensional item response models (the Rasch model, 1PL, 2PL, 3PL, graded response model, partial credit and generalized partial credit model, nominal response model, and multiple-choice model) and four separate calibration linking methods (mean/sigma, mean/mean, Haebara, and Stocking-Lord). All of these models and methods are well documented in the literature, although the parameterizations can vary. The following two sub-sections are included to acquaint the reader with the specific parameterizations used in the package.

2.1. Item response models

Let the variable X_{ij} represent the response of examinee i to item j. Given a test consisting of dichotomously scored items, $X_{ij} = 1$ for a correct item response, and $X_{ij} = 0$ for an incorrect response. The response probabilities for the three-parameter logistic model (3PL; Birnbaum 1968) take the following form

$$P_{ij} = P\left(X_{ij} = 1|\theta_i, a_j, b_j, c_j\right) = c_j + (1 - c_j)\frac{\exp\left[Da_j\left(\theta_i - b_j\right)\right]}{1 + \exp\left[Da_j\left(\theta_i - b_j\right)\right]} \tag{1}$$

where θ_i is an examinee's ability on a single construct, a_j is the item discrimination, b_j is the item difficulty, c_j is the lower asymptote (guessing parameter), and D is a scaling constant.

If the guessing parameter is constrained to be zero, Equation 1 becomes the two-parameter logistic model (2PL; Birnbaum 1968), and if it is further constrained so that the discrimination parameters for all items are equal, it becomes the one-parameter logistic model (1PL). The Rasch model (Rasch 1960) is a special case of the 1PL where all of the item discriminations are constrained to equal one.

When items are polytomously scored (i.e., items with three or more score categories), the response X_{ij} is coded using a set of values $k = \{1, ..., K_j\}$ where K_j is the total number of

[1] By constraining the parameters in these models, other models like Andrich's (1978) rating scale model or Samejima's (1979) extension of the nominal response model can be specified.

categories for item j. When the values of k correspond to successively ordered categories, the response probabilities can be modeled using either the graded response model (GRM; Samejima 1969) or the generalized partial credit model (GPCM; Muraki 1992). The graded response model takes the following form

$$\tilde{P}_{ijk} = \tilde{P}\left(X_{ij} = k|\theta_i, a_j, b_{jk}\right) = \begin{cases} 1 & k = 1 \\ \dfrac{\exp\left[Da_j\left(\theta_i - b_{jk}\right)\right]}{1 + \exp\left[Da_j\left(\theta_i - b_{jk}\right)\right]} & 2 \leq k \leq K_j \\ 0 & k > K_j \end{cases} \tag{2}$$

where a_j is the item slope and b_{jk} is a threshold parameter. The threshold parameters can be alternately formatted as deviations from an item-specific difficulty commonly referred to as a location parameter. That is, b_{jk} can be respecified as $b_j + g_{jk}$ where the location parameter b_j is equal to the mean of the b_{jk} and the g_{jk} are deviations from this mean.

In Equation 2, the $\tilde{P}_{ijk}$ correspond to cumulative probabilities, yet the equation can be reformulated to identify the probability of responding in a given category. This is accomplished by taking the difference between the $\tilde{P}_{ijk}$ for adjacent categories. These category probabilities are formulated as

$$P_{ijk} = \tilde{P}_{ijk} - \tilde{P}_{ij(k+1)}. \tag{3}$$

The generalized partial credit model takes the following form

$$P_{ijk} = P\left(X_{ij} = v|\theta_i, a_j, b_{jk}\right) = \frac{\exp\left[\sum_{v=1}^{k} Da_j\left(\theta_i - b_{jv}\right)\right]}{\sum_{h=1}^{K_j} \exp\left[\sum_{v=1}^{h} Da_j\left(\theta_i - b_{jv}\right)\right]} \tag{4}$$

where b_{jk} is an intersection or step parameter. As with the graded response model, the b_{jk} for each item can be reformulated to include a location parameter and step-deviation parameters (e.g., $b_j + g_{jk}$). Further, the slope parameters for the GPCM can be constrained to be equal across all items. When they equal one, this is known as the partial credit model (PCM; Masters 1982). For both the PCM and the GPCM, the parameter b_{j1} can be arbitrarily set to any value because it is cancelled from the numerator and denominator (see Muraki 1992 for more information). For all of the functions in **plink** that use either of these models, b_{j1} is excluded, meaning only $K_j - 1$ step parameters should be specified.

The GRM, PCM, and GPCM assume that the values for k correspond to successively ordered categories, but if the responses are not assumed to be ordered, they can be modeled using the nominal response model (NRM; Bock 1972) or the multiple-choice model (MCM; Thissen and Steinberg 1984). The nominal response model takes the following form

$$P_{ijk} = P\left(X_{ij} = k|\theta_i, a_{jk}, b_{jk}\right) = \frac{\exp\left(a_{jk}\theta_i + b_{jk}\right)}{\sum_{h=1}^{K_j} \exp\left(a_{jh}\theta_i + b_{jh}\right)} \tag{5}$$

where a_{jk} is a category slope and b_{jk} is a category "difficulty" parameter. For the purpose of identification, the model is typically specified under the constraint that

$$\sum_{k=1}^{K_j} a_{jk} = 0 \quad \text{and} \quad \sum_{k=1}^{K_j} b_{jk} = 0.$$

The final model supported by **plink** is the multiple-choice model. It is an extension of the NRM that includes lower asymptotes on each of the response categories and additional parameters for a "do not know" category. The model is specified as

$$P_{ijk} = P\left(X_{ij} = k|\theta_i, a_{jk}, b_{jk}, c_{jk}\right) = \frac{\exp\left(a_{jk}\theta_i + b_{jk}\right) + c_{jk}\exp\left(a_{j0}\theta_i + b_{j0}\right)}{\sum_{h=0}^{K_j}\exp\left(a_{jh}\theta_i + b_{jh}\right)} \tag{6}$$

where K_j is equal to the number of actual response categories plus one, a_{j0} and b_{j0} are the slope and category parameters respectively for the "do not know" category, and a_{jk} and b_{jk} have the same interpretation as the parameters in Equation 5. This model is typically identified using the same constraints on a_{jk} and b_{jk} as the NRM, and given that c_{jk} represents the proportion of individuals who "guessed" a specific distractor, the MCM imposes an additional constraint, where

$$\sum_{k=1}^{K_j} c_{jk} = 1.$$

2.2. Calibration methods

The ultimate goal of test linking is to place item parameters and/or ability estimates from two or more tests onto a common scale. When there are only two tests, this involves finding a set of linking constants to transform the parameters from one test (the *from* scale) to the scale of the other (the *to* scale). The parameters associated with these tests are subscripted by an F and T respectively. When there are more than two tests, linking constants are first estimated for each pair of "adjacent" tests (see Section 3.4) and then chain-linked together to place the parameters for all tests onto a base scale. For a given pair of tests, the equation to transform θ_F to the θ_T scale is

$$\theta_T = A\theta_F + B = \theta_F^* \tag{7}$$

where the linking constants A and B are used to adjust the standard deviation and mean respectively, and the $*$ denotes a transformed value on the *to* scale. See Kim and Lee (2006) for an explanation of the properties and assumptions of IRT that form the basis for this equation, the following transformations, and a more detailed explanation of the linking methods.

Since the item parameters are inextricably tied to the θ scale, any linear transformation of the scale will necessarily change the item parameters such that the expected probabilities will remain unchanged. As such, it can be readily shown that a_j and b_{jk} for the GRM, GPCM, and dichotomous models on the *from* scale can be transformed to the *to* scale by

$$a_{jF}^* = a_{jF}/A \tag{8a}$$

$$b_{jkF}^* = Ab_{jkF} + B \tag{8b}$$

where the constants A and B are the same as those used to transform θ_F (Lord and Novick 1968; Baker 1992). Since the NRM and MCM are parameterized using a slope/intercept

formulation (i.e., $a_{jk}\theta_i + b_{jk}$) rather than a slope/difficulty formulation (i.e., $a_j[\theta_i - b_{jk}]$), the slopes and category parameters are transformed using (Baker 1993a; Kim and Hanson 2002)

$$a^*_{jkF} = a_{jkF}/A \tag{9a}$$
$$b^*_{jkF} = b_{jkF} - (B/A)\, a_{jkF}. \tag{9b}$$

When lower asymptote parameters are included in the model, as with the 3PL and MCM, they are unaffected by the transformation; hence, $c^*_{jF} = c_{jF}$.

Equations 7 to 9b illustrate the transformation of item and ability parameters from the *from* scale to the *to* scale; however, a reformulation of these equations can be used to transform the parameters on the *to* scale to the *from* scale. These transformations are important when considering symmetric linking (discussed later). The item parameters for the GRM, GPCM and dichotomous models are transformed by

$$a^{\#}_{jT} = Aa_{jT} \tag{10a}$$
$$b^{\#}_{jkT} = (b_{jkT} - B)/A \tag{10b}$$

and the item parameters for the NRM and MCM are transformed by

$$a^{\#}_{jkT} = Aa_{jkT} \tag{11a}$$
$$b^{\#}_{jkT} = b_{jkT} + Ba_{jkT} \tag{11b}$$

where the # denotes a transformed value on the *From* scale. Again, the lower asymptote parameters remain unaffected, so $c^{\#}_{jT} = c_{jT}$.

