Journal of Statistical Software

July 2016, Volume 71, Issue 5. doi: 10.18637/jss.v071.i05

Generating Adaptive and Non-Adaptive Test Interfaces for Multidimensional Item Response Theory Applications

R. Philip Chalmers
York University

Abstract

Computerized adaptive testing (CAT) is a powerful technique to help improve measurement precision and reduce the total number of items required in educational, psychological, and medical tests. In CATs, tailored test forms are progressively constructed by capitalizing on information available from responses to previous items. CAT applications primarily have relied on unidimensional item response theory (IRT) to help select which items should be administered during the session. However, multidimensional CATs may be constructed to improve measurement precision and further reduce the number of items required to measure multiple traits simultaneously.

A small selection of CAT simulation packages exist for the R environment; namely, **catR** (Magis and Raîche 2012), **catIrt** (Nydick 2014), and **MAT** (Choi and King 2014). However, the ability to generate graphical user interfaces for administering CATs in real-time has not been implemented in R to date, support for multidimensional CATs have been limited to the multidimensional three-parameter logistic model, and CAT designs were required to contain IRT models from the same modeling family. This article describes a new R package for implementing unidimensional and multidimensional CATs using a wide variety of IRT models, which can be unique for each respective test item, and demonstrates how graphical user interfaces and Monte Carlo simulation designs can be constructed with the **mirtCAT** package.

Keywords: MIRT, CAT, multidimensional CAT, R, GUI.

1. Introduction

Computerized adaptive testing (CAT) is a methodology designed to reduce the length of educational, psychological, and medical tests. In contrast to fixed linear tests (e.g., paper-and-pencil forms, or digital surveys where questions are administered in sequence), CATs

attempt to select optimal items based on selection rules that capitalize on pre-calibrated item information and the participants' provisional trait estimates (Weiss 1982). Throughout a CAT session, the trait estimates are updated as the responses to items are collected. The trait estimates serve as a basis in determining which items should be administered next, and the associated standard errors for the estimates help inform whether the CAT session should be terminated early. In common CAT designs, items are administered when they are believed to effectively reduce the expected standard error of measurement of the latent trait values. Administering items which optimally reduce the standard error of measurement helps to create efficient test forms that improve the measurement reliability for a given participant by using a smaller subset of test items (Wainer and Dorans 2000).

In order to implement CATs effectively, various item-level characteristics must be known a priori. Specifically, the parameters required to operationalize the item selection process, as well as to compute provisional latent trait estimates, are generally adopted from the item response theory paradigm (IRT; Lord 1980). IRT parameters can be estimated for tests containing unidimensional or multidimensional latent trait structures, and offer a parametric mechanism to model the interaction between participants and item characteristics (Reckase 2009). CATs based on a unidimensional latent trait assumption have been extensively studied in methodological literature; however, with the advent of modern computing power, multidimensional CATs are becoming more popular (Reckase 2009). Multidimensional CATs (MCATs) are a useful alternative to administering multiple unidimensional CATs in situations where the traits are correlated (Segall 1996) or when items simultaneously capture variation in multiple traits (i.e., have "cross-loadings" in the vernacular of linear factor analysis; Mulaik 2010). Correlations between latent traits provide additional information about the locations of auxiliary traits, and in turn help to improve the overall measurement precision between the trait estimates. Due to the increase in statistical information, MCATs will often require fewer items than independently administered unidimensional CATs to reach the same measurement precision (Mulder and Linden 2009).

Several important prerequisites are required before building interfaces to be used for MCATs. A cursory overview of these prerequisites include:

- *Obtaining a suitable item pool.* The item pool (or bank) is a relatively large set of items that can be selected from during an MCAT application. The associated item parameters must have been calibrated for the population of interest beforehand using multidimensional IRT software (e.g., the **mirt** package in R Chalmers 2012, or an equivalent). In situations where more than one population will be administered items from the pool, all items should contain limited to no differential functioning to ensure that the selection of items is unbiased (Chalmers, Counsell, and Flora 2016; Wainer and Dorans 2000).

- *Initializing the MCAT session.* Before an MCAT session can begin, initial latent trait estimates and hyper-parameter distribution definitions are generally required. The initial trait estimates often serve as the basis for selecting the initial item (if the initial item was not explicitly declared), while the hyper-parameter distributions are included as an added information component in the item selection process. The hyper-parameters are also included to add prior distributional information when updating the ability estimates throughout the MCAT session. When there is little to no prior information about the ability estimates, the starting values are generally selected to equal the mean of the latent trait distribution (often this is simply a vector of zeros).

- *Selecting the next item to administer.* Several criteria have been proposed for unidimensional CATs to select optimal items for ability and classification designs, many of which have been implemented in unidimensional CAT software in R (e.g., see Magis and Raîche 2012). Fewer MCAT criteria have been proposed in the literature, though a small number of criteria are available. MCAT selection methods include the determinant-rule (D-rule), trace of the information or asymptotic covariance matrix (T-rule and A-rule, respectively), weighted composite rule (W-rule), eigenvalue-rule (E-rule), and the Kullback-Leibler divergence criteria (Kullback and Leibler 1951). Due to their importance in MCAT applications, these criteria are explained in more detail below.

- *(Optional) – Selecting a pre-CAT design.* Because MCAT estimation methods are based on responses to previous items, it can be desirable to run a "pre-CAT" stage before beginning the actual MCAT. In the pre-CAT stage, a small selection of items are administered under more controlled settings to ensure that, during the MCAT stage, the methods have enough information to be properly executed.

- *Selecting the IRT scoring method.* Multiple criteria have been defined for obtaining provisional trait estimates. These criteria include: maximum-likelihood (ML) estimation, evaluating the expected or maximum values of the a posteriori distribution, weighted likelihood estimation, and several others (Bock and Aitkin 1981; Warm 1989). However, ML estimation requires special care because it cannot be used if responses are at the extreme ends of the categories (i.e., all-correct, all-incorrect). One possible solution to this issue is to use Bayesian methods (such as maximum a posteriori estimation) until a sufficient amount of variability in the responses are available for proper ML estimation. Another potential solution when selecting the ML algorithm is to include a pre-MCAT stage to collect responses until suitable ML estimates can be obtained.

- *Terminating the application.* Deciding how to terminate an MCAT session is important for many practical reasons. MCATs may be terminated according to multiple criteria in a single session. For example, terminating a test based on the standard error of measurement is desirable if inferences about the precision of each latent trait estimate is required, though for multidimensional models the choice of whether this criteria should be applied globally or specifically for each latent trait must be specified. Tests may also be terminated after a specific number of items have been administered, the time allotted for answering the test has expired, the latent traits can be classified as above or below a set of predetermined latent cutoff values (Eggen 1999), and so on.

Much of the superficial information listed above is also important for unidimensional CAT applications. Conversely, literature relevant to unidimensional CATs will largely be relevant for MCATs because they share the same underlying methodology. Therefore, additional information regarding MCAT methodology can largely be obtained from previous CAT publications, such as Magis and Raîche (2012) and the references therein.

A small number of R packages exist for studying CAT designs through Monte Carlo simulations, including **catR** (Magis and Raîche 2012) and **catIrt** (Nydick 2014), which exclusively focus on unidimensional IRT models, and **MAT** (Choi and King 2014), which exclusively investigates the properties of the multidimensional three-parameter logistic model (M3PL). Hitherto, these packages have provided useful simulation tools for Monte Carlo research of

CAT design combinations with homogeneous IRT models; however, they have not been organized for real-time implementation of CATs, do not provide resources to build graphical user interfaces (GUIs), exclusively support either unidimensional or multidimensional CATs, and do not support mixing different classes of IRT models into CAT designs.

As more R packages are developed for studying unidimensional and multidimensional CATs, a number of pertinent features remain missing. The **mirtCAT** package described in this article has been designed to address many of these missing features in the R environment. Specifically, **mirtCAT** provides front-end users with functions for generating CAT GUIs to be used in their research applications, and includes several tools for investigating the statistical properties of heterogeneous CAT designs by way of Monte Carlo simulations. The remainder of this article describes the theory behind applying MCATs, provides examples of how Monte Carlo simulation studies can be organized with the code from the package, and demonstrates how real-time unidimensional and multidimensional CATs – as well as standard questionnaire designs – can be generated to collect item response data.

2. Multidimensional computerized adaptive testing

A number of multidimensional IRT models have been proposed for dichotomous and polytomous response data. For ease of presentation, we will only focus on the multidimensional four-parameter logistic model (M4PL) for dichotomous responses (coded as 0 and 1 for incorrect and correct answers, respectively), which is an extension of the multidimensional three-parameter logistic model (Reckase 2009), and the multidimensional nominal response model for polytomous items (Thissen, Cai, and Bock 2010). Multidimensional IRT models often contain a unidimensional counterpart as a special case when only one latent trait is modeled; therefore, the following theory relates to unidimensional IRT models as well.[1]

The probability that a participant positively endorses the j-th dichotomous item with an M4PL structure is

$$\mathsf{P}_j(y = 1|\boldsymbol{\theta}) = \mathsf{P}_j(y = 1|\boldsymbol{\theta}, \mathbf{a}_j, d_j, g_j, u_j) = g_j + \frac{u_j - g_j}{1 + \exp(-(\mathbf{a}_j^\top \boldsymbol{\theta} + d_j))}, \tag{1}$$

where the complementary probability for answering the item incorrectly is $\mathsf{P}_j(y = 0|\boldsymbol{\theta}) = 1 - \mathsf{P}_j(y = 1|\boldsymbol{\theta})$. The g_j and u_j parameters are restricted to be between 0 and 1 (where $g_j < u_j$), and serve to bound the probability space within $g_j \leq \mathsf{P}_j(y = 1|\boldsymbol{\theta}) \leq u_j$. The g parameter is useful when there is a non-zero probability for participants to randomly guess an item correctly. The u_j parameter, on the other hand, controls the probability that participants will carelessly answer an item incorrectly. The multidimensional two-parameter logistic model (M2PL) can be recovered from Equation 1 when the g_j and u_j are fixed to 0 and 1, respectively, and the multidimensional three-parameter model (M3PL) is realized when fixing only u_j to 1. Finally, $\boldsymbol{\theta}$ is taken to be a D-dimensional vector of random ability or latent trait values, d_j is the item intercept parameter representing the relative item "easiness", and $\mathbf{a}_j$ is a vector of slope parameters that modulate how $\boldsymbol{\theta}$ influences the probability function.