This package supports four of the most commonly used methods for estimating linking constants under an equivalent or non-equivalent groups common item design (Kolen and Brennan 2004). Within this framework, a subset of $S \leq J$ common items between the *to* and *from* tests are used to estimate the constants. The mean/sigma (Marco 1977) and mean/mean (Loyd and Hoover 1980) methods, known as moment methods, are the simplest approaches to estimating A and B because they only require the computation of means and standard deviations for various item parameters. For the mean/sigma, only the b_{sk} are used. That is,

$$A = \frac{\sigma(b_T)}{\sigma(b_F)} \tag{12a}$$
$$B = \mu(b_T) - A\mu(b_F) \tag{12b}$$

where the means and standard deviations are taken over all S common items and K_s response categories. One potential limitation of this approach, however, is that it does not consider the slope parameters. The mean/mean, on the other hand, uses both the a_{sk} and b_{sk} to estimate the linking constants where

$$A = \frac{\mu(a_F)}{\mu(a_T)} \tag{13a}$$
$$B = \mu(b_T) - A\mu(b_F). \tag{13b}$$

Both of these approaches assume that the items are parameterized using a slope/difficulty formulation, but because NRM and MCM items use a slope/intercept parameterization, the

b_{sk} must be reformulated as $\tilde{b}_{sk} = -b_{sk}/a_{sk}$ before computing the means and standard deviations. Given this reparameterization, there are several issues related to the use of NRM and MCM items with the moment methods (see Kim and Lee 2006, for more information).

As an alternative to the moment methods, Haebara (1980) and Stocking and Lord (1983) developed characteristic curve methods that use an iterative approach to estimate the linking constants by minimizing the sum of squared differences between item characteristic curves and test characteristic curves for the common items for the two methods respectively. These methods are typically implemented by finding the constants that best characterize the *from* scale parameters on the *to* scale; however, this assumes that the parameters on the *to* scale were estimated without error. For this reason, Haebara (1980) proposed a criterion that simultaneously considers the transformation of parameters from the *from* scale to the *to* scale and vice-versa. The former case—the typical implementation—is referred to as a non-symmetric approach and the later is referred to as a symmetric approach.

The Haebara method minimizes the following criterion

$$Q = Q_1 + Q_2 \tag{14a}$$

where

$$Q_1 = \frac{1}{L_1} \sum_{m=1}^{M} \sum_{s=1}^{S} \sum_{k=1}^{K_s} \left[P_{sk}\left(\theta_{mT}\right) - P^{*}_{sk}\left(\theta_{mT}\right) \right]^2 W_1\left(\theta_{mT}\right) \tag{14b}$$

and

$$Q_2 = \frac{1}{L_2} \sum_{m=1}^{M} \sum_{s=1}^{S} \sum_{k=1}^{K_s} \left[P_{sk}\left(\theta_{mF}\right) - P^{\#}_{sk}\left(\theta_{mF}\right) \right]^2 W_2\left(\theta_{mF}\right) \tag{14c}$$

where

$$L_1 = \sum_{m=1}^{M} W_1\left(\theta_{mT}\right) \sum_{s=1}^{S} K_s \quad \text{and} \quad L_2 = \sum_{m=1}^{M} W_2\left(\theta_{mF}\right) \sum_{s=1}^{S} K_s.$$

The θ_{mT} are a set of M points on the *to* scale where differences in expected probabilities are evaluated, $P_{sk}\left(\theta_{mT}\right)$ are expected probabilities based on the untransformed *to* scale common item parameters, $P^{*}_{sk}\left(\theta_{mT}\right)$ are expected probabilities based on the transformed *from* scale common item parameters, and the $W_1\left(\theta_{mT}\right)$ are a set of quadrature weights corresponding to θ_{mT}. The θ_{mF} are a set of points on the *from* scale where differences in expected probabilities are evaluated, $P_{sk}\left(\theta_{mF}\right)$ are expected probabilities based on the untransformed *from* scale common item parameters, $P^{\#}_{sk}\left(\theta_{mF}\right)$ are expected probabilities based on the transformed *to* scale common item parameters, and the $W_2\left(\theta_{mF}\right)$ are a set of quadrature weights corresponding to θ_{mF}. L_1 and L_2 are constants used to standardize the criterion function to prevent the value from exceeding upper or lower bounds in the optimization. The inclusion of L_1 and L_2 does not affect the estimated linking constants (Kim and Lee 2006). Q is minimized in the symmetric approach, but only $Q1$ is minimized in the non-symmetric approach.

The Stocking-Lord method minimizes the following criterion

$$F = F_1 + F_2 \tag{15a}$$

where

$$F_1 = \frac{1}{L^{*}_1} \sum_{m=1}^{M} \left[\sum_{s=1}^{S} \sum_{k=1}^{K_s} U_{sk} P_{sk}\left(\theta_{mT}\right) - \sum_{s=1}^{S} \sum_{k=1}^{K_s} U_{sk} P^{*}_{sk}\left(\theta_{mT}\right) \right]^2 W_1\left(\theta_{mT}\right) \tag{15b}$$

and

$$F_2 = \frac{1}{L_2^*} \sum_{m=1}^{M} \left[\sum_{s=1}^{S} \sum_{k=1}^{K_s} U_{sk} P_{sk}\left(\theta_{mF}\right) - \sum_{s=1}^{S} \sum_{k=1}^{K_s} U_{sk} P_{sk}^{\#}\left(\theta_{mF}\right) \right]^2 W_2\left(\theta_{mF}\right) \tag{15c}$$

where

$$L_1^* = \sum_{m=1}^{M} W_1\left(\theta_{mT}\right) \quad \text{and} \quad L_2^* = \sum_{m=1}^{M} W_2\left(\theta_{mF}\right).$$

To create the test characteristic curves, the scoring function U_{sk} must be included to weight each response category. These values are typically specified as $U_{sk} = \{0, ..., K_s - 1\}$, which assumes that the categories are ordered. F is minimized in the symmetric approach, but only $F1$ is minimized in the non-symmetric approach.

3. Preparing the data

There are four necessary elements that must be created to prepare the data prior to linking a set of tests using the function `plink`:

1. an object containing the item parameters,
2. an object specifying the number of response categories for each item,
3. an object identifying the item response models associated with each item,
4. an object identifying the common items between groups.

In short, these elements create a blueprint of the unique and common items across two or more tests. The following section describes how to specify the first three elements for a single set of item parameters then shows how the elements for two or more groups can be combined—incorporating the common item object—for use in `plink`. The section concludes with a discussion of methods for importing data from commonly used IRT estimation software. If the parameters are imported from one of the software packages identified in Section 3.5, no additional formatting is required (i.e., Sections 3.1, 3.2, and 3.3 can be skipped).

3.1. Formatting the item parameters

The key elements of any IRT-based linking method—using a common item design—are the item parameters, but depending on the program used to estimate them, they may come in a variety of formats. **plink** is designed to allow a fair amount of flexibility in how the parameters are specified to minimize the need for reformatting. This object, named `x` in all related functions, can be specified for single-format or mixed-format items as a vector, matrix, or list.

Vector formulation

When the Rasch model is used, `x` can be formatted as a vector of item difficulties, but for all other models a matrix or list specification must be used (the Rasch model can be specified using these formulations as well).

Matrix formulation

The general format for structuring `x` as a matrix can be thought of as an additive column approach. The object should be an $N \times R$ matrix for N items and R equal to the number of parameters for the item with the most parameters. The left-most columns are typically for discrimination/slope parameters, the next column (if applicable) is for location parameters, the next set of columns is for difficulty/threshold/step/category parameters, and the final set of columns is for lower asymptote (guessing) parameters.

For dichotomous items, `x` can include one, two, or three columns (see formulation (16)). The item response model for these items is not explicitly specified; rather, it is determined based on the included item parameters. For instance, instead of formatting `x` as a vector for the Rasch model, an $N \times 1$ matrix of item difficulties can be supplied. An $N \times 2$ matrix can also be used with all of the values in the first column equal to one and difficulty parameters in the second column. For discrimination values other than one—for the 1PL—`x` should include at least two columns where the discrimination parameters are identical for all items. In all of these cases the lower asymptote values will default to zero; however, three columns can be specified where the values in the last column all equal zero. For the 2PL, `x` should include at least two columns for the discrimination and difficulty parameters respectively. As with the 1PL, the lower asymptote values will default to zero, but a third column of zeros can be included. For the 3PL, `x` should include all three columns.

$$\begin{bmatrix} a_1 \\ \cdot \\ a_j \end{bmatrix} \begin{bmatrix} b_1 \\ \cdot \\ b_j \end{bmatrix} \begin{bmatrix} c_1 \\ \cdot \\ c_j \end{bmatrix} \tag{16}$$

For GRM, PCM, and GPCM items, `x` may include up to three blocks of parameters (see formulation (17)). If no location parameter is included, the first column will contain the slopes and the remaining columns will include the threshold/step parameters. When a location parameter is included, the first column will be for the slopes, the second column for the location parameters, and the remaining columns for the threshold/step deviation parameters. For the PCM, if the slope is equal to one, the column of slope parameters is not required; otherwise, this column needs to be included.

$$\begin{bmatrix} a_1 \\ \cdot \\ \cdot \\ a_j \end{bmatrix} \begin{bmatrix} b_{11} & \cdot & \cdot & b_{1k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ b_{j1} & \cdot & \cdot & b_{jk} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} a_1 \\ \cdot \\ \cdot \\ a_j \end{bmatrix} \begin{bmatrix} b_1 \\ \cdot \\ \cdot \\ b_j \end{bmatrix} \begin{bmatrix} g_{11} & \cdot & \cdot & g_{1k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ g_{j1} & \cdot & \cdot & g_{jk} \end{bmatrix} \tag{17}$$

For the nominal response model, `x` should include two blocks of parameters (see formulation (18)). The first k columns are for the slopes (ordered in the same manner as the category parameters) and the next k columns are for the category parameters. One nuance of this formulation is the placement of missing values when items have different numbers of response categories. When extracting NRM parameters from the entire matrix of item parameters, the k columns of slopes are treated as a single block, meaning all of the category parameters must begin in column $k + 1$. Therefore, missing values should appear at the end of a given row within the block of slopes or category parameters. Visually, it will seem as if there is a

gap in the middle of the row (see formulation (20)).

$$\begin{bmatrix} a_{11} & \cdot & \cdot & a_{1k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ a_{j1} & \cdot & \cdot & a_{jk} \end{bmatrix} \begin{bmatrix} b_{11} & \cdot & \cdot & b_{1k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ b_{j1} & \cdot & \cdot & b_{jk} \end{bmatrix} \tag{18}$$

The specification of item parameters for the multiple-choice model is very similar to the specification for NRM items with the only difference being the addition of a block of lower asymptote parameters (see formulation (19)). The same placement of missing values applies here as well.

$$\begin{bmatrix} a_{11} & \cdot & \cdot & a_{1k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ a_{j1} & \cdot & \cdot & a_{jk} \end{bmatrix} \begin{bmatrix} b_{11} & \cdot & \cdot & b_{1k} \\ \cdot & \cdot & & \cdot \\ \cdot & & \cdot & \cdot \\ b_{j1} & \cdot & \cdot & b_{jk} \end{bmatrix} \begin{bmatrix} c_{11} & \cdot & c_{1k-1} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ c_{j1} & \cdot & c_{jk-1} \end{bmatrix} \tag{19}$$

As an illustration of how to format the item parameters for a mixed-format test, say we have nine items: four 3PL items (items 1, 4, 5, and 8), three GRM items with location parameters (items 2, 3, and 9 with 3, 5, and 3 categories respectively), and two NRM items (items 6 and 7 with 4 and 5 categories respectively). The matrix should be formatted as follows where all of the blank spaces would contain `NA`s.

$$\begin{bmatrix} a_1 & b_1 & c_1 & & & & & & & \\ a_2 & b_2 & g_{21} & g_{22} & & & & & & \\ a_3 & b_3 & g_{31} & g_{32} & g_{33} & g_{34} & & & & \\ a_4 & b_4 & c_4 & & & & & & & \\ a_5 & b_5 & c_5 & & & & & & & \\ a_{61} & a_{62} & a_{63} & a_{64} & & b_{61} & b_{62} & b_{63} & b_{64} & \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & b_{71} & b_{72} & b_{73} & b_{74} & b_{75} \\ a_8 & b_8 & c_8 & & & & & & & \\ a_9 & b_9 & g_{91} & g_{92} & & & & & & \end{bmatrix} \tag{20}$$

Using actual values, the matrix `x` would look like this

```
     [,1]  [,2]  [,3]  [,4]  [,5] [,6]  [,7]  [,8]  [,9] [,10]
 [1,] 0.84 -1.63  0.25    NA    NA   NA    NA    NA    NA    NA
 [2,] 1.22 -0.47 -0.83  0.83    NA   NA    NA    NA    NA    NA
 [3,] 1.10 -0.04 -1.40 -0.28  0.54 1.15    NA    NA    NA    NA
 [4,] 1.08  0.84  0.16    NA    NA   NA    NA    NA    NA    NA
 [5,] 0.97 -0.14  0.14    NA    NA   NA    NA    NA    NA    NA
 [6,] 0.90  0.52 -0.47 -0.96    NA 0.13 -0.21 -0.26  0.34    NA
 [7,] 0.83  0.38 -0.36 -0.08 -0.82 0.56  0.86 -1.19 -1.20  0.99
 [8,] 1.13  2.03  0.02    NA    NA   NA    NA    NA    NA    NA
 [9,] 0.87  1.46 -0.28  0.28    NA   NA    NA    NA    NA    NA
```

List formulation

The creation of `x` as a list is similar to the matrix formulation in that it is an additive element approach. The list can contain one, two, or three elements. The first element is typically for discrimination/slope parameters, the second element is for location parameters (if applicable) and difficulty/threshold/step/category parameters, and the third element is for lower asymptote parameters. However, the number of elements may vary depending on the item response model(s). Within each list element, the parameters should be formatted as a vector or matrix. The combination of multiple models is equivalent to formatting each type of item parameter for each response model separately, stacking the matrices on top of one another—filling in any missing cells with `NA`s if necessary—then combining these elements into a list (see the documentation included in **plink** for more information). Below is an illustration of how the item parameters above would look using the list formulation.

```
$a
      [,1] [,2]  [,3]  [,4]  [,5]
 [1,] 0.84   NA    NA    NA    NA
 [2,] 1.22   NA    NA    NA    NA
 [3,] 1.10   NA    NA    NA    NA
 [4,] 1.08   NA    NA    NA    NA
 [5,] 0.97   NA    NA    NA    NA
 [6,] 0.90 0.52 -0.47 -0.96    NA
 [7,] 0.83 0.38 -0.36 -0.08 -0.82
 [8,] 1.13   NA    NA    NA    NA
 [9,] 0.87   NA    NA    NA    NA

$b
       [,1]  [,2]  [,3]  [,4] [,5]
 [1,] -1.63    NA    NA    NA   NA
 [2,] -0.47 -0.83  0.83    NA   NA
 [3,] -0.04 -1.40 -0.28  0.54 1.15
 [4,]  0.84    NA    NA    NA   NA
 [5,] -0.14    NA    NA    NA   NA
 [6,]  0.13 -0.21 -0.26  0.34   NA
 [7,]  0.56  0.86 -1.19 -1.20 0.99
 [8,]  2.03    NA    NA    NA   NA
 [9,]  1.46 -0.28  0.28    NA   NA

$c
[1] 0.25   NA   NA 0.16 0.14   NA   NA 0.02   NA
```

3.2. Specifying response categories

Since the item parameters can be formatted in different ways, particularly for polytomous items, it is necessary to identify the number of response categories for each item. This is by far the simplest object to create. In the functions that use it, the argument is named `cat`. For a single set of item parameters, `cat` is a vector. The values for dichotomous items will always

equal 2 while the values for polytomous items will vary depending on the number of response categories. The values for items corresponding to the multiple-choice model should equal the number of response categories plus one—for the "do not know" category. For instance, `cat` would equal five for an MCM item with four actual responses. The ordering of values in `cat` should coincide with the order of item parameters in `x`. To create this object for the set of items in Section 3.1, `cat` can be specified as

```
R> cat <- c(2, 3, 5, 2, 2, 4, 5, 2, 3)
```

3.3. Specifying item response models

The third required element is an object that identifies the item response model used for each item. This is known as a `poly.mod` object. It is created using the function `as.poly.mod`. The function has three arguments:

`n`: The total number of items.

`model`: A character vector identifying the IRT models used to estimate the item parameters.

`items`: A list identifying the item numbers (i.e., the rows in `x`) corresponding to the given model in `model`.

The `model` argument can include the following elements: `"drm"` for dichotomous response models (i.e., Rasch, 1PL, 2PL, or 3PL), `"grm"` for the graded response model, `"gpcm"` for the partial credit/generalized partial credit model, `"nrm"` for the nominal response model, and `"mcm"` for the multiple-choice model.

When all of the items are dichotomous, only `n` is needed. If all of the items correspond to a single polytomous model, only the first two arguments are needed, but if two or more item response models are used, all three arguments are required. For example, the `poly.mod` object for the items in Section 3.1 can be created as

```
R> pm <- as.poly.mod(9, c("drm", "grm", "nrm"),
+    list(c(1, 4, 5, 8), c(2, 3, 9), 6:7))
```

The order of elements in `model` is not important, but the order of the list elements in `items` does matter. The list elements must correspond to the order of the elements in `model`. As such, the `poly.mod` object could be respecified as

```
R> pm <- as.poly.mod(9, c("grm", "drm", "nrm"),
+    list(c(2, 3, 9), c(1, 4, 5, 8), 6:7))
```

3.4. Combining elements and identifying common items

The three elements described above (`x`, `cat`, and `poly.mod`) characterize the items for a single test, but in the context of test linking we will necessarily have two or more sets of item parameters (these objects must be created for each test). As an alternative to keeping track of several objects for each test, we can combine these elements into a single object using the

`as.irt.pars` function (this type of object is also required for `plink`). This function creates an `irt.pars` object that characterizes the items for one or more tests and includes built-in validation checks to ensure that there are no obvious incongruences between the parameters, response categories, and item response models. The `as.irt.pars` function has the following arguments:

`x`: An object containing item parameters. When multiple groups are present, `x` should be a list of parameter objects (a combination of objects using the vector, matrix, and/or list formulation).

`common`: An $S \times 2$ matrix or list of matrices identifying the common items between adjacent groups in `x`. This argument is only applicable when `x` includes two or more groups.

`cat`: A vector or list of vectors (for two or more groups) identifying the number of response categories.

`poly.mod`: A `poly.mod` object or list of `poly.mod` objects (for two or more groups).

`dimensions`: A numeric vector identifying the number of modeled dimensions in each group. The default is `1`.

`location`: A logical vector indicating whether the parameters for each group in `x` include a location parameter. The default is `FALSE`.

`grp.names`: A character vector of group names.

To create an `irt.pars` object for the single set of parameters specified in Section 3.1 we would use

```
R> pars <- as.irt.pars(x, cat = cat, poly.mod = pm, location = TRUE)
```

where `x`, `cat`, and `pm` are the objects created in Sections 3.1, 3.2 and 3.3 respectively. When creating an `irt.pars` object that includes item attributes for multiple tests, it is assumed that there is a set of common items between each "adjacent" test (i.e., adjacent list elements in `x`). Hence, it is necessary to create an object, `common`, that identifies these common items. `common` should be formatted as an $S_{xy} \times 2$ matrix (for two tests), or a list of matrices (for more than two tests), for S common items between each pair of adjacent tests x and y. The values in a given matrix are the row numbers corresponding to the rows in the matrix/list of item parameters for the two paired tests. For example, say we have two tests, D and E, with 60 items each where the last 10 items on test D are the same as the first 10 items on test E. `common` would be created as

```
R> common <- matrix(c(51:60, 1:10), 10, 2)
R> common
```

```
      [,1] [,2]
 [1,]   51    1
 [2,]   52    2
 [3,]   53    3
```

```
 [4,]   54    4
 [5,]   55    5
 [6,]   56    6
 [7,]   57    7
 [8,]   58    8
 [9,]   59    9
[10,]   60   10
```

In words, this means that item 51 on test D is the same as item 1 on test E, and so on. The ordering of items—rows—in this matrix is not important (i.e., the values do not need to be in, say, ascending order).