The multidimensional nominal response model (MRNM) can be used to model K-unordered

[1] Although only two IRT models are presented below, in principle many other IRT models may be substituted in empirical applications.

polytomous response categories that are coded $k = 0, 1, \ldots, K-1$. This model has the form

$$\mathsf{P}_j(y = k|\boldsymbol{\theta}) = \mathsf{P}_j(y = k|\boldsymbol{\theta}, \mathbf{a}_j, \boldsymbol{\alpha}_j, \mathbf{d}_j) = \frac{\exp\left(\boldsymbol{\alpha}_{jk}\mathbf{a}_j^\top\boldsymbol{\theta} + \mathbf{d}_{jk}\right)}{\sum_{k=0}^{K-1}\exp\left(\boldsymbol{\alpha}_{jk}\mathbf{a}_j^\top\boldsymbol{\theta} + \mathbf{d}_{jk}\right)}, \tag{2}$$

where the $\mathbf{a}_j$ and $\boldsymbol{\theta}$ terms have the same interpretation as in Equation 1. Equation 2 contains unique intercept values ($\mathbf{d}_{jk}$) and so-called "scoring" parameters ($\boldsymbol{\alpha}_{jk}$) for each respective category. For identification purposes, the first element of $\mathbf{d}_{jk}$ and $\boldsymbol{\alpha}_{jk}$ are often constrained to be equal to 0, while the last element of $\boldsymbol{\alpha}_{jk}$ is constrained to be $K-1$. The $\boldsymbol{\alpha}_{jk}$ values represent the relative ordering of the categories; larger $\boldsymbol{\alpha}_{jk}$ values indicate that the category has a stronger relationship with higher levels of $\boldsymbol{\theta}$. When specific constraints are applied to Equation 2, various specialized IRT models can be recovered. For instance, when the scoring parameters are constrained to have equal interval spacing ($\boldsymbol{\alpha}_j = 0, 1, 2, \ldots, K-1$) the multidimensional generalized partial credit model (MGPCM) is realized, and when $K = 2$ the MRNM will become equivalent to the M2PL model.

2.1. Predicting latent trait scores

After item responses have been collected, various estimates for $\boldsymbol{\theta}$ can be computed. The $\hat{\boldsymbol{\theta}}$ estimates are obtained using the observed item responses, the item trace-line functions given their respective item parameters ($\boldsymbol{\psi}_j$), and (potentially) prior distributional information about $\boldsymbol{\theta}$. Multiple methods exist for obtaining $\hat{\boldsymbol{\theta}}$ values, such as weighted and unweighted maximum-likelihood estimation (WLE and ML, respectively; Bock and Aitkin 1981; Warm 1989), Bayesian methods such as the expectation or maximum of the posterior distribution (EAP and MAP, respectively; Bock and Aitkin 1981), and several others which have seen less use in applied settings (e.g., EAP for sum scores; Thissen, Pommerich, Billeaud, and Williams 1995). ML estimation of $\boldsymbol{\theta}$ for a given response pattern requires optimizing the likelihood function

$$l(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi}) = \prod_{j=1}^{J}\prod_{k=0}^{K_j-1}\mathsf{P}_j(y = k|\boldsymbol{\theta}, \boldsymbol{\psi}_j)^{\chi_{jk}}, \tag{3}$$

where χ_{jk} is a dichotomous indicator variable (coded as 0 or 1) used to select the probability terms corresponding to the endorsed categories. In practice, however, it is generally more effective to use the log of Equation 3,

$$LL(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi}) = \sum_{j=1}^{J}\sum_{k=0}^{K_j-1}\chi_{jk}\cdot\log\left[\mathsf{P}_j(y = k|\boldsymbol{\theta}, \boldsymbol{\psi}_j)\right]. \tag{4}$$

Optimizing the log-likelihood directly results in ML estimates; however, obtaining a possible maximum requires that $\mathbf{y}$ contain a mix of 0 to $K_j - 1$ responses across the J items. If there is no variability in the response vector, such that $SD(\mathbf{y}) = 0$, then $\hat{\boldsymbol{\theta}}$ will tend to $-\infty$ or ∞ during optimization. Bayesian methods generally do not from suffer this particular limitation because they include additional information about the distribution of $\boldsymbol{\theta}$ through a prior density function, $\phi(\cdot)$, with hyper-parameters, $\boldsymbol{\eta}$. The posterior function utilized in Bayesian prediction methods is

$$\pi(\boldsymbol{\theta}|\mathbf{y}, \boldsymbol{\psi}, \boldsymbol{\eta}) = \frac{l(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})\phi(\boldsymbol{\theta}|\boldsymbol{\eta})}{\int l(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})\phi(\boldsymbol{\theta}|\boldsymbol{\eta})}, \tag{5}$$

where Equation 5 is either integrated across to find the EAP estimates or maximized to find the MAP estimates. In multidimensional IRT applications, the prior density function is typically assumed to be from the multivariate Gaussian family with mean vector $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$; however, other multivariate density functions are possible.

Following the computation of $\hat{\boldsymbol{\theta}}$, a measure of precision is required to make inferences about the statistical precision of the estimates. As is the case with standard ML estimation theory, computing a quadratic approximation of the curvature in Equation 4 provides a suitable measure of the parameter variability (Fisher 1925). The computation of $\mathsf{VAR}(\hat{\boldsymbol{\theta}})$ is determined by inverting the $D \times D$ matrix of second derivatives with respect to $\boldsymbol{\theta}$ (also known as the Hessian or negative of the observed information matrix),

$$\mathsf{VAR}(\hat{\boldsymbol{\theta}}) = \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}|\mathbf{y}, \boldsymbol{\psi}) = -\left(\frac{\partial^2 LL(\mathbf{y}|\hat{\boldsymbol{\theta}}, \boldsymbol{\psi})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}\right)^{-1}. \tag{6}$$

Standard error estimates for each element in $\hat{\boldsymbol{\theta}}$ are then obtained by taking the square-root of each diagonal element in the asymptotic covariance matrix, $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}|\mathbf{y}, \boldsymbol{\psi})$. If the log of Equation 5 is used instead of Equation 4 then prior information about $\boldsymbol{\theta}$ will also be included in the computation of the Hessian. Due to the added statistical information in Bayesian methods, $\boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}|\mathbf{y}, \boldsymbol{\psi}, \boldsymbol{\eta})$ will generally provide slightly smaller standard errors when an informative prior distribution is included in the computations.[2]

2.2. Item selection for MIRT models

Selecting optimal items in MCATs is generally more complicated than unidimensional CATs because items should only be selected if they effectively improve the precision of multiple traits. In this section, we focus on item selection methods that are tailored towards obtaining the lowest $SE(\hat{\boldsymbol{\theta}})$ for all individuals sampled. An interesting area that is not investigated in this section is when items are selected so to optimally classify individuals above or below predefined cutoff values. Although various methods exist for unidimensional models, classification-based applications for MCATs have rarely been investigated in the literature and continue be an important area for future research (Reckase 2009).

Selecting items according to the maximum information principle (Lord 1980) requires evaluating the Fisher-information matrix for each remaining item in the pool. The Fisher information is defined as

$$\mathcal{F}(\boldsymbol{\theta}) = -\mathsf{E}\left(\frac{\partial^2 LL(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{\psi})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}\right) \tag{7}$$

where the inverse of Equation 7 serves as another suitable measure to approximate sampling variability of $\boldsymbol{\theta}$. For notational clarity, the vector of parameters $\boldsymbol{\psi}$ is omitted from the following presentation because the item parameters are assumed to be constant. $\mathcal{F}(\boldsymbol{\theta})$ is useful in MCAT applications because it contains no reference to the observed response patterns, and therefore can be used to predict the amount of *expected* information contributed by items that have not been administered. However, in MCAT applications $\boldsymbol{\theta}$ is not known beforehand; therefore, provisional estimates ($\hat{\boldsymbol{\theta}}$) are used instead as plausible stand-ins for $\boldsymbol{\theta}$. The $\hat{\boldsymbol{\theta}}$ values are continually updated throughout the MCAT session to provide better approximates to the

[2]The Bayesian analogue of $SE(\hat{\boldsymbol{\theta}})$ is the actually posterior standard deviation of the trait estimates, $PSD(\hat{\boldsymbol{\theta}})$. For simplicity, however, in the remainder of the text the two terms are used interchangeably.

unobserved $\boldsymbol{\theta}$ values. Because the precision about $\boldsymbol{\theta}$ improves as more items are administered, the Fisher information criteria will in turn progressively select more suitable items for the unobserved latent trait values.

As outlined in work by Segall (1996), selecting the most informative item requires evaluating $\mathcal{F}(\hat{\boldsymbol{\theta}})$ for each of the M remaining items in the item pool. Due to the local independence assumption in IRT (Lord 1980), the information contributed by the addition of the m-th item is additive, such that

$$\mathcal{F}_{J+m}(\hat{\boldsymbol{\theta}}) = \mathcal{F}_J(\hat{\boldsymbol{\theta}}) + \mathcal{F}_m(\hat{\boldsymbol{\theta}}), \tag{8}$$

where $\mathcal{F}_J(\hat{\boldsymbol{\theta}})$ is the sum of the information matricies for the previously answered items. The matrix in Equation 8 is evaluated for the $m = 1, 2, \ldots, M$ remaining items in the pool. The M information matrices are then compared by reducing the multidimensional information to suitable scalar values according to how the joint variability should be quantified. If prior distribution information is included in the selection process then the following formula can be used to compute a Bayesian variant of the expected information (Segall 1996)

$$\mathcal{F}_{J+m}(\hat{\boldsymbol{\theta}}) = \mathcal{F}_J(\hat{\boldsymbol{\theta}}) + \mathcal{F}_m(\hat{\boldsymbol{\theta}}) - (\partial^2 \log(\phi(\boldsymbol{\theta}|\boldsymbol{\eta})))^{-1}. \tag{9}$$

In the situation where a multivariate normal prior distribution is included, Equation 9 can be expressed as

$$\mathcal{F}_{J+m}(\hat{\boldsymbol{\theta}}) = \mathcal{F}_J(\hat{\boldsymbol{\theta}}) + \mathcal{F}_m(\hat{\boldsymbol{\theta}}) + \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}. \tag{10}$$

One potentially optimal approach to quantify the amount of joint item information in $\mathcal{F}_{J+m}(\hat{\boldsymbol{\theta}})$ is to select the item which provides the largest matrix determinant. The item with the maximum determinant indicates which item provides the largest increase in the volume of $\mathcal{F}_J(\hat{\boldsymbol{\theta}})$; consequently, this selection property will maximally decrease the overall volume in $\boldsymbol{\Sigma}_J(\hat{\boldsymbol{\theta}})$ as well, where $\boldsymbol{\Sigma}_J(\hat{\boldsymbol{\theta}}) = \mathcal{F}_J^{-1}(\hat{\boldsymbol{\theta}})$ (Segall 1996). This criterion is called the "D-rule", and is more formally expressed as

$$\text{D-rule} = \max\left(|\mathcal{F}_{J+1}(\hat{\boldsymbol{\theta}})|, |\mathcal{F}_{J+2}(\hat{\boldsymbol{\theta}})|, \ldots, |\mathcal{F}_{J+M}(\hat{\boldsymbol{\theta}})|\right). \tag{11}$$