Now that all of the necessary objects have been created, they can be can combined in a single `irt.pars` object. This is accomplished in one of two ways: the objects created above (`x`, `cat`, `poly.mod`) for each test can be combined with the `common` object by running `as.irt.pars` directly, or `irt.pars` objects can be created for each test first and then combined with `common` using the `combine.pars` function. Using the first approach, if we have three tests D, E, and F with corresponding objects `x.D`, `x.E`, `x.F`, `cat.D`, `cat.E`, `cat.F`, `poly.mod.D`, `poly.mod.E`, `poly.mod.F`, `common.DE`, and `common.EF`, the `irt.pars` object would be created as follows

```
R> pars <- as.irt.pars(x = list(x.D, x.E, x.F),
+    common = list(common.DE, common.EF), cat = list(cat.D, cat.E, cat.F),
+    poly.mod = list(poly.mod.D, poly.mod.E, poly.mod.F))
```

The item parameter objects, response category vectors, `poly.mod` objects, and the common item matrices are combined as a list for each type of object separately then passed to the function.

For the second approach, the `combine.pars` function can be used to create an `irt.pars` object for multiple groups. Say we originally created an `irt.pars` object `pars.DE` by combining the information for tests D and E then later created an `irt.pars` object `pars.F` for a single test F. We can combine these two objects into a single object using `common.EF`.

```
R> pars <- combine.pars(x = list(pars.DE, pars.F), common = common.EF)
```

3.5. Importing parameters from IRT software

In certain cases the item parameters will come in a format that necessitates the creation of `x`, `cat`, and `poly.mod` and subsequently the creation of an `irt.pars` object, but if the parameters are estimated using common IRT software, they can be imported as an `irt.pars` object without having to create any of these objects directly. **plink** includes functions to import item parameters (and ability estimates) from **BILOG-MG** (Zimowski, Muraki, Mislevy, and Bock 2003), **PARSCALE** (Muraki and Bock 2003), **MULTILOG** (Thissen 2003), **ICL** (Hanson 2002), and the R packages **eRm** (Mair and Hatzinger 2007) and **ltm** (Rizopoulos 2006).[2] These functions are named `read.bilog`, `read.parscale`, `read.multilog`, `read.icl`, `read.erm`, and `read.ltm` respectively. They include four principal arguments:

[2]Multidimensional parameters can be imported from **TESTFACT** (Wood, Wilson, Muraki, Schilling, Gibbons, and Bock 2003) and **BMIRT** (Yao 2008).

`file`: The filename of the file containing the item or ability parameters.

`ability`: A logical value indicating whether the file contains ability parameters. The default is `FALSE`.

`loc.out`: A logical value indicating whether threshold/step parameters should be formatted as deviations from a location parameter (not applicable for `read.bilog`). The default is `FALSE`.

`as.irt.pars`: A logical value indicating whether the item parameters should be imported as an `irt.pars` object. The default is `TRUE`.

In addition to the four arguments above, there are other function-specific arguments. For instance, with `read.erm` and `read.ltm`, there is no `file` argument because the output is created in R. The main argument in these functions, `x`, is the output object from one of the following functions in **eRm**: `RM`, `RSM`, `PCM`, `LLTM`, `RSM`, or `PCM`; or from **ltm**:[3] `rasch`, `ltm`, `tpm`, `grm`, or `gpcm`. For the `read.icl` function, a `poly.mod` object must be created because no information about the item type is included in the `.par` file, and for the functions `read.bilog` and `read.parscale` a logical argument, `pars.only`, can be included to indicate whether information like standard errors should be included with the returned parameters.

Relative to the other applications, **MULTILOG** has the greatest flexibility for specifying item response models, yet the information in the `.par` file provides minimal information about the model(s) and the associated constraints (the `.par` file only includes contrast parameters). As such, for `read.multilog`, it is necessary to create both a `cat` and `poly.mod` object, and depending on the specified model(s), it may be necessary to include the arguments `drm.3PL` and/or `contrast`. `drm.3PL` is a logical argument indicating whether the 3PL was used to model the dichotomous items (the default is `TRUE`). The `contrast` argument is a bit more complex. With the exception of the 1PL, 2PL, and GRM, all of the models in **MULTILOG** are constrained versions of the multiple-choice model where various contrast parameters are estimated (see Thissen and Steinberg 1986). These can include deviation, polynomial, or triangular contrasts for individual parameters on specific items. The `contrast` argument is used to identify these constraints. A full explanation of this argument, in addition to information on importing parameters from the other software packages, is included in the documentation in **plink**.

4. Running the calibration

Once an `irt.pars` object with two or more tests has been created, the function `plink` can be used to estimate linking constants and (if desired) transform all of the item and/or ability parameters onto a base scale. The function includes one essential argument `x`, and twelve optional arguments.[4] These arguments are presented in the context of several examples.

`x`: An `irt.pars` object with two or more groups.

[3] **plink** and **ltm** both have functions named `grm` and `gpcm`. With both packages running, it may be necessary to call the appropriate function using `plink::grm`, `plink::gpcm`, `ltm::grm`, or `ltm::gpcm`

[4] There are two additional arguments, `dilation` and `dim.order`, that only pertain to multidimensional linking methods.

`rescale`: A character value identifying the linking constants to use to transform the parameters to the base scale. Applicable values are "MS", "MM", "HB", and "SL" for the mean/sigma, mean/mean, Haebara, and Stocking-Lord methods respectively.

`ability`: A list of θ values where the number of list elements equals the number of groups in `x`.

`method`: A character vector identifying the method(s) to use when estimating the linking constants. Applicable values are "MS", "MM", "HB", and "SL". If missing, linking constants will be estimated using all four methods.

`weights.t`: A list containing quadrature points and weights on the *to* scale for use with the characteristic curve methods.

`weights.f`: A list containing quadrature points and weights on the *from* scale for use with the characteristic curve methods. This argument will be ignored if `symmetric` = `FALSE`.

`startvals`: A vector of starting values for A and B respectively for use in the characteristic curve methods or a character value equal to "MS" or "MM" indicating that estimates from the given moment method should be used. If the argument is missing, values from the mean/sigma method are used.

`exclude`: A character vector or list identifying common items that should be excluded when estimating the linking constants.

`score`: An integeridentifying the scoring function to use for the Stocking-Lord method. When `score = 1`, the ordered categories for each item are scored from 0 to $k - 1$, and when `score = 2`, the categories are scored from 1 to k. The default is `1`. A vector of scores for each response category can also be supplied, but this is only recommended for advanced users.

`base.grp`: An integer identifying the reference scale—base group—onto which all item and ability parameters should be placed. The default is `1`.

`symmetric`: A logical value indicating whether symmetric optimization should be used for the characteristic curve methods. The default is `FALSE`.

`rescale.com`: A logical value. If `TRUE`, rescale the common item parameters using the estimated linking constants; otherwise, insert the non-transformed common item parameters into the set of unique transformed item parameters. The default is `TRUE`.

`grp.names`: A character vector of names for each group in `x` (i.e., names for each test). If group names are identified when creating the `irt.pars` object, this argument is unnecessary.

`...`: Further arguments passed to other methods.

4.1. Two groups, dichotomous data

The simplest linking scenario is the case where there are only two tests and all of the items (unique and common) are dichotomously scored. This example uses the `KB04` dataset which

reproduces the data presented by Kolen and Brennan (2004) in Table 6.5 (p. 192). There are 36 items on each test with 12 common items between them. The `KB04` dataset is formatted as a list with two elements. The first element is a list of length two containing the item parameters for "new" and "old" forms respectively, and the second element is a matrix identifying the common items between the two tests. These elements correspond to the objects `x` and `common`, but `cat` and `poly.mod` still need to be created. The following code is used to create these objects and the combined `irt.pars` object

```
R> cat <- rep(2, 36)
R> pm <- as.poly.mod(36)
R> x <- as.irt.pars(KB04$pars, KB04$common, cat = list(cat, cat),
+    poly.mod = list(pm, pm), grp.names = c("new", "old"))
```

Once this object is created, `plink` can be run without specifying any additional arguments.

```
R> out <- plink(x)
R> summary(out)

-------  old/new*  -------
Linking Constants

                    A        B
Mean/Mean     0.821513 0.457711
Mean/Sigma    0.855511 0.441053
Haebara       0.907748 0.408260
Stocking-Lord 0.900655 0.425961
```

There are two things to notice in this output. First, no `method` argument was specified, so linking constants were estimated using all four approaches, and second, there is an asterisk included next to the group name "new" indicating that this is the base group (this will be of particular importance in the examples with more than two groups). Although not obvious in this example, no rescaled parameters are returned. To return the rescaled item parameters, the `rescale` argument must be included. More than one method can be used to estimate the linking constants, but parameters can only be rescaled using a single approach, meaning only one method can be specified for `rescale`.