Another potentially useful criterion for selecting items is the maximum trace of $\mathcal{F}_{J+m}(\hat{\boldsymbol{\theta}})$,

$$\text{T-rule} = \max\left(\text{Tr}(\mathcal{F}_{J+1}(\hat{\boldsymbol{\theta}})), \text{Tr}(\mathcal{F}_{J+2}(\hat{\boldsymbol{\theta}})), \ldots, \text{Tr}(\mathcal{F}_{J+M}(\hat{\boldsymbol{\theta}}))\right). \tag{12}$$

While the T-rule does not guarantee the largest reduction in volume for $\boldsymbol{\Sigma}_J(\hat{\boldsymbol{\theta}})$, it does select items which increase the average unweighted information about the latent traits, and also allows for unequal domain score weights to be applied if certain latent traits are deemed to be more important a priori. Applying weights to the T-rule helps to measure important traits more accurately because items will be selected with greater frequency if they measure the traits of interest. A closely related selection criterion to the T-rule is the asymptotic covariance rule, or A-rule (Mulder and Linden 2009), which selects items based on the minimum (potentially weighted) trace of $\boldsymbol{\Sigma}_{J+m}(\hat{\boldsymbol{\theta}})$,

$$\text{A-rule} = \min\left(\text{Tr}(\boldsymbol{\Sigma}_{J+1}(\hat{\boldsymbol{\theta}})), \text{Tr}(\boldsymbol{\Sigma}_{J+2}(\hat{\boldsymbol{\theta}})), \ldots, \text{Tr}(\boldsymbol{\Sigma}_{J+M}(\hat{\boldsymbol{\theta}}))\right). \tag{13}$$

Much like the T-rule, the A-rule does not guarantee the maximum increase in the volume of the information matrix. Instead, the A-rule attempts to reduced the marginal expected standard

error for each $\hat{\boldsymbol{\theta}}$ by ignoring the covariation between traits. Next, the eigenvalue rule (E-rule) has been proposed to select the item which minimizes the general variance of the ability estimates by selecting the smallest possible value from the set of eigenvalues in each $\boldsymbol{\Sigma}_{J+m}(\hat{\boldsymbol{\theta}})$. However, the E-rule may not optimally select items in a way that maximally reduces the standard error of measurement for all latent traits, and in general is not recommended for routine use (Mulder and Linden 2009). Finally, the W-rule can be used to select the maximum of the weighted information criteria, $W = \mathbf{w}^\top \mathcal{F}_{J+m}(\hat{\boldsymbol{\theta}})\mathbf{w}$, where $\mathbf{w}$ is a weight vector subject to the constraint $\mathbf{1}^\top \mathbf{w} \equiv 1$ (Linden 1999). As with the optional weights for the T-rule and A-rule, the W-rule is an effective selection mechanism when specific latent traits should have lower measurement error than other traits.

An alternative approach to selecting items using the Fisher information given provisional $\hat{\boldsymbol{\theta}}$ values is the Kullback-Leibler information (Chang and Ying 1996). This approach has the potential benefit over traditional information-based methods in that it can account for uncertainty of the $\hat{\boldsymbol{\theta}}$ values when only a small number of items have been administered (Chang and Ying 1996). The Kullback-Leibler information is

$$KL(\boldsymbol{\theta}||\boldsymbol{\theta}_0) = \mathsf{E}_{\boldsymbol{\theta}_0}\left(LL(\mathbf{y}|\boldsymbol{\theta}_0) - LL(\mathbf{y}|\boldsymbol{\theta})\right), \tag{14}$$

where $\boldsymbol{\theta}_0$ is the vector of true ability parameters, and the double bars in $KL(\boldsymbol{\theta}||\boldsymbol{\theta}_0)$ signify that $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_0$ should be treated distinctly. Chang and Ying (1996) suggested that Equation 14 should be evaluated over the range $\boldsymbol{\theta} \pm \boldsymbol{\Delta}_n$, where $\boldsymbol{\Delta}_n$ may decrease by a factor of $\sqrt{n}$ as the number of items administered increases. A numerical integration approach is also possible for the Kullback-Leibler information, though for multidimensional IRT models this may be less useful because of the (often cumbersome) numerical evaluation of the integration grid across all latent trait dimensions (see Reckase 2009, p. 335, for a similar observation about evaluating the KL information in MCATs).

2.3. Exposure control and content balancing

An unfortunate consequence when maintaining item pools is that informative items are often selected with greater frequency than less informative items. Exposing a smaller selection of items too often may lead to item security issues, loss of investments for items that are rarely selected, or in some cases may cause a decrease in content validity coverage due to reduced item sampling variability. Selection methods can be adjusted by including "exposure control" methods to help avoid overusing highly informative items (Linden and Glas 2010). Several methods of exposure control exist, though perhaps the most intuitive approach is the method proposed by McBride and Martin (1983). In their method, McBride and Martin suggest sampling from the n most optimal items (given the selection criteria) rather than simply selecting the most optimal item, and further recommend gradually reducing n as the examinee progresses through the test (the so-called 5-4-3-2-1 approach). This helps generate item variability in earlier stages of the test where item overexposure is more likely to occur.

Simulation-based item exposure methods, such as the Sympson-Hetter (SH) approach (e.g., Veldkamp and Linden 2008), provide a different approach to controlling item overexposure. The SH method requires items to be pre-assigned a fixed value ranging from 0 to 1, and during the CAT session a simulation experiment is performed to determine whether or not a selected item is to be administered. For instance, after an optimal item is determined from the item pool, a random uniform value (r) is then drawn and compared to the item's assigned

SH value. If r is less than the assigned SH value then the item is administered, otherwise it is discarded from the item pool and the next most optimal item undergoes the same simulation experiment; this process continues until an item is selected and administered.

Unfortunately, it is not clear whether more advanced exposure control methods outperform simple heuristic methods in empirical settings (e.g., see, Revuelta 1998). An undesirable side-effect when implementing exposure control methods is that there is loss of item selection efficiency, and in most applications this loss of efficiency will require more items to be administered before termination criteria based on the standard error of measurement can be obtained. Using exposure control early in the MCAT session, where it often is most important, also generates uncertainty about selection methods that theoretically should perform well earlier in the CAT session (such as the Kullback-Leibler criteria).

Another area of interest when administering test items is the use of "content balancing" to ensure that specific types of item content appear during the CAT session. Content balancing generally involves the classification of items to predetermined groups, and these groups are assigned a proportion or percentage value relating to how often the content groups should appear in the CAT session. A simple yet effective method for content balancing, proposed by Kingsbury and Zara (1991), involves comparing the empirically obtained content proportions to the desired content proportions. After computing the selection criteria for each remaining item in the item pool, the proportion of items in the content domains for all the items previously administered are subtracted from the desired content domain percentages for the respective domains. The content domain that has the largest difference between the desired proportion is then selected, and the item with the most optimal selection criteria within the selected domain is then administered. This approach ensures that content balancing is efficiently achieved throughout the session while quasi-maintaining the optimal item selection criteria.

Content balancing methods mainly have been studied in unidimensional CAT applications, however they can be applied equally well to MCATs. Fortunately, MCAT designs can implicitly offer a suitable approach to balancing content domains. As Segall (1996) explained, an MCAT session primarily intended to measure one trait could be organized to form a bi-factor structure (e.g., Gibbons and Hedeker 1992; Gibbons *et al.* 2007) to achieve a content balancing effect. Within the bi-factor model, each content grouping can be organized as a specific latent trait, where item slopes only relate to the respective subsets of homogeneous content groupings. Given the bi-factor structure, items could then be selected using criteria that weight the selection process in favor of selecting items which measure the general trait, while also including information about the specific traits; this would result in a probabilistic selection mechanism for sampling the content domains indirectly with the item selection criteria. Of course, practitioners may still wish to include more traditional content balancing methods if other properties of the test should be selected. In this case, multiple content balancing methods could be combined to form a balanced sampling design. For instance, in addition to selecting specific contents with the bi-factor design, a CAT session may also be organized to contain 80% multiple-choice questions, 10% reading comprehension questions, and 10% fill-in-the-blank questions, and these proportions can be controlled using Kingsbury and Zara's (1991) method.

2.4. Termination criteria

In the interest of time, item bank security, and avoiding fatigue effects, one or more stopping criteria should be included in MCATs. One reasonable approach to terminating the MCAT application is to require all $SE(\hat{\boldsymbol{\theta}}_k) \leq \delta$, where δ is a maximally tolerable standard error of measurement for all the latent traits. However, if some elements in $\hat{\boldsymbol{\theta}}$ should be measured with more precision then unique δ_k values for each $\hat{\boldsymbol{\theta}}_k$ value should be defined. For example, if the MCAT is organized to contain a bi-factor structure then the test developer may wish to terminate the session when only the primary trait reaches a predefined δ_k value. Unequal δ_k values should be used in conjunction with the W-rule, weighted T-rule, or weighted A-rule, so that items which accurately measure the traits of interest are selected with greater frequency.

Classification-based criteria also exist for terminating MCATs when cutoff values are supplied for each trait. In classification-based MCATs, the session may be terminated when the confidence intervals (given $1 - \alpha$) for each $\hat{\boldsymbol{\theta}}$ do not contain the pre-specified cutoff values. When the $CI(\hat{\boldsymbol{\theta}})$s do not contain the cutoff values then the individuals may be classified as above or below the cutoff values for the respective traits, otherwise the MCAT results will suggest that not enough information exists to reject the plausibility of the cutoffs. More specific methods for terminating CATs are also possible (e.g., use of loss functions or risk analyses), however these are not explored in this article.

Finally, termination criteria can be based on other practical considerations as well, such as setting the maximum number of items that can be administered in a given session, stopping the CAT after a specific amount of time has elapsed, the $\hat{\boldsymbol{\theta}}$ values are changing very little as new items are added, and so on.

3. The mirtCAT package

The **mirtCAT** package (available from the Comprehensive R Archive Network at `https://CRAN.R-project.org/package=mirtCAT`) provides tools for test developers to generate GUIs for CATs as well as functions for Monte Carlo simulation studies involving CAT designs. **mirtCAT** uses the HTML generating tools available in the **shiny** package (RStudio Inc. 2014) to generate real-time CATs for interactive sessions within standard web-browsing software. The **mirtCAT** package builds and extends upon the estimation tools available in the **mirt** package (Chalmers 2012), and provides a wide range of support for any mixture of unidimensional and multidimensional IRT models to be used for CATs.

Currently supported models in **mirtCAT** include unidimensional and multidimensional versions of the 4PL model (Lord 1980), the graded response model and its rating scale counterpart (Muraki 1992; Muraki and Carlson 1995; Samejima 1969), the generalized partial credit and nominal model (Thissen *et al.* 2010), the partially compensatory model (Chalmers and Flora 2014; Sympson 1977), the nested-logit model (Suh and Bolt 2010), the ideal-point model (Maydeu-Olivares, Hernández, and McDonald 2006), and polynomial or product constructed latent trait combinations (Bock and Aitkin 1981). Additionally, the **mirt** package supports the use of prior parameter distributions, linear and non-linear parameter constraints (e.g., see, Chalmers 2015), and specification of fixed parameter values; hence, nested versions of the previously mentioned models can be estimated from empirical data. For instance, the 1PL model (Thissen 1982) can be formed in **mirt** because it is a highly nested version of the M4PL model with equality constraints for the slope parameters.