In the following example, the parameters are rescaled using the Stocking-Lord method with the "old" form (i.e., the second set of parameters in `x`) treated as the base scale.

```
R> out <- plink(x, rescale = "SL", base.grp = 2)
R> summary(out)

-------  new/old*  -------
Linking Constants

                    A         B
Mean/Mean     1.217266 -0.557155
Mean/Sigma    1.168892 -0.515543
Haebara       1.112133 -0.465667
Stocking-Lord 1.119231 -0.485545
```

The function `link.pars` can be used to extract the rescaled parameters (the only argument is the output object from `plink`). These parameters are returned as an `irt.pars` object. Similarly, the function `link.con` can be used to extract the linking constants.

To illustrate the use of additional arguments, the estimation is respecified using a symmetric approach with 30 standard normal quadrature points and weights (created using the `as.weight` function). A set of ability estimates is also included. To keep it simple, the abilities for both groups are the same and range from -4 to 4 logits. That is,

```
R> ability <- list(group1 = -4:4, group2 = -4:4)
```

The respecification is implemented as follows

```
R> out <- plink(x, rescale = "SL", ability = ability, base.grp = 2,
+    weights.t = as.weight(30, normal.wt = TRUE), symmetric = TRUE)
R> summary(out)

-------  new/old*  -------
Linking Constants

                     A         B
Mean/Mean     1.217266 -0.557155
Mean/Sigma    1.168892 -0.515543
Haebara       1.088233 -0.504619
Stocking-Lord 1.109529 -0.519901

Ability Descriptive Statistics

         new     old
Mean -0.5199  0.0000
SD    3.0386  2.7386
Min  -4.9580 -4.0000
Max   3.9182  4.0000
```

The most obvious difference in this output is the inclusion of summary statistics for the rescaled ability estimates, but the differences in estimated constants for the characteristic curve methods relative to the previous estimation should also be noted. As with rescaled item parameters, the function `link.ability` can be used to extract the rescaled ability parameters.

```
R> link.ability(out)

$new
[1] -4.9580 -3.8485 -2.7390 -1.6294 -0.5199  0.5896  1.6992  2.8087
[9]  3.9182

$old
[1] -4 -3 -2 -1  0  1  2  3  4
```

4.2. Two groups, mixed-format data

The next set of examples illustrate how two tests with mixed-format items can be linked using `plink`. These examples use the `dgn` dataset which includes 55 items on two tests modeled using the 3PL, generalized partial credit model, and nominal response model. `dgn` is a list that includes four elements. The first element is a list of item parameters, the second is a list of numbers of response categories, the third is a list of lists that identifies the items associated with each response model, and the final element is a matrix identifying the common items between tests. The `irt.pars` object can be created as follows

```
R> pm1 <- as.poly.mod(55, c("drm", "gpcm", "nrm"), dgn$items$group1)
R> pm2 <- as.poly.mod(55, c("drm", "gpcm", "nrm"), dgn$items$group2)
R> x <- as.irt.pars(dgn$pars, dgn$common, dgn$cat, list(pm1, pm2))
```

Let us start by running `plink` without any additional arguments.

```
R> out <- plink(x)
R> summary(out)

-------  group2/group1*  -------
Linking Constants

                     A        B
Mean/Mean     1.069847 0.076175
Mean/Sigma    0.981779 0.065953
Haebara       1.071950 0.168550
Stocking-Lord 1.080143 0.161408
```

Notice that the B constants for the moment methods are quite different from those for the characteristic curve methods. This is likely due to the inclusion of NRM common items. To illustrate how the linking constants change when the NRM common items are excluded, the `exclude` argument is used. Descriptive statistics for the common item parameters are also displayed when summarizing the output.

```
R> out1 <- plink(x, exclude = "nrm")
R> summary(out1, descrip = TRUE)

-------  group2/group1*  -------
Linking Constants

                     A        B
Mean/Mean     1.076746 0.174981
Mean/Sigma    1.076365 0.174879
Haebara       1.070915 0.167160
Stocking-Lord 1.082034 0.158226

Common Item Descriptive Statistics
```

```
Model: 3PL
Number of Items: 7

                  a         b        c
N Pars:    7.000000  7.000000 7.000000
Mean: To   0.955429 -0.304286 0.192286
Mean: From 1.028714 -0.443857 0.188857
SD: To     0.091653  1.025097 0.065622
SD: From   0.098714  0.952583 0.076230

Model: Generalized Partial Credit Model
Number of Items: 3

                  a         b
N Pars:    3.000000 12.000000
Mean: To   1.097667 -0.002167
Mean: From 1.182000 -0.165250
SD: To     0.094813  1.114006
SD: From   0.101853  1.035063

Model: All
Number of Items: 10

                   a         b        c
N Pars:    10.000000 19.000000 7.000000
Mean: To    0.998100 -0.113474 0.192286
Mean: From  1.074700 -0.267895 0.188857
SD: To      0.113251  1.091870 0.065622
SD: From    0.121933  1.014405 0.076230
```

4.3. Six groups, mixed-format data

In the final example, the `reading` dataset is used to illustrate how the parameters from multiple mixed-format tests can be chain-linked together using `plink`. For these data there are six tests that span four grades and three years (see Table 1). The adjacent groups follow a stair-step pattern (e.g., the grade 3 and grade 4 tests in year 0 are linked then the grade 4 tests in years 0 and 1 are linked, etc.) As with `dgn`, the object `reading` includes most of the elements needed to create the `irt.pars` object, but it is still necessary to create the `poly.mod` objects for each test. The following code is used for this purpose.

```
R> pm1 <- as.poly.mod(41, c("drm", "gpcm"), reading$items[[1]])
R> pm2 <- as.poly.mod(70, c("drm", "gpcm"), reading$items[[2]])
R> pm3 <- as.poly.mod(70, c("drm", "gpcm"), reading$items[[3]])
R> pm4 <- as.poly.mod(70, c("drm", "gpcm"), reading$items[[4]])
R> pm5 <- as.poly.mod(72, c("drm", "gpcm"), reading$items[[5]])
R> pm6 <- as.poly.mod(71, c("drm", "gpcm"), reading$items[[6]])
R> pm <- list(pm1, pm2, pm3, pm4, pm5, pm6)
```

	Year 0	Year 1	Year 2
Grade 3	X		
Grade 4	X	X	
Grade 5		X	X
Grade 6			X

Table 1: Linking design.

Next, the `irt.pars` object can be compiled. To distinguish between the groups in the output, a set of group names is specified. The number before the decimal in each name is the grade, and the value after the decimal corresponds to the year.

```
R> grp.names <- c("Grade 3.0", "Grade 4.0", "Grade 4.1", "Grade 5.1",
+    "Grade 5.2", "Grade 6.2")
R> x <- as.irt.pars(reading$pars, reading$common, reading$cat, pm,
+    grp.names = grp.names)
```

For this example, only the characteristic curve methods are used to estimate the linking constants (using the `method` argument) and the grade 5, year 1 test is treated as the base group.

```
R> out <- plink(x, method = c("HB", "SL"), base.grp = 4)
R> summary(out)

-------  Grade 3.0/Grade 4.0  -------
Linking Constants

                     A         B
Haebara       1.020173 -0.470805
Stocking-Lord 1.043527 -0.411081

-------  Grade 4.0/Grade 4.1  -------
Linking Constants

                     A         B
Haebara       0.985305 -0.099593
Stocking-Lord 1.016654 -0.088504

-------  Grade 4.1/Grade 5.1*  -------
Linking Constants

                     A         B
Haebara       1.102458 -0.445897
Stocking-Lord 1.090150 -0.421204

-------  Grade 5.2/Grade 5.1*  -------
```

```
Linking Constants

                     A         B
Haebara       1.057478 -0.065181
Stocking-Lord 1.037606 -0.070660

-------  Grade 6.2/Grade 5.2  -------
Linking Constants

                     A        B
Haebara       1.056546 0.239984
Stocking-Lord 1.058446 0.211559
```

Notice the ordering of the group labels in the summary output, as well as the inclusion of the asterisk. For the first three pairs of adjacent tests, the labels in the header indicate that the associated linking constants can be used to place the parameters for the lower group (relative to the ordering of groups in `x`) onto the scale of the higher group. At this point we get to the base group and the order of the labels in the headers changes. They now indicate that the associated linking constants can be used to place the parameters for the higher group onto the scale of the lower group.

In all of these examples, the specification of the `plink` function remains essentially unchanged regardless of the number of groups or the combination of item response models. As such, most of the work in linking a set of tests is tied to the creation of the `irt.pars` object. After that, it is simply a matter of deciding which optional arguments (if any) to include.

References

Andrich D (1978). "A Rating Formulation for Ordered Response Categories." *Psychometrika*, **43**(4), 561–573.

Baker FB (1992). "Equating Tests under the Graded Response Model." *Applied Psychological Measurement*, **16**(1), 87–96.

Baker FB (1993a). "Equating Tests under the Nominal Response Model." *Applied Psychological Measurement*, **17**(3), 239–251.

Baker FB (1993b). ***EQUATE**: A Computer Program for the Characteristic Curve Method of IRT Equating*. University of Wisconsin, Madison. Version 1.0.

Birnbaum A (1968). "Some Latent Trait Models and Their Use in Inferring an Examinee's Ability." In FM Lord, MR Novick (eds.), "Statistical Theories of Mental Test Scores," pp. 397–479. Addison-Wesley, Reading, MA.

Bock RD (1972). "Estimating Item Parameters and Latent Ability when Responses are Scored in Two or More Nominal Categories." *Psychometrika*, **37**(1), 29–51.

Doran HC (2010). ***MiscPsycho**: Miscellaneous Psychometrics*. R package version 1.6, URL http://CRAN.R-project.org/package=MiscPsycho.

Gulliksen H (1950). *Theory of Mental Tests*. John Wiley & Sons, New York.

Haebara T (1980). "Equating Logistic Ability Scales by a Weighted Least Squares Method." *Japanese Psychological Research*, **22**(3), 144–149.

Han KT (2007). ***IRTEQ**: Windows Application that Implements IRT Scaling and Equating*. University of Massachusetts, Amherst. Version 1.0, URL http://www.umass.edu/remp/software/irteq/.

Han KT, Hambelton RK (2007). ***WinGen2**: Windows Software that Generates IRT Model Parameters and Item Responses*. University of Massachusetts, Amherst. Version 1.0, URL http://www.umass.edu/remp/software/wingen/.

Hanson BA (2000). ***mcmequate**: Equating Software for the Multiple-Choice Model*. Version 1.0, URL http://www.b-a-h.com/software/mcmequate/index.html.

Hanson BA (2002). ***IRT Command Language***. Version 1.0, URL http://www.b-a-h.com/software/irt/icl/.

Hanson BA, Zeng L (1995). ***ST**: A Computer Program for IRT Scale Transformation*. University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Hull CL (1922). "The Conversion of Test Scores Into Series Which Shall Have Any Assigned Mean and Degree of Dispersion." *Journal of Applied Psychology*, **6**(4), 298–300.

Kelley TL (1923). *Statistical Method*. Macmillan, New York.

Kim J, Hanson BA (2002). "Test Equating under the Multiple-Choice Model." *Applied Psychological Measurement*, **26**(3), 255–270.

Kim S, Kolen MJ (2003). ***POLYST**: A Computer Program for Polytomous IRT Scale Transformation*. University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Kim S, Kolen MJ (2004). ***STUIRT**: A Computer Program for Scale Transformation under Unidimensional Item Response Theory Models*. University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Kim S, Lee W (2006). "An Extension of Four IRT Linking Methods for Mixed-Format Tests." *Journal of Educational Measurement*, **43**(1), 53–76.

Kolen MJ (1981). "Comparison of Traditional and Item Response Theory Methods for Equating Tests." *Journal of Educational Measurement*, **18**(1), 1–11.

Kolen MJ (2004). ***POLYEQUATE***. University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Kolen MJ, Brennan RL (2004). *Test Equating, Scaling, and Linking: Methods and Practices*. Springer-Verlag, New York, 2nd edition.

Lee K, Oshima TC (1997). ***Iplink**: Item Parameter Linking*. Version 2.0, URL http://education.gsu.edu/eps/4493.html.

Levine RE (1955). "Equating the Score Scales of Alternative Forms Administered to Samples of Different Ability." *Research Bulletin 55-23*, Educational Testing Services, Princeton, NJ.

Linn RL (1993). "Linking Results of Distinct Assessments." *Applied Measurement in Education*, **6**(1), 83–102.

Linn RL, Levine MV, Hastings CN, Wardrop JL (1980). "An Investigation of Item Bias in a Test of Reading Comprehension." *Technical Report 163*, University of Illinois, Center for the Study of Reading, Urbana, IL.

Lord FM (1952). "A Theory of Test Scores." *Psychometric Monographs*. No. 7.

Lord FM (1980). *Applications of Item Response Theory to Practical Testing Problems*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Lord FM, Novick MR (1968). *Statistical Theories of Mental Test Scores*. Addison-Wesley, Reading, MA.

Loyd BH, Hoover HD (1980). "Vertical Equating Using the Rasch Model." *Journal of Educational Measurement*, **17**(3), 179–193.

Mair P, Hatzinger R (2007). "Extended Rasch Modeling: The **eRm** Package for the Application of IRT Models in R." *Journal of Statistical Software*, **20**(9), 1–20. URL http://www.jstatsoft.org/v20/i09/.

Marco GL (1977). "Item Characteristic Curve Solutions to Three Intractable Testing Problems." *Journal of Educational Measurement*, **14**(2), 139–160.

Masters GN (1982). "A Rasch Model for Partial Credit Scoring." *Psychometrika*, **47**(2), 149–174.

Muraki E (1992). "A Generalized Partial Credit Model: Application of an EM Algorithm." *Applied Psychological Measurement*, **16**(2), 159–176.

Muraki E, Bock RD (2003). ***PARSCALE** 4: IRT Item Analysis and Test Scoring for Rating-Scale Data*. Scientific Software International, Inc., Lincolnwood, IL. Version 4.0, URL http://www.ssicentral.com/.

Partchev I (2009). ***irtoys**: Simple Interface to the Estimation and Plotting of IRT Models*. R package version 0.1.2, URL http://CRAN.R-project.org/package=irtoys.

Petersen NS, Cook LL, Stocking ML (1983). "IRT Versus Conventional Equating Methods: A Comparative Study of Scale Stability." *Journal of Educational Statistics*, **8**(2), 137–156.

Rasch G (1960). *Probabilistic Models for Some Intelligence and Attainment Tests*. Danish Institute for Educational Research, Copenhagen, Denmark.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Rizopoulos D (2006). "**ltm**: An R Package for Latent Variable Modeling and Item Response Theory Analyses." *Journal of Statistical Software*, **17**(5), 1–25. URL http://www.jstatsoft.org/v17/i05/.

Samejima F (1969). "Estimation of Latent Ability Using a Response Pattern of Graded Scores." *Psychometric Monograph Supplement*, **4**(34).

Samejima F (1979). "A New Family of Models for the Multiple Choice Item." *Research Report 79-4*, University of Tennessee, Department of Psychology, Knoxville, TN.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York.

Stocking ML, Lord FM (1983). "Developing a Common Metric in Item Response Theory." *Applied Psychological Measurement*, **7**(2), 201–210.

Thissen D (2003). ***MULTILOG** 7: Multiple, Categorical Item Analysis and Test Scoring Using Item Response Theory*. Scientific Software International Inc., Lincolnwood, IL. Version 7.0, URL http://www.ssicentral.com/.

Thissen D, Steinberg L (1984). "A Response Model for Multiple Choice Items." *Psychometrika*, **49**(4), 501–519.

Thissen D, Steinberg L (1986). "A Taxonomy of Item Response Models." *Psychometrika*, **51**(4), 567–577.

Thurstone LL (1925). "A Method of Scaling Psychological and Educational Tests." *Journal of Educational Psychology*, **16**(7), 433–451.

Thurstone LL (1938). "Primary Mental Abilities." *Psychometric Monographs*. No. 1.

Weeks JP (2010). "**plink**: An R Package for Linking Mixed-Format Tests Using IRT-Based Methods." *Journal of Statistical Software*, **35**(12), 1–33. URL http://www.jstatsoft.org/v35/i12/.

Wood R, Wilson DT, Muraki E, Schilling SG, Gibbons R, Bock RD (2003). ***TESTFACT** 4: Test Scoring, Item Statistics, and Item Factor Analysis*. Scientific Software International Inc., Lincolnwood, IL. Version 4.0, URL http://www.ssicentral.com/.

Yao L (2008). ***BMIRT**: Bayesian Multivariate Item Response Theory*. CTB/McGraw-Hill, Monterey, CA. Version 1.0.

Yen WM (1986). "The Choice of Scale for Educational Measurement: An IRT Perspective." *Journal of Educational Measurement*, **23**(4), 299–325.

Zimowski MF, Muraki E, Mislevy RJ, Bock RD (2003). ***BILOG–MG** 3: Multiple-Group IRT Analysis and Test Maintenance for Binary Items*. Scientific Software International Inc., Lincolnwood, IL. Version 3.0, URL http://www.ssicentral.com/.

A. Additional features

The primary purpose of **plink** is to facilitate the linking of tests using IRT-based methods; however, there are three other notable features of the package. **plink** can be used to compute response probabilities, conduct IRT true-score and observed-score equating (Kolen and Brennan 2004), and plot item response curves and comparison plots for examining item parameter drift.

A.1. Computing response probabilities

For all of the item response models described in Section 2 there are associated functions for computing response probabilities. These include `drm` for Rasch, 1PL, 2PL, and 3PL items, `grm` for graded response model items, `gpcm` for partial credit/generalized partial credit model items, `nrm` for nominal response model items, and `mcm` for multiple-choice model items. There is also a function `mixed` for computing the response probabilities for mixed-format items. There are two principal arguments for the functions:

`x`: A vector/matrix/list[5] of item parameters or an `irt.pars` object.

`theta`: A vector of θ values for which response probabilities should be computed. If not specified, an equal interval range of values from -4 to 4 is used with an increment of 0.5.

In addition to these arguments, there are also some model-specific arguments. For the `drm` function there is a logical argument `incorrect` that identifies whether response probabilities for incorrect responses should be computed. For `grm` there is a logical argument, `catprob`, that identifies whether category or cumulative probabilities should be computed, and for `grm` and `gpcm` there is a logical argument, `location`, that indicates whether the parameters in `x` include a location parameter. Finally, in the functions `drm`, `grm`, and `gpcm` there is an argument `D` that can be used to specify a value for a scaling constant. When `mixed` is used, a single argument `D` can be specified and applied to all applicable models; otherwise, the arguments `D.drm`, `D.grm`, and `D.gpcm` can be used for each model respectively.

All of these functions output an object of class `irt.prob`. To illustrate the use of `mixed`, probabilities are computed for two dichotomous (3PL) items and one polytomous (GPCM) item with four categories using nine θ values ranging from -4 to 4 logits.

```
R> out <- mixed(mixed.pars, theta = -4:4)
R> round(get.prob(out), 3)
```

```
  theta1 item_1.1 item_2.1 item_3.1 item_3.2 item_3.3 item_3.4
1     -4    0.214    0.137    0.783    0.192    0.024    0.001
2     -3    0.265    0.144    0.631    0.293    0.068    0.008
3     -2    0.395    0.162    0.425    0.374    0.166    0.035
4     -1    0.619    0.197    0.216    0.361    0.303    0.120
5      0    0.829    0.267    0.077    0.243    0.388    0.291
6      1    0.940    0.385    0.020    0.118    0.356    0.506
```

[5] If one of these formulations is used, the objects `cat` and/or `poly.mod` may need to be passed to the function.

```
7      2    0.981    0.548    0.004    0.045    0.257    0.694
8      3    0.994    0.715    0.001    0.015    0.161    0.823
9      4    0.998    0.844    0.000    0.005    0.093    0.902
```

The probabilities are extracted from the output using the function `get.prob`. Note that the item names include decimal values. These values identify the response category for the given item. When any of these functions (`drm`, `grm`, etc.) are used with an `irt.pars` object with multiple groups, the output is returned as a list. The subsequent use of `get.prob` with this object will return a list containing the matrices of expected probabilities.