3.1. A simple non-adaptive GUI example

The **mirtCAT** package generates interactive GUIs to run CATs by providing inputs to the `mirtCAT()` function. When generating a GUI, the `mirtCAT()` function requires a `data.frame` object containing questions, response options, and output types. When only a `data.frame` is supplied, a non-adaptive test (i.e., a survey) will be initialized because no IRT parameters were defined for selecting the items adaptively.

Currently, the required inputs names in the `data.frame` object are:

- *Question* – A character vector containing all the question stems.
- *Option.#* – Possible response options for each item, where # corresponds to the specific category. For instance, a test with 4 unique response options for each item would contain the columns "Option.1", "Option.2", "Option.3", and "Option.4". If some items have fewer categories than others then `NA` placeholders must be used to omit the unused options.
- *Type* – Indicates the type of response input to use from the **shiny** package. The supported types are: `"radio"` for radio buttons, `"select"` for a pull-down box for selecting inputs, `"text"` for requiring typed user input, `"checkbox"` for collecting multiple checkboxes of responses for each item, `"slider"` for slider-style inputs, and `"none"` when only an item stem should be presented.
- *Answer* or *Answer.#* – (Optional) A character vector (or multiple character vectors) indicating the scoring key for items that have correct answers. If there is no correct answer for a question then `NA` values must be specified as placeholders. When a `checkbox` type is used with this input then responses are scored according to how many matches were selected.
- *Stem* – (Optional) A character vector of absolute or relative paths pointing to external markdown (.md) or HTML (.html) files, which can be used as item stems. `NA`s are used if the item has no corresponding file.
- **...** – Additional optional argument inputs that are passed to the associated **shiny** construction functions. For the `"slider"` input, however, a column for the `"min"`, `"max"`, and `"step"` arguments must be defined.

When generating surveys with **mirtCAT**, only the *Type*, *Question*, and *Option* inputs are typically required. If questions are to be scored in real time (as they generally are in ability measuring CATs) then a suitable *Answer* vector must be supplied. Finally, if specific graphical stimuli should be included then the paths pointing to the item-stem files must be included in the *Stem* input.

Before generating a real-time CAT GUI, it is informative to first generate a simple survey to highlight the fundamental `mirtCAT()` inputs. For example, say that a researcher wishes to build a small survey with only three rating scale items, where each item contains five rating-scale options ranging from "Strongly Disagree" to "Strongly Agree". Initializing the HTML interface to collect responses can be accomplished with the following code:

```
R> library("mirtCAT")
R> options <- matrix(c("Strongly Disagree", "Disagree", "Neutral",
```

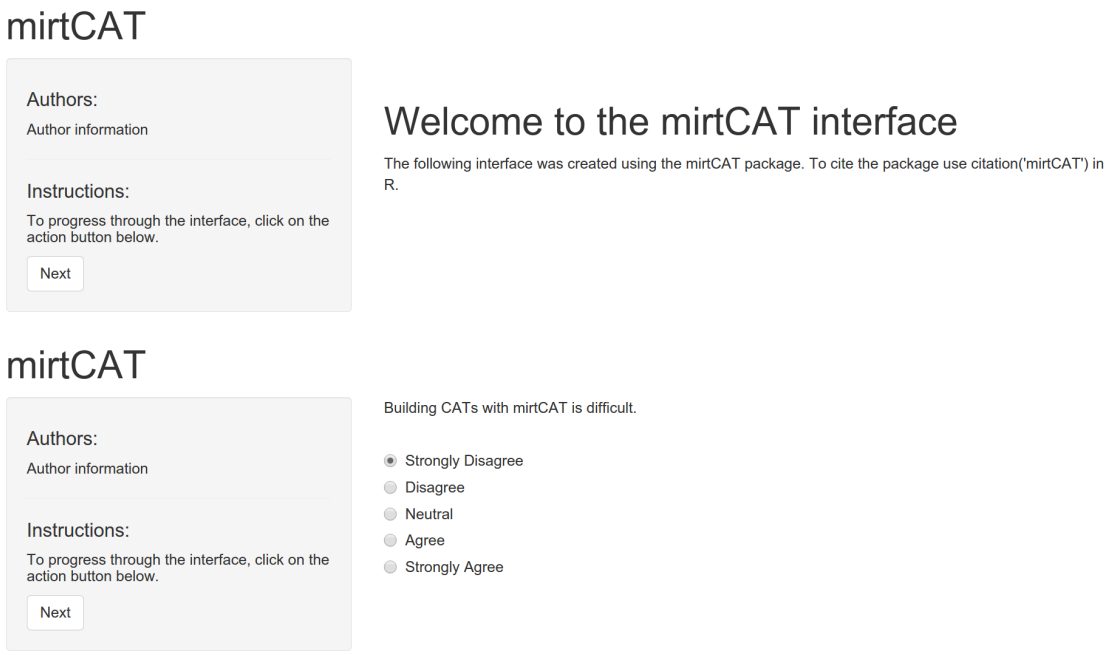

Figure 1: Screen-captures of the interface generated with **mirtCAT** in full-screen mode.

```
+    "Agree", "Strongly Agree"), nrow = 3, ncol = 5, byrow = TRUE)
R> questions <- c("Building CATs with mirtCAT is difficult.",
+    "mirtCAT requires a substantial amount of coding.",
+    "I would use mirtCAT in my research.")
R> df <- data.frame(Question = questions, Option = options, Type = "radio")
R> results <- mirtCAT(df = df)
```

When defining the *Option* input in a `data.frame`, unique names are automatically generated and create the suitable labels Option.1, Option.2, Option.3, Option.4, and Option.5; this is how the `data.frame()` function manages ambiguous labels by default. After calling the `mirtCAT()` function, an interface is generated and embedded within the operating system's default web browser. Figure 1 depicts screen-captures of the default GUI for the initial page and for the first prompted item page.

After the GUI has been generated, the survey will continue to administer each of the defined items until the item bank has been exhausted. Upon completion, the interface will disconnect from the web browser and the R session that was previously suspended for the duration of the GUI will become active again. If the session was assigned to an object, as it was above, the R work-space will contain the saved responses from the GUI. The next section elaborates on the structure of the **mirtCAT** package in much more detail, and demonstrates the flexibility of building tailored interfaces for various MCAT designs.

3.2. mirtCAT package details

The `mirtCAT()` function has inputs broken into design, GUI, item selection, and implementation criteria by supplying the following inputs

Arguments	Description	Possible inputs
`df`	Named `data.frame` containing questions, options, answers, output types, and graphical stem locations.	`data.frame` or missing.
`mo`	Single group object defined by the **mirt** package. This is required if the test is to be scored.	Object of class `SingleGroupClass` or missing.
`method`	Character string for selecting the method of predicting $\hat{\boldsymbol{\theta}}$ values.	`"MAP"`, `"EAP"`, `"ML"`, `"WLE"`, `"EAPsum"`, `"fixed"`
`criteria`	Adaptive and non-adaptive item selection criteria to be used. Some inputs are available exclusively for unidimensional or multidimensional tests.	`"seq"`, `"random"`, `"MI"`, `"MEPV"`, `"MLWI"`, `"MPWI"`, `"MEI"`, `"IKL"`, `"IKLP"`, `"IKLn"`, `"IKLPn"`, `"Drule"`, `"DPrule"`, `"Erule"`, `"EPrule"`, `"Trule"`, `"TPrule"`, `"Arule"`, `"APrule"`, `"Wrule"`, `"WPrule"`, `"KL"`, `"KLn"`
`start_item`	Numeric value or character string indicating which item is to be administered first. Default is 1 to administer the first item in the test. If a suitable character string is supplied instead, the initial item will be selected from the options available in `criteria`.	Numeric scalar or character string.
`local_pattern`	If `df` was supplied, a character matrix (with one row) specifying how a participant responded to the test, otherwise an integer matrix (with potentially more than one row). If supplied, the GUI will not be generated and instead the MCAT results will be evaluated off-line.	Character or integer matrix.
`design_elements`	A logical value indicating whether a list containing the design objects should be returned.	Logical scalar.
`cl`	An optional cluster object defined with the **parallel** package. Used for simulation designs that should be run in parallel.	Cluster object defined by `parallel::makeCluster()`.
`...`	Additional arguments to be passed to function such as `mirt::fscores()`.	–

Table 1: Selection of inputs for `mirtCAT()`.

```
mirtCAT(df, mo, method = "MAP", criteria = "seq", start_item = 1,
  local_pattern = NULL, design_elements = FALSE, cl = NULL,
  design = list(), shinyGUI = list(), preCAT = list(), ...)
```

A selection of important arguments for `mirtCAT()` is displayed in Table 1. Three list arguments (`preCAT`, `design`, and `shinyGUI`) are omitted from this table because they have special design and structural properties for the GUI and CAT design. These list objects define various characteristics about the test, persons, or CAT design, and modify elements of the GUI

generated with the **shiny** package. Tables 2 and 3 describe the possible inputs for these lists at a superficial level, and more detailed descriptions are available in the package help files.

The `mo` input contains the IRT models with their associated parameters, where the required '`mo`' object can be defined using multiple approaches. For instance, if test constructors have a suitable dataset that can be used to calibrate the IRT parameters directly then obtaining an estimated model from the `mirt()` function is one possible and straightforward approach. Following convergence of the item parameters estimated from **mirt**, the model object may be readily passed to the `mo` input to define the CAT parameters for real-time GUIs or Monte Carlo simulation work.

If, on the other hand, calibration data are not available, but population parameters are known a priori (as is the case for Monte Carlo simulation designs), then a suitable `matrix` or `data.frame` can be passed to the `generate.mirt_object()` function in **mirtCAT** to create a suitable '`mo`' object. The column names in the matrix input correspond to the item parameter names used in **mirt**, which can be located in the package's help files (see `help("mirt")` for details). For example, given a 10-item dichotomous test, if the developer wishes to set the slopes equal to [1.0, 1.2, 0.9, 0.8, 1.1, 1.2, 0.8, 0.7, 0.5, 1.0], intercepts to [−1.0, 1.5, 0.0, 0.5, −0.5, −1.0, 0.0, 0.1, 1.1, −0.2], lower-bound parameters to 0.2, and organize the latent trait parameters to have a prior distribution of $N(0, 2)$, then this can be accomplished with

```
R> a <- c(1.0, 1.2, 0.9, 0.8, 1.1, 1.2, 0.8, 0.7, 0.5, 1.0)
R> d <- c(-1.0, 1.5, 0.0, 0.5, -0.5, -1.0, 0.0, 0.1, 1.1, -0.2)
R> g <- rep(0.2, 10)
R> pars <- data.frame(a1 = a, d = d, g = g)
R> lc <- matrix(2)
R> mirt_object <- generate.mirt_object(pars, itemtype = "3PL",
+    latent_covariance = lc)
R> coef(mirt_object, simplify = TRUE)

$items
         a1    d   g u
Item.1  1.0 -1.0 0.2 1
Item.2  1.2  1.5 0.2 1
Item.3  0.9  0.0 0.2 1
Item.4  0.8  0.5 0.2 1
Item.5  1.1 -0.5 0.2 1
Item.6  1.2 -1.0 0.2 1
Item.7  0.8  0.0 0.2 1
Item.8  0.7  0.1 0.2 1
Item.9  0.5  1.1 0.2 1
Item.10 1.0 -0.2 0.2 1

$means
F1
 0

$cov
```

Arguments	Possible inputs	Description
`preCAT`	`min_items`	Minimum number of items to use before starting MCAT. Default is 0.
	`max_items`	Maximum number of items to use before starting MCAT. Default is 0 (disabled).
	`criteria`	Item selection criteria (see Table 1). Default is `"random"`.
	`method`	$\hat{\boldsymbol{\theta}}$ prediction method (see Table 1). Default is `"MAP"`.
	`response_variance`	Logical; if there is variability in the responses (such that ML estimation can theoretically succeed) then stop the preCAT stage early? Default is `FALSE`.
`design`	`min_SEM`	Minimum standard error for each $\hat{\boldsymbol{\theta}}$ value. Default is 0.3.
	`delta_thetas`	Termination criteria for change in estimated trait values, $\Delta = \lvert\hat{\boldsymbol{\theta}}_j - \hat{\boldsymbol{\theta}}_{j+1}\rvert$. Default is 0, which disables this criteria.
	`theta.start`	$\hat{\boldsymbol{\theta}}$ starting values. Default is a vector of 0's.
	`min_items`	Minimum number of items to administer. Default is 1.
	`max_items`	Maximum number of items to administer. Default is the length of the test.
	`quadpts`	Number of quadrature points to use for numerical integration of each latent trait. Default is 61.
	`theta_range`	Integration range (if required). Default is `c(-6,6)`.
	`weights`	Weights to use when with weighted selection criteria such as `criteria = "Wrule"` (also applies to the T-rule and A-rule). Default uses equal weights for each trait.
	`KL_delta`	Δ parameter required for specifying the evaluation range of the Kullback-Leibler criteria.
	`content`	An optional character vector indicating the type of content measured by an item.
	`content_prop`	An optional named numeric vector indicating the distribution of item content proportions. A `content` vector must also be supplied to indicate the item membership.
	`classify`	A numeric vector indicating cut-off values for classification above or below some prior thresholds.
	`classify_CI`	A numeric vector (between 0 and 1) indicating the confident intervals used to classify individuals as above or below values in `classify`.
	`exposure`	Can be two types of vectors. If a numeric vector between 0 and 1 then the Sympson-Hetter exposure control is used, otherwise if an integer vector where all values are greater than or equal to 1 the optimal selection criteria are sampled from given the supplied sequence.
	`constraints`	A named list specifying various item selection constraints useful to control how items are administered or scored.
	`customNextItem`	A function to be used when the selection of the items should be completely customized instead of using the fixed selection methods provided. For further details refer to the documentation for the `findNextItem` function.

Table 2: Pre-CAT and design-based inputs for `mirtCAT()`.

```
   F1
F1  2
```

Arguments	Possible inputs	Description
`shinyGUI`	`title`	A character vector for the title of the GUI.
	`authors`	A character vector for the authors of the GUI. Can be an empty string to omit the author information.
	`instructions`	A three part character vector indicating how to progress through the GUI.
	`firstpage`	A list containing elements defined using **shiny** to be displayed on the first page.
	`begin_message`	Text to display on the page prior to beginning the CAT
	`demographics`	A list of person information page used in the GUI for collecting demographic information generated using tools from the **shiny** package.
	`demographics_inputIDs`	A character vector if IDs required if a custom demographics input is supplied.
	`max_time`	Maximum amount of time to run the test (in seconds). Default is `Inf`, therefore no time limit.
	`temp_file`	An optional file location to temporarily save responses to the disc while the GUI is running.
	`resume_file`	If `temp_file` was used, a path that allows the CAT to resume with the previously saved file
	`lastpage`	A function that returns a list of HTML elements for **shiny** that are to be displayed on the last page.
	`css`	A character string defining Cascading Style Sheet (CSS) arguments to modify the look and feel of the HTML elements.
	`forced_choice`	A logical value indicating whether respondents are required to provide response to each item (for use in surveys only). Default is `TRUE`.
	`stopApp`	A logical value whether `shiny::stopApp()` should be executed when the interface is complete. Default is `TRUE`.
	`ui`	A function used to redefine the graphical user interface using functions from **shiny**. If omitted the default interface is used.

Table 3: GUI inputs for `mirtCAT()`.

The default hyper-parameters in `generate.mirt_object()` are assumed to be from a standard multivariate normal distribution; however, these defaults may be overwritten (as they were above).

Several methods for predicting $\hat{\boldsymbol{\theta}}$ scores are available through the `fscores()` function in **mirt** by supplying an appropriate character vector to the `method` argument. Namely, the estimation method can be selected to fix estimates at their previous values, estimate traits using ML, MAP, EAP, EAP for sum scores, and WLE, or estimate values using imputation variants of these estimators if an asymptotic parameter covariance matrix was computed beforehand. The hyper-parameters for the prior distributions of $\boldsymbol{\theta}$ are obtained from the internal '`GroupPars`' element in `mo`. For all Bayesian prediction methods, `fscores()` contains a specialized `custom_den` argument for users to define a customized density function if they wish to supply their own prior distribution function. The aforementioned prediction methods are available in both the pre-CAT and CAT stages; however, users should bear in mind that

methods which can handle extreme response patterns (such as MAP) may be beneficial in the pre-CAT stage.

There are multiple item selection `criteria` available in **mirtCAT**, some of which are only applicable to unidimensional or multidimensional models. Criteria applicable to both unidimensional and multidimensional adaptive tests are the `"KL"` and `"KLn"` method for the point-wise Kullback-Leibler divergence and the point-wise Kullback-Leibler with a decreasing delta value ($\Delta \cdot \sqrt{n}$, where n is the number of items previous answered), respectively, `"IKLP"` and `"IKL"` for the integration based Kullback-Leibler criteria with and without the prior density weight, and `"IKLn"` and `"IKLPn"` for the $\sqrt{n}$ sequentially weighted counter-parts of the integration criteria (Chang and Ying 1996). Possible inputs for unidimensional adaptive tests include `"MI"` for the maximum information criteria, `"MEPV"` for minimum expected posterior variance, `"MLWI"` for maximum-likelihood with weighted information, `"MPWI"` for maximum posterior weighted information, and `"MEI"` for maximum expected information (see Magis and Raîche 2012, and the references therein for further elaboration of these methods).

Possible inputs for multidimensional adaptive tests include the `"Drule"` for the maximum determinant of the information matrix, `"Trule"` for the maximum (weighted) trace of the information matrix, `"Arule"` for the minimum (weighted) trace of the asymptotic covariance matrix, `"Erule"` for the minimum eigenvalue of the information matrix, and `"Wrule"` for the weighted information criteria. The multivariate selection criteria have posterior weighted analogues for Bayesian selection, which are available by passing `"DPrule"`, `"TPrule"`, `"EPrule"`, and `"WPrule"`, where the "P" indicates the use of a prior distribution. Finally, non-adaptive selection methods include the sequential (`"seq"`) and random (`"random"`) criteria, which can be used in both adaptive and non-adaptive tests.

3.3. Auxiliary functions

Upon completion of the `mirtCAT()` function, an S3 object of class `'mirtCAT'` is returned and contains information about the raw and scored response pattern, person demographics supplied in the survey, order in which the items were administered, estimation history, and final trait estimates. Three generic S3 methods, `print()`, `summary()`, and `plot()`, have been defined to help summarize the returned object. The `print()` method will display the number of items administered and, if `mirtCAT()` was supplied suitable item parameters, the final $\hat{\boldsymbol{\theta}}$ and $SE(\hat{\boldsymbol{\theta}})$ estimates. `summary()` will return a list of more detailed information about the raw and scored response patterns, items administered, item response times (in seconds), history of the $\hat{\boldsymbol{\theta}}$ and $SE(\hat{\boldsymbol{\theta}})$ estimates, and so on. Finally, `plot()` will generate figures based on the estimation history of $\hat{\boldsymbol{\theta}}$ and $SE(\hat{\boldsymbol{\theta}})$ (or confidence intervals) to display how the test was progressively scored.

When constructing CATs, developers may wish to experiment with their CAT designs by supplying fixed response patterns. **mirtCAT** eases the construction of plausible response patterns through the `generate_pattern()` function, and allows the CAT interface to be run without generating a GUI by passing an object containing suitable response patterns to `mirtCAT(..., local_pattern)`. For example, a participant with a latent ability score of $\theta = 1$ could create the following response pattern.

```
R> set.seed(1)
R> pattern <- generate_pattern(mirt_object, Theta = 1)
R> pattern
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    1    1    1    0    1    1    1    1     0
```

The `generate_pattern()` function sequentially generates plausible responses for each item in the item pool, and stores these values into a matrix. If `generate_pattern()` were supplied the `df` object with the respective item options then the function would return a character matrix of plausible responses instead of an integer matrix. The `Theta` input to `generate_pattern()` can be either a numeric vector to generate a single response pattern or a matrix of latent trait values to generate a matrix of plausible patterns corresponding to the rows in `Theta`. Supplying a matrix of trait values is especially useful for generating plausible response patterns for Monte Carlo simulation work, as we will observe in Section 5.