A.2. IRT true-score and observed-score equating

After the item parameters from two or more tests have been placed on a common scale, IRT true-score and observed-score equating methods can be used to relate number-correct scores across tests. In IRT, the true score for a given θ value is equivalent to the sum of expected probabilities across items at the specified ability. In true-score equating, the goal is to find the θ value associated with a given true-score on the base scale, then use this value to determine the corresponding true score(s) on the other test(s). When the items have lower asymptotes greater than zero (e.g., when using the 3PL), true-scores are only identified for values greater than the sum of the guessing parameters. In these instances, an ad hoc procedure developed by Kolen (1981) can be used to determine estimated true-scores below this point—but still in the range of observed scores—for each of the tests. In practice, true-score equating is typically implemented using all possible number-correct scores on the base scale.

In observed-score equating the goal is still to find equivalent number-correct scores across tests, but the mechanism for accomplishing this is vastly different than the true-score approach. In short, compound binomial/multinomial distributions of observed scores are created using synthetic populations for each test and then combined using a set of *a priori* established weights. These distributions are then equated using traditional equipercentile methods to identify the corresponding observed scores on the different tests (see Kolen and Brennan 2004, for a complete explanation of observed-score equating).

In **plink**, the `equate` function is used for IRT true-score and observed-score equating using all of the item response models described in section 2. The function has one essential argument, `x`, and nine optional arguments. The available arguments are as follows:

`x`: An `irt.pars` object with two or more groups or the output from `plink` containing rescaled item parameters.

`method`: A character vector identifying the equating method(s) to use. Values can include `"TSE"` and/or `"OSE"` for true-score and observed-score equating respectively.

`true.scores`: A numeric vector of true-score values to be equated. If missing, values corresponding to all possible observed scores will be used.

`ts.low`: A logical value. If `TRUE`, extrapolate values for the equated true-scores in the range of observed scores from one to the value below the lowest estimated true-score. The default is `TRUE`.

`base.grp`: An integer identifying the group for the base scale.

`score`: An integer identifying the scoring function to use to compute the true-scores. When `score = 1`, the ordered categories for each item are scored from 0 to $k - 1$, and when `score = 2`, the categories are scored from 1 to k. The default is `1`.

`startval`: An integer starting value for the first value of `true.score`.

`weights1`: A list containing quadrature points and weights to be used in the observed-score equating for population 1.

`weights2`: A list containing quadrature points and weights to be used in the observed-score equating for population 2.

`syn.weights`: A vector of length two or a list containing vectors of length two with synthetic population weights to be used for each pair of tests for populations 1 and 2 respectively for observed-score equating. If missing, weights of 0.5 will be used for both populations for all groups. If `syn.weights` is a list, the number of list elements should be equal to the number of groups in `x` minus one.

`...`: Further arguments passed to or from other methods.

As an illustration of the `equate` function, the example presented in (Kolen and Brennan 2004, pp. 191-198) is recreated using the dichotomous item parameters from the `KB04` dataset. As a first step, the forms are linked using the mean/sigma method, excluding item 27, with the "old" test as the base scale and the scaling constant, D, equal to 1.7.

```
R> pm <- as.poly.mod(36)
R> x <- as.irt.pars(KB04$pars, KB04$common,
+    cat = list(rep(2, 36), rep(2, 36)), poly.mod = list(pm, pm))

R> out <- plink(x, rescale = "MS", base.grp = 2, D = 1.7,
+    exclude = list(27, 27), grp.names = c("new", "old"))
```

As a next step, the `equate` function is run using the "new" form as the reference scale. For the true-score equating, all number-correct scores are used. The lowest values are determined using the ad hoc procedure. For the observed-score equating, the synthetic distribution is created using a specific set of quadrature points and weights with synthetic weights of 1 and 0 for the two populations respectively. In the output only the first ten equated true/observed scores are displayed. The marginal and synthetic population distributions are included in the output for the observed-score equating, but they are not displayed here to conserve space.

```
R> wt <- as.weight(theta = c(-5.21, -4.16, -3.12, -2.07, -1.03, 0.02, 1.06,
+    2.11, 3.15, 4.20), weight = c(0.0001, 0.0028, 0.0302, 0.1420, 0.3149,
+    0.3158, 0.1542, 0.0359, 0.0039, 0.0002))

R> eq.out <- equate(out, method = c("TSE", "OSE"), weights1 = wt,
+    syn.weights = c(1, 0), D = 1.7)
```

Equated true-scores

```
R> eq.out$tse[1:10,]
```

```
       theta new   old
 [1,]     NA   1 0.888
 [2,]     NA   2 1.776
 [3,]     NA   3 2.664
 [4,]     NA   4 3.552
 [5,]     NA   5 4.440
 [6,]     NA   6 5.328
 [7,] -4.336   7 6.134
 [8,] -2.769   8 7.187
 [9,] -2.063   9 8.396
[10,] -1.607  10 9.624
```

Equated observed-scores

```
R> eq.out$ose$scores[1:10,]
```

```
      new   old
 [1,]   0 0.000
 [2,]   1 0.618
 [3,]   2 1.580
 [4,]   3 2.546
 [5,]   4 3.519
 [6,]   5 4.503
 [7,]   6 5.506
 [8,]   7 6.533
 [9,]   8 7.587
[10,]   9 8.662
```

A.3. Plotting results

Two types of unidimensional plots can be created with **plink**: plots of item response curves and comparison plots for examining item parameter drift. The `plot` function—based on the `xyplot` function in the **lattice** package (Sarkar 2008)—has one essential argument, `x`, and ten optional arguments:[6]

`x`: An `irt.prob` or `irt.pars` object.

`separate`: A logical value identifying whether to plot the item category curves for polytomous items in separate panels.

`combine`: A numeric vector identifying the number of response categories to plot in each panel. If `NULL`, the curves will be grouped by item. This is typically used to plot curves for more than one item in a panel.

[6]There is an additional, optional argument `type` that can be used to create different multidimensional plots.

`items`: A numeric vector identifying the items to plot. When there are more than two groups (when `x` is an `irt.pars` object), `items` should be specified as a list with length equal to the number of groups where the list elements contain numeric vectors for the items that should be plotted for each group.

`item.names`: A vector of item names. When there are two or more groups (when `x` is an `irt.pars` object), `item.names` should be specified as a list with length equal to the number of groups in `x` where the list elements contain vectors of item names for each group.

`panels`: The number of panels to display in the output window. If the number of items is greater than `panels`, the plots will be created on multiple pages.

`drift`: A character vector identifying the plots to create to examine item parameter drift. Acceptable values are `a`, `b`, `c` for the various parameters respectively, `pars` to compare all of these parameters, `TCC` to compare test characteristic curves, `ICC` to compare item characteristic curves, or `all` to produce all of these plots.

`groups`: A numeric vector identifying the groups in `x` for which plots should be created (only applicable when there are two or more groups). When drift plots are being created, the values in `groups` should correspond to the group number of the lowest group of each pair of adjacent groups in `x`.

`grp.names`: A character vector of group names to use when creating the drift plots.

`sep.mod`: A logical value. If `TRUE`, use different markers in the drift plots to identify parameters related to different item response models.

`drift.sd`: A numeric value identifying the number of standard deviations to use when creating the perpendicular confidence region for the drift comparison plots. The default is `3`.

When plotting item response curves based on an `irt.pars` object, probabilities will be computed first using the `mixed` function. Any of the arguments in Appendix A.1 can be included. For example, a panel of item response curves is created using the `irt.pars` object from Section 3.1 with incorrect response probabilities plotted for the dichotomous items. A key is also included to identify the curves associated with each response category. The result is shown in Figure 1.

```
R> plot(pars, incorrect = TRUE, auto.key = list(space = "right"))
```

To illustrate the drift plots, the rescaled item parameters from Section 4.3 are compared for the Grade 5 tests in years 1 and 2. Different plot indicators for the two response models are used and a perpendicular confidence interval of two standard deviations is specified. The result is shown in Figure 2.

```
R> plot(out, drift = "pars", sep.mod = TRUE, groups = 4, drift.sd = 2)
```