After generating one or more response patterns, the matrix is then passed to the `local_pattern` argument to execute the CAT session(s) off-line. As a simple example of how one might use these response patterns, the following CAT was designed to select all items with the maximum information (`"MI"`) selection criteria.

```
R> result <- mirtCAT(mo = mirt_object, local_pattern = pattern,
+    start_item = "MI", criteria = "MI")
R> print(result)

 n.items.answered   Theta_1 SE.Theta_1
               10 0.3355295  0.7412856

R> summary(result)

$final_estimates
            Theta_1
Estimates 0.3355295
SEs       0.7412856

$raw_responses
 [1] "1" "2" "1" "2" "2" "2" "1" "2" "2" "2"

$scored_responses
 [1] 0 1 0 1 1 1 0 1 1 1

$items_answered
 [1]  5  2 10  3  4  6  1  7  8  9

$thetas_history
           Theta_1
 [1,]  0.000000000
 [2,] -0.548189757
 [3,] -0.263133573
 [4,] -0.596607706
 [5,] -0.315643850
```

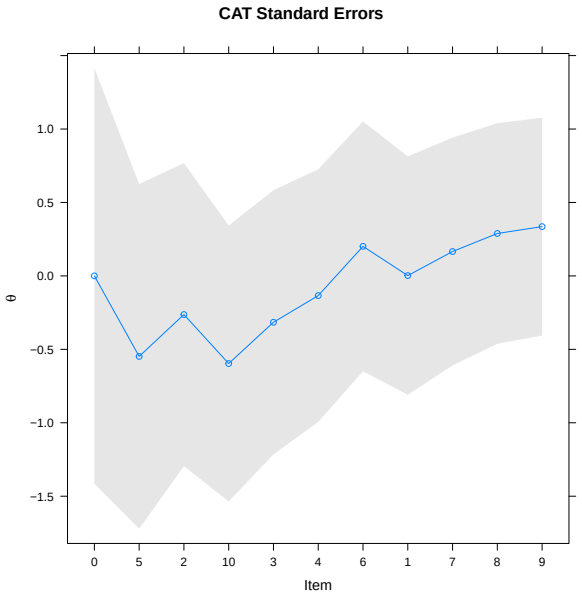

Figure 2: Output of `plot()` method for '`mirtCAT`' objects.

```
 [6,] -0.134086560
 [7,]  0.201131905
 [8,]  0.002223685
 [9,]  0.165947607
[10,]  0.288851261
[11,]  0.335529460

$thetas_SE_history
        Theta_1
 [1,] 1.4142136
 [2,] 1.1733174
 [3,] 1.0314490
 [4,] 0.9390705
 [5,] 0.8992738
 [6,] 0.8600807
 [7,] 0.8499782
 [8,] 0.8114719
 [9,] 0.7753160
[10,] 0.7507978
[11,] 0.7412856
```

```
R> plot(result)
```

The plot is shown in Figure 2.

Administering an adaptive test using an item bank with only 10-items clearly is not optimal for accurately measuring $\theta = 1$, however this example is only intended to demonstrate how

the output is summarized. After calling the `summary(result)` function, a list containing various CAT elements is returned. The `$items_answered` and `$scored_responses` elements together indicate that item five was answered first with scored response 1, followed by item six with a scored response of 0, then item ten with a scored response of 1, and so on until item nine was chosen last by default with a scored response of 1. The `$raw_responses` element from the `summary()` output indicates which category was selected in the GUI or simulated response pattern; for off-line analyses this element can largely be ignored and simply indicates placeholder categories. The corresponding $\hat{\theta}$ and $SE(\hat{\theta})$ estimates are shown in the `$thetas_history` and `$thetas_SE_history` elements, and demonstrate how the ability estimates, and their respective standard errors, successively change after each item is administered.

At this point is useful to note the connection with **mirt**, which thus far has silently performed all the computations of the $\hat{\theta}$ and $SE(\hat{\theta})$ estimates. The list of outputs returned by `summary(result, sort = FALSE)` can readily be used by functions in the **mirt** package by selecting various elements from the list output. More specifically, the unsorted `$scored_responses` element can be added to a calibration dataset (if the parameters were previously estimated) by simple use of the `rbind()` function. Including new response data to the original calibration dataset can be useful when recalibrating the parameter estimates at a later time. Additionally, the unsorted response pattern could be passed into `fscores(..., response.pattern = pattern)` to further examine what the $\hat{\theta}$ would have been given alternative prediction methods. For instance, if the above response pattern were to be estimated with the ML prediction method then the estimates would be

```
R> responses <- summary(result, sort = FALSE)$scored_responses
R> fscores(mirt_object, response.pattern = responses, method = "ML")

     Item.1 Item.2 Item.3 Item.4 Item.5 Item.6 Item.7 Item.8 Item.9
[1,]      0      1      1      1      0      1      1      1      1
     Item.10        F1     SE_F1
[1,]       0 0.4609681 0.8596507
```

Because all items were administered, the unsorted response pattern is identical to the pattern generated from `generate_pattern()`. If items were not responded to due to early termination of the CAT then `NA` values would be present in the items containing no observations.

4. Single case MCAT example

In this section, an MCAT design and graphical user interface are constructed using the code located in Appendix A. The questions and item parameters were arranged to emulate how a multidimensional mathematical achievement test with cross-factor loadings and correlated latent traits could be managed. The item bank consisted of 120 items in total, and a `data.frame` object with the questions and answers was included. The first 30 items were constructed to measure only a hypothetical "Addition" trait, while the last 30 items measured only on a "Multiplication" trait. The middle 60 items were evenly split to contain a mix of the Addition and Multiplication slopes. However, the first half of these items were designed to relate more to the Addition trait (contained larger slopes), while the last 30 items were designed to

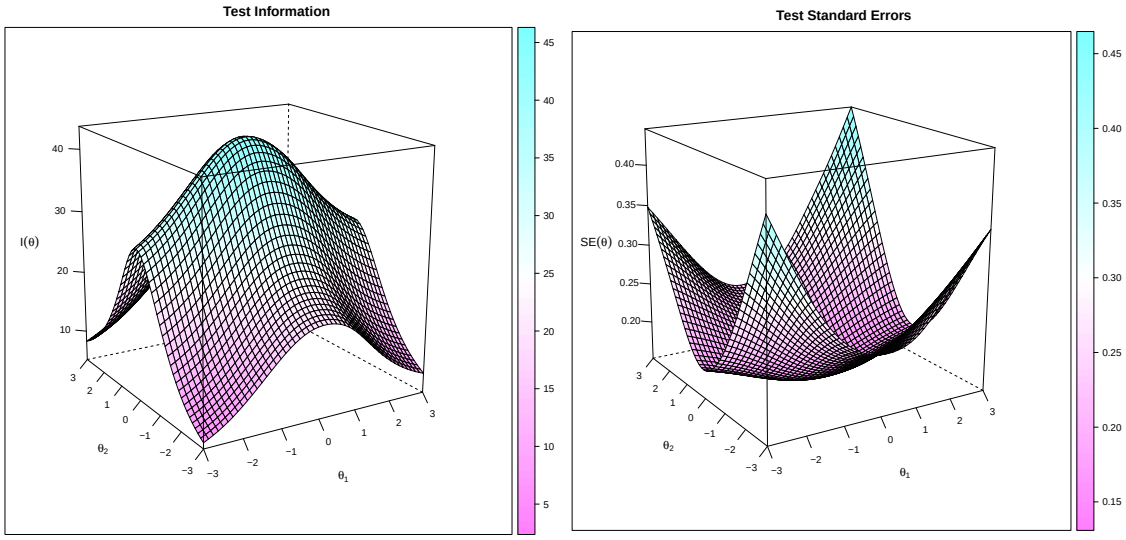

Figure 3: Plots of test implied response functions from models estimated in the **mirt** package.

relate more to the Multiplication trait. The expected information and standard error plots below indicate that respondents with abilities closer to the center of the distributions will be measured with the most accuracy, while those in the more extreme ends of the ability range will be measured with much less precision.

```
R> plot(mod, type = "info", theta_lim = c(-3, 3))
R> plot(mod, type = "SE", theta_lim = c(-3, 3))
```

The resulting plots are shown in Figure 3.

Given the objects defined in Appendix A, we first generate a plausible observed response pattern for a participant with abilities $\boldsymbol{\theta} = [-0.5, 0.5]$.

```
R> set.seed(1)
R> pat <- generate_pattern(mo = mod, Theta = c(-0.5, 0.5), df = df)
R> head(pat)

[1] "145" "195" "200" "232" "207" "175"
```

The character responses indicates that, among the options in `df[1, ]`, the category pertaining to the option `"145"` was selected as the correct answer, `"195"` was selected for the second item among the possible options in `df[2, ]`, and so on for the remaining 118 items. To determine how the MCAT session would behave if each item were administered in sequence, the `min_items` argument could be increased to ensure that all items are selected; alternatively, the `min_SEM` input could be decreased to a much smaller value to accomplish the same goal.

```
R> result <- mirtCAT(df = df, mo = mod, local_pattern = pat,
+    design = list(min_items = 120))
R> print(result)
```

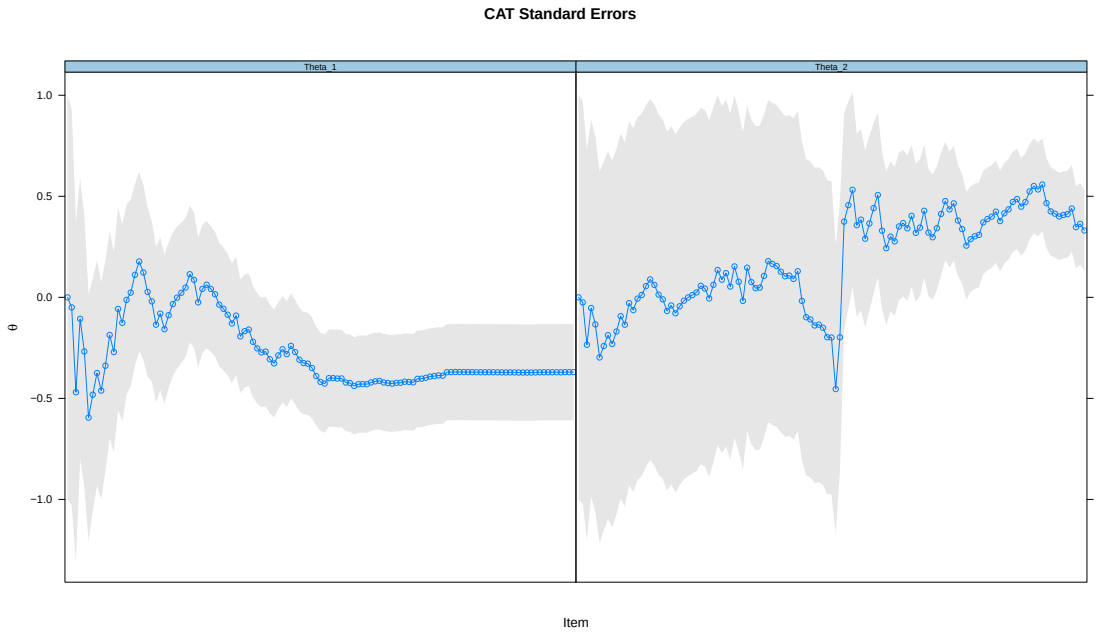

Figure 4: Estimated θ values and standard errors in dependence of items administered.

```
n.items.answered    Theta_1   Theta_2 SE.Theta_1 SE.Theta_2
             120 -0.3700409 0.3307961  0.2382641  0.1991176
```

```
R> plot(result, scales = list(x = list(at = NULL))
```

The plot is shown in Figure 4.

Some initial observations can be made from inspecting the graphical output generated by `plot(result)`. By design, the test parameters were simulated to almost exclusively measure the Addition trait for the first 60 items, and the consequences of this are clearly seen in the ability estimates and their respective standard errors. The standard errors for `Theta_1` rapidly decreased in the first half of the test, while the standard errors for `Theta_2` stayed roughly the same until the second half of the test began[3]. As well, the point estimates for `Theta_1` were able to move closer towards the population value of -0.5 in the first half, but only after the second half of the test began does `Theta_2` begin to move towards the population value of 0.5. Although not shown above, the results from `summary(results)` revealed that both traits were measured with a standard error less than 0.4 after the 73rd item was administered.

Administering all items in an item pool is generally not desirable in real testing situations when the item parameters are available. Therefore, we will instead implement a multidimensional adaptive test design to select items that are more suitable for the observed response pattern. First, we choose an MCAT item selection criteria to help maximally increase the information in both traits simultaneously. Because the latent traits are deemed to be of equal importance

[3]Had the latent traits been orthogonal the second trait estimates and standard errors would have remained completely stationary. However, because the traits are slightly correlated, information about the first trait will provide indirect information about the second trait.

in this example, the use of the D-rule for selecting items is a reasonable choice. Next, we set the stopping criteria for the standard error of measurement to 0.4 for all traits. The response pattern previously simulated is then reanalyzed in `mirtCAT()` with

```
R> set.seed(1234)
R> MCATresult <- mirtCAT(df = df, mo = mod, criteria = "Drule",
+    local_pattern = pat, start_item = "Drule",
+    design = list(min_SEM = 0.4))
R> print(MCATresult)

 n.items.answered    Theta_1   Theta_2 SE.Theta_1 SE.Theta_2
               18 -0.5315822 0.7334699  0.3922312  0.3525484

R> summary(MCATresult)

$final_estimates
             Theta_1   Theta_2
Estimates -0.5315822 0.7334699
SEs        0.3922312 0.3525484

$raw_responses
 [1] "3" "2" "1" "3" "3" "5" "5" "3" "1" "3" "3" "1" "1" "3" "4" "1" "5" "4"

$scored_responses
 [1] 1 0 0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 0

$items_answered
 [1]  40 118  59  65  41   3  93  31  87  63 107  24  96  82   5  15 110  21

$thetas_history
         Theta_1      Theta_2
 [1,]  0.0000000  0.000000000
 [2,]  0.3300104  0.279113425
 [3,]  0.1417429 -0.280431989
 [4,] -0.3095254 -0.381602240
 [5,] -0.2671319 -0.118300123
 [6,] -0.6102685 -0.162056553
 [7,] -0.3903506 -0.144267168
 [8,] -0.3774346  0.003104662
 [9,] -0.5382688 -0.006176610
[10,] -0.5065322  0.176070232
[11,] -0.4950992  0.384451064
[12,] -0.4864141  0.521754123
[13,] -0.3538263  0.527263358
[14,] -0.3480257  0.626719891
[15,] -0.3465018  0.697100900
[16,] -0.4941311  0.690762595
```

```
[17,] -0.4398394  0.693084295
[18,] -0.4374101  0.737414618
[19,] -0.5315822  0.733469944

$thetas_SE_history
        Theta_1   Theta_2
 [1,] 1.0000000 1.0000000
 [2,] 0.8676229 0.9072747
 [3,] 0.8150130 0.5880719
 [4,] 0.6409551 0.6047584
 [5,] 0.6340877 0.4731553
 [6,] 0.5635720 0.4760173
 [7,] 0.4990944 0.4734596
 [8,] 0.4982264 0.4252720
 [9,] 0.4648866 0.4251890
[10,] 0.4618842 0.3967178
[11,] 0.4609951 0.3806421
[12,] 0.4605138 0.3706390
[13,] 0.4363271 0.3715648
[14,] 0.4364367 0.3653976
[15,] 0.4365725 0.3593208
[16,] 0.4196257 0.3580614
[17,] 0.4045024 0.3584700
[18,] 0.4045165 0.3533521
[19,] 0.3922312 0.3525484
```

For this particular response pattern, the MCAT session was terminated after only 18 items were administered. As can be seen from the summary results above, and the generated plot below, the items were effectively selected so to reduce the standard error estimates more rapidly. Improving the standard errors consequently improves the rate at which the point estimates converge to their population values. When compared to administering items in an ordered sequence, the MCAT with the D-rule criteria was able to obtain the same degree of measurement precision with 55 fewer items. Clearly, even for a small test bank such as the one simulated here, the payoff to implementing MCATs can be quite meaningful compared to more traditional item selection methods (see Figure 5).

```
R> plot(MCATresult)
```

4.1. Customizing GUI elements

To demonstrate how the previous MCAT example can be transformed into a useful GUI, we will now focus on the related graphical inputs required for the `mirtCAT()` function. The code in Appendix A generated character vectors for the questions, options, and answers for 120 items, and placed these values in an objected called `df`. The `df` object can be passed to the `mirtCAT(df = ...)` input, which will generate simple text-based question stems using suitable HTML code. However, in the code below we will include two additional items to demonstrate how more stimulating item stems can be presented. When defining items, test

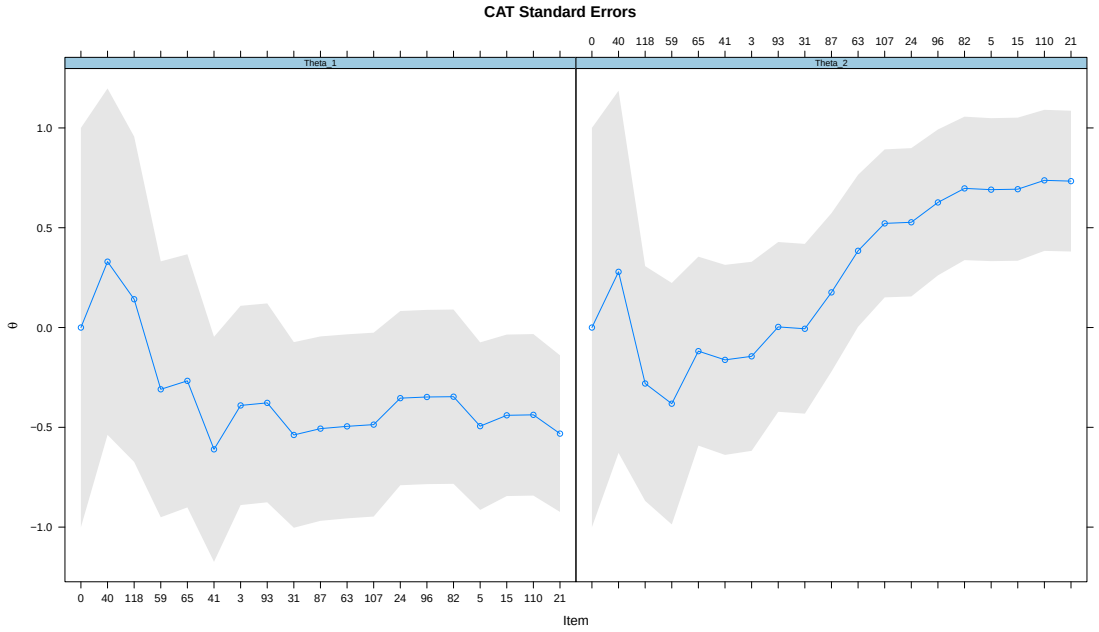

Figure 5: Estimated θ values and standard errors in dependence of items administered.

designers will often wish to present stimuli other than the default text output, and instead include materials in the form of images, tables, maps, and so on. In such cases, the `df` input may include a *Stem* character vector to point to previously defined markdown or HTML files. The following code adds two items to the existing `df` object which point to external HTML files. The `rbind.fill()` function from the **plyr** package (Wickham 2011) is used below to quickly fill in missing values with `NA`s, which can be useful when combining two `data.frame` objects. Screen captures of the graphical item stems can be seen in Figure 6.

```
R> type <- c("radio", "select")
R> questions <- c("", "Which equation in the above table is INCORRECT?")
R> options <- rbind(c("1236", "1238", "1240", "1242", "1244"),
+    c("A)", "B)", "C)", "D)", "E)"))
R> answers <- c("1236", "D)")
R> stem_locations <- c("/path/to/Math-stem.html", "/path/to/Table-stem.html")
R> twoitems <- data.frame(Question = questions, Option = options,
+    Answer = answers, Stem = stem_locations, Type = type,
+    inline = c(TRUE, NA), stringsAsFactors = FALSE)
R> library("plyr")
R> df$inline <- FALSE
R> df2 <- rbind.fill(twoitems, df)
R> GUIresults <- mirtCAT(df = df2, design = list(max_items = 2))
```

The test designer may also wish to include additional CAT design properties such as including a pre-CAT section for collecting item information prior to running the CAT, implementing random sampling exposure control, using a fixed test length, collecting demographic information before the test begins, and so on. The following code demonstrates how to set up

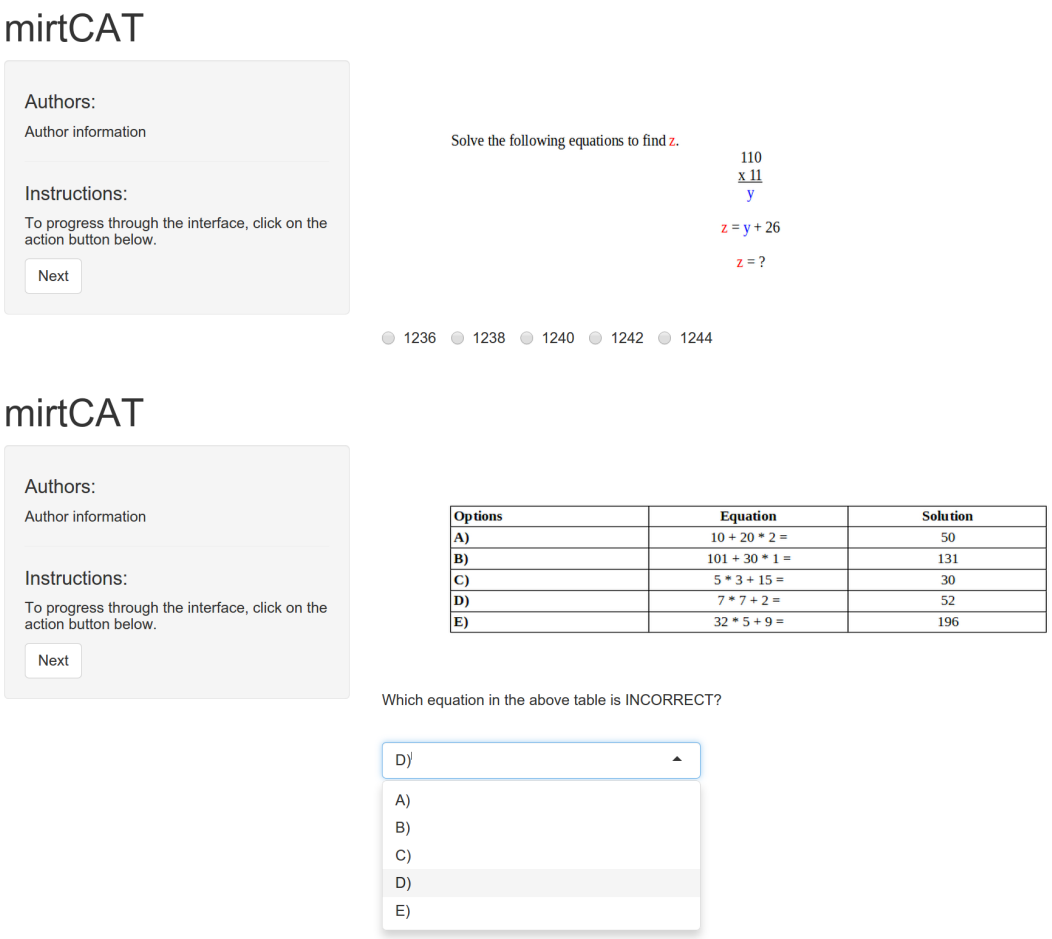

Figure 6: Include external HTML files to add additional content to the item stems.