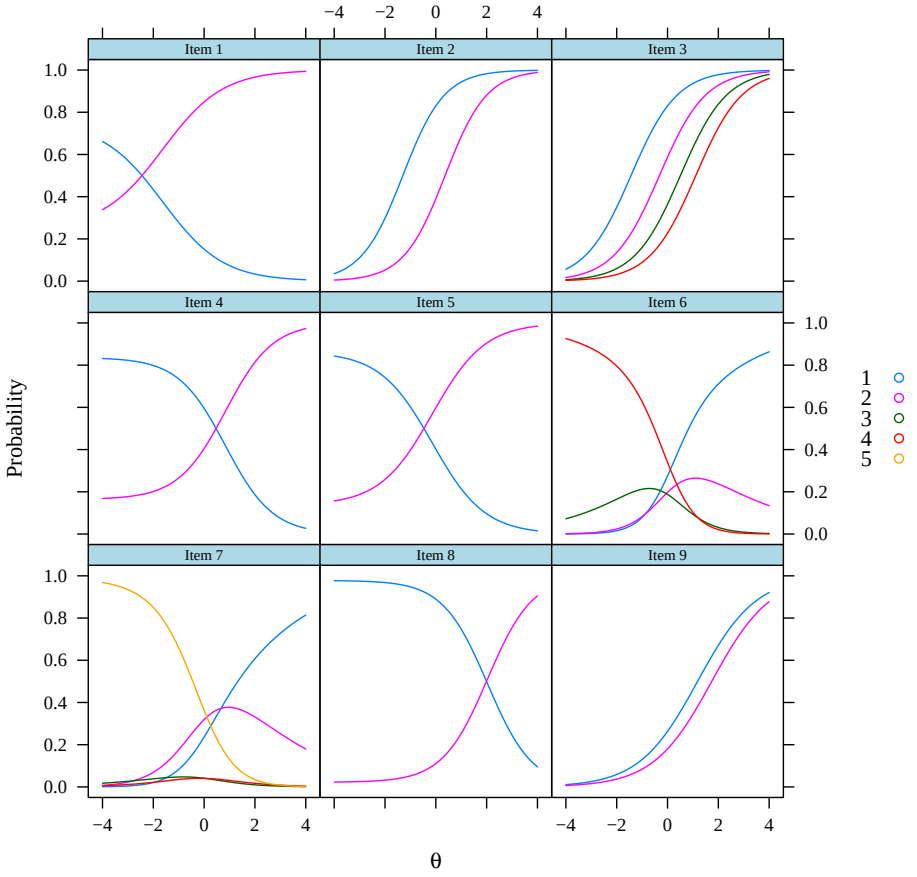

Figure 1: Item response curves.

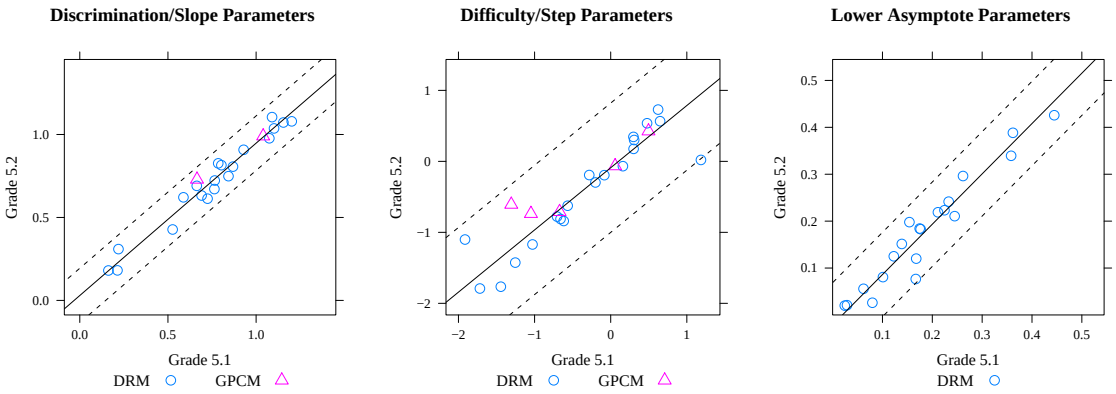

Figure 2: Common item parameter comparison.

B. Related software

There are a number of software applications currently available for conducting IRT-based linking; however, no formal comparison of these programs has ever been done. The following section provides an overview/critique of each of these applications then compares the estimated linking constants across programs for a set of mixed-format item parameters under various conditions (e.g., non-symmetric versus symmetric optimization and uniform versus normal weights) to determine if there are any appreciable differences.

One of the earliest programs to see widespread use was **EQUATE** (Baker 1993b) although now it is rarely used in practice. In the original version the program implemented the Stocking-Lord method for items modeled using the 1PL, 2PL, and 3PL. Two years later it was updated to allow for use of the graded response and nominal response models (the NRM utilizes the Haebara method). The program only allows for uniform weights at fixed ability intervals and does not support mixed-format tests, but it does include functionality to transform item and ability parameters using the estimated linking constants.

In the same year as the update to **EQUATE**, Hanson and Zeng (1995) released the program **ST** for linking dichotomous items, implementing the mean/sigma, mean/mean, Haebara, and Stocking-Lord methods. In addition to the inclusion of the moment methods, the program allows one to specify quadrature points and weights for the characteristic curve methods. Hanson (2000) also developed the program **mcmequate** for linking nominal response and multiple-choice model items using the Haebara method, but unlike **ST** no weighting options are available.

Two years after the release of **ST**, Lee and Oshima (1997) released the program **IpLink** which allows for both unidimensional and multidimensional linking of dichotomously scored items using the mean/sigma and characteristic curve methods with some flexibility for specifying quadrature points (the weights cannot be explicitly defined). Unlike the previous applications, **IpLink** has a graphical user interface (GUI). Still, the program does not appear to have been used much in practice for unidimensional linking.

As an extension of these programs (excluding the multidimensional methods), Kim and Kolen (2003) developed **POLYST** for linking dichotomous and polytomous items. The program implements the same four linking methods as **ST** for the 1PL, 2PL, 3PL, GRM, PCM/GPCM, NRM, and MCM, although the calibration can only be run for items corresponding to a single model at a time. In addition to the inclusion of more polytomous models, **POLYST** added increased functionality over the previous applications by allowing for symmetric and non-symmetric optimization with the characteristic curve methods and extensive options for weighting the response probabilities in the criterion function.

As a further extension of **POLYST**, Kim and Kolen (2004) developed the program **STUIRT** to handle mixed-format tests. **STUIRT** also includes two additional features: the ability to check for local minima near the final solution of the characteristic curve methods and functionality to create input files for use in **POLYEQUATE** (Kolen 2004) to conduct IRT true-score and observed-score equating. In **ST**, **mcmequate**, **IpLink**, **POLYST**, and **STUIRT** there is no functionality for transforming item parameters or ability estimates using the estimated linking constants.

One of the more recently developed applications is **IRTEQ** (Han 2007). This program implements the same four linking methods as **STUIRT** in addition to the robust mean/sigma method (Linn, Levine, Hastings, and Wardrop 1980) for the same item response models with

the exception of the nominal response and multiple-choice models. There are fewer options for weighting the response probabilities in the criterion function, and the program does not allow for symmetric optimization; however, the program does include functionality for rescaling item/ability parameters and conducting true-score equating. Two additional features of **IRTEQ** are the availability of a GUI—the program can still be run from a syntax file if desired—and the ability to create plots comparing the item parameters and test characteristic curves.

Two R packages, **irtoys** (Partchev 2009) and **MiscPsycho** (Doran 2010), also implement various linking methods, but both packages only include functionality for dichotomous items. **irtoys** is essentially a recreation of **ST** and **MiscPsycho** only implements the Stocking-Lord method.

There are two shortcomings to all of these applications: formatting of the input files, with the possible exception of the R packages, and functionality for linking multiple tests. The first five programs (not including **IpLink**) require the creation of control files that can be highly sensitive to formatting. For instance, including an extra carriage return, lowercase letter, etc. in the file may cause the program not to run at all. For **IRTEQ**, the issue relates specifically to how the item parameters must be formatted; they must conform to the **PARSCALE** (Muraki and Bock 2003) or **WinGen2** (Han and Hambelton 2007) output format. This is an added hassle for individuals not using **PARSCALE** to estimate item parameters or **WinGen2** to generate item parameters. The R packages require the item parameters to be formatted as a matrix or list for **irtoys** and **MiscPsycho** respectively, but depending on how the parameters are formatted when brought into R, some reformatting may be required. As described earlier, **plink** was written to provide a fair amount of flexibility in the formatting of item parameters by allowing them to be specified as vectors, matrices, lists, or imported from common IRT estimation programs.

The second major shortcoming is that none of the above programs allow for chain linking of item and/or ability parameters across multiple tests. In all of these applications, linking constants can only be estimated for two groups at a time, meaning multiple control files must be created to estimate each set of constants. Then, as a second step, item parameters and/or ability estimates must be iteratively transformed using another application. One of the key goals in developing **plink** was to overcome this limitation.

There are three features missing from **plink** that are available in other programs: the use of polygonal approximation and the ability to check for local minima with the characteristic curve methods (as implemented in **STUIRT**), and the availability of the robust mean/sigma method (as implemented in **IRTEQ**). However, **plink** provides greater flexibility for formatting and importing item parameters than any other program, it is the only program that supports chain-linking, and (although not addressed here) it includes extensive functionality for multidimensional test linking.

B.1. Comparing the applications

To examine the comparability of these applications and **plink**, linking constants were estimated with each program using the mixed-format item parameters available in **STUIRT** (example 3). There are two groups of 20 items (all common) which include ten 3PL items, five graded response model items, and five nominal response model items. Since most of the applications are unable to handle mixed-format tests, the item parameters were separated into five comparison groups: 3PL items only, GRM items only, NRM items only, 3PL+GRM

items, and 3PL+GRM+NRM items. Linking constants were estimated for each comparison group, when applicable, using all available methods in each application. For the characteristic curve methods, a combination of two additional options, when applicable, were specified: uniform versus normal weights and symmetric versus non-symmetric optimization.

There were no differences in the linking constants estimated using the moment methods, but one should not expect any given that estimates are based solely on means and standard deviations. The only instance where differences might occur is with the mean/sigma method if the denominator for the standard deviations is $n - 1$ versus n. **ST** has the option to use either, but all other programs that implement this approach use n in the denominator. For the characteristic curve methods, all of the programs produced nearly identical results, and when differences did occur they were at or beyond the third decimal place. Since all of the programs provide consistent estimates for the linking constants, the only distinguishing features are the availability of options and ease of use.

Affiliation:

Jonathan Weeks
School of Education
University of Colorado at Boulder
UCB 249
Boulder, CO 80309, United States of America
E-mail: jonathan.weeks@colorado.edu