various design constraints, and also defines more suitable front-end material for the GUI. As is clear from the following code, customizing some GUI elements through the `shinyGUI` input can require users to be familiar with the HTML generating functions defined in the **shiny** package (RStudio Inc. 2014).

```
R> design_list <- list(max_items = 30, min_items = 10, min_SEM = 0.4,
+    exposure = rep(3, 120))
R> preCAT_list <- list(max_items = 5, criteria = "DPrule", method = "EAP")
R> title <- "Example Test"
R> authors <- "I. M. D. Author"
R> firstpage <- list(h2("Example Test"),
+    h5("Please answer each item to the best of your ability.
+      The results of this test will remain completely anonymous
+      and are only used for research purposes."))
R> demographics <- list(textInput(inputId = "occupation",
+      label = "What is your occupation?", value = ""),
+    selectInput(inputId = "gender", label = "Please select your gender.",
```

```
+      choices = c("", "Male", "Female", "Other"), selected = ""))
R> shinyGUI_list <- list(title = title, authors = authors,
+    demographics = demographics,
+    demographics_inputIDs = c("occupation", "gender"),
+    firstpage = firstpage)
R> GUIresults <- mirtCAT(df = df, mo = mod, criteria = "Drule",
+    start_item = "DPrule", shinyGUI = shinyGUI_list,
+    design = design_list, preCAT = preCAT_list)
```

The above MCAT design begins with a pre-CAT stage where a total of five items are selected using the Bayesian D-rule criteria, while the traits are updated using the EAP method. Next, the MCAT begins, and items are selected according to the D-rule criteria. However, instead of selecting the most optimal D-rules in both stages, the top three most optimal items are randomly sampled to determine which item is to be administered next[4]. The total number of items administered are required to be between 10 and 30; however, the test may be terminated early if the standard errors for the traits are both less than 0.4. With respect to the GUI presentation, simple elements such as the title and author names are modified using basic R code, while the first, last, and demographic pages were customized using HTML generating functions from the **shiny** package.

5. Monte Carlo simulations

In this section, functions in the **mirtCAT** package for generating Monte Carlo simulation studies are presented, and the results are compared to existing R packages capable of analyzing CAT designs. The first simulation generates a unidimensional CAT design, and compares the results from **mirtCAT** to the **catIrt** and **catR** packages (version 0.5-0 and 3.4, respectively). The second design generates a two-dimensional MCAT design, but now compares the results to the **MAT** package (version 2.2). Finally, a third simulation design was constructed using **mirtCAT** to generate an MCAT design with mixed-item types, exposure control, content balancing, and a weighted item selection criterion.

5.1. Unidimensional simulation design

A unidimensional item bank was constructed to contain 1000 3PL item response models. The item slope parameters were drawn from a log-normal distribution, $a \sim \log N(0.2, 0.3)$, the intercept parameters were drawn from a standard normal distribution, $d \sim N(0, 1)$, the lower-bound parameters were all set to a constant, $g = 0.2$, the latent trait values were drawn from a standard normal distribution, and $N = 5000$ plausible response patterns were generated given the latent trait values and item parameters. Using the **mirtCAT** package, the plausible response patterns were generated using the following code:

```
R> nitems <- 1000
R> N <- 5000
R> Theta <- matrix(rnorm(N))
R> a <- matrix(rlnorm(nitems, 0.2, 0.3), nitems)
```

[4]When the random sampling exposure control method draws a constant number of options throughout the CAT session this is known as the *randomesque* method (Kingsbury and Zara 1989).

```
R> d <- rnorm(nitems)
R> pars <- data.frame(a1 = a, d = d, g = 0.2)
R> mirt_object <- generate.mirt_object(pars, "3PL")
R> responses <- generate_pattern(mirt_object, Theta = Theta)
```

Analyzing the matrix of response patterns requires passing the object to the `local_pattern` argument in `mirtCAT()`. To allow for comparisons between existent R packages, a compatible CAT design was constructed. The design was organized such that all items were selected using the maximum-information criteria (including the initial item), the $\hat{\theta}$ values were updated using EAP estimation given a standard normal prior, the number of items selected were required to be between 10 and 50, and the CAT was terminated early if $SE(\hat{\theta}) < 0.25$.

When performing Monte Carlo simulation studies, front-end users should consider using multi-core architecture methods. Invoking more than one processor to perform the computations can potentially reduce the estimation times by a factor proportional to the number of cores available. `mirtCAT()` explicitly supports parallel computing by accepting a `cl` argument, where `cl` is a suitable socket-type object to be used by functions in the **parallel** package. Analyzing the previously defined CAT design for each response pattern, while capitalizing on multi-core estimation, can be expressed as

```
R> library("parallel")
R> cl <- makeCluster(detectCores())
R> design <- list(min_SEM = 0.25, min_items = 10, max_items = 50)
R> mirtCAT_results <- mirtCAT(mo = mirt_object, local_pattern = responses,
+    start_item = "MI", method = "EAP", criteria = "MI", design = design,
+    cl = cl))
```

The resulting object returned by `mirtCAT()` is a list containing independent '`mirtCAT`' objects, where each element corresponds to the respective row in the `responses` input. These elements can further be extracted and analyzed using various simulation summary statistics (e.g., bias, root mean-square deviation, correlations, et cetera; see Section 5.3 and the code in Appendix B for a more complete example).

The unidimensional CAT design and generated response patterns were then analyzed using code from the **catIrt** and **catR** packages. **mirtCAT** was executed twice to compare the effect of single versus multi-core architecture (eight processors were selected for the multi-core execution using an Intel i7, 3.40GHz processor; Operating system: Ubuntu, version 16.04 LTS). All final ability estimates correlated equally well with the generated population values ($r \approx 0.9689$), and returned nearly the exact same estimates ($r > 0.9999$). However, where these packages differed was in the estimation time required to complete the simulation. The **catIrt** package was the least efficient at estimating this design, requiring 5733 seconds to complete the simulation (approximately 95 minutes), while **catR** required 1703 seconds (approximately 28 minutes).[5] The **mirtCAT** package, however, required 565 seconds for execution with single-core architecture (9 minutes and 25 seconds) and only 144 seconds (2 minutes and 24 seconds) when using the internally organized multi-core architecture. As is evident from this simulation, multi-core architecture can be highly effective when performing Monte Carlo studies for CATs.

[5]Note that **catR** required a `for()` loop in order to execute the simulated response patterns because the current version does not support multiple response inputs.

5.2. Multidimensional simulation design

The second simulation compared numerical results to the **MAT** package for a simple MCAT design. A relatively large item bank was organized to consist of 1000 M2PL items with two latent traits. The slope and intercept parameters were drawn from the same distribution as in the previous simulation, however the $\boldsymbol{\theta}$ parameters were drawn from a standard multivariate normal distribution with an inter-factor correlation of $r = 0.5$. The MCAT design used the D-rule throughout the session (including the initial item), the test was terminated if more than 30 items were selected or all $SE(\hat{\boldsymbol{\theta}}) < 0.3$, and the trait estimates were computed using MAP estimation with a multivariate normal prior distribution of $MVN(\mathbf{0}, \begin{bmatrix} 1 & \\ .5 & 1 \end{bmatrix})$. The estimation algorithm was exclusively required to be the MAP algorithm because it is the only method supported in **MAT**.

The $\hat{\boldsymbol{\theta}}$ estimates recovered by **MAT** and **mirtCAT** were essentially equivalent ($r > 0.9999$), and correlated equally well with the population $\boldsymbol{\theta}$ values ($r \approx 0.9377$ for the first trait, $r \approx 0.6093$ for the second trait). Estimation efficiency slightly favored the **MAT** package, requiring approximately 259 versus 394 seconds to complete the simulation. Nevertheless, the **MAT** package contains several limitations; namely, the **MAT** package currently only supports MAP estimation of the latent traits with a multivariate normal prior (whereas **mirtCAT** can support any defined prior density function), only supports the M3PL model, has limited support for other CAT related properties (such as content balancing, exposure control, pre-CAT stages, terminating the MCAT according to classification rules, and so on), and contains no public functions to help build customized MCAT designs (the majority of the package is written in self-contained C++ code). Therefore, the package may be of limited use when researchers wish to study more realistic MCATs, when investigating IRT models other than the M3PL model (i.e., a mixture of IRT models), and when developers require tools to build MCATs for real-time applications.

5.3. MCAT simulation with item selection factors

A final simulation was organized to demonstrate how **mirtCAT** can be used for tests with more complex IRT model combinations. An item pool of size 360 was constructed to consist of an even number of M4PL models and MGPCMs (with five response options). The test was organized to have a bi-factor structure with three specific-item traits. Sufficient measurement precision was only required for the general factor, where the specific traits were treated as nuisance factors that were only included to account for inter-item dependencies. The general factor slopes were drawn from a log-normal distribution, $a_g \sim \log N(0.2, 0.3)$, while the specific factor slopes were drawn from $a_s \sim \log N(-1, 0.2)$. Each item was structured such that only one specific trait could influence each item, and each specific trait loaded uniquely on 120 items. For the 180 multidimensional 4PL models, all g and u parameters were set to the constants 0.2 and 0.95, respectively, while the intercepts were drawn from a standard normal distribution, $d \sim N(0, 1)$. The MGPCM intercepts were all drawn from $d_k \sim N(0, 2)$ and sorted from lowest to highest for each item. A matrix of multivariate ability parameters were sampled from a standard multivariate normal distribution with uncorrelated traits, $\boldsymbol{\theta} \sim MVN(\mathbf{0}, \mathbf{I})$. Finally, given the item and person parameters, a total of 1000 plausible response patterns were generated using the `generate_pattern()` function.

The MCAT design was organized to select items that were more informative of the general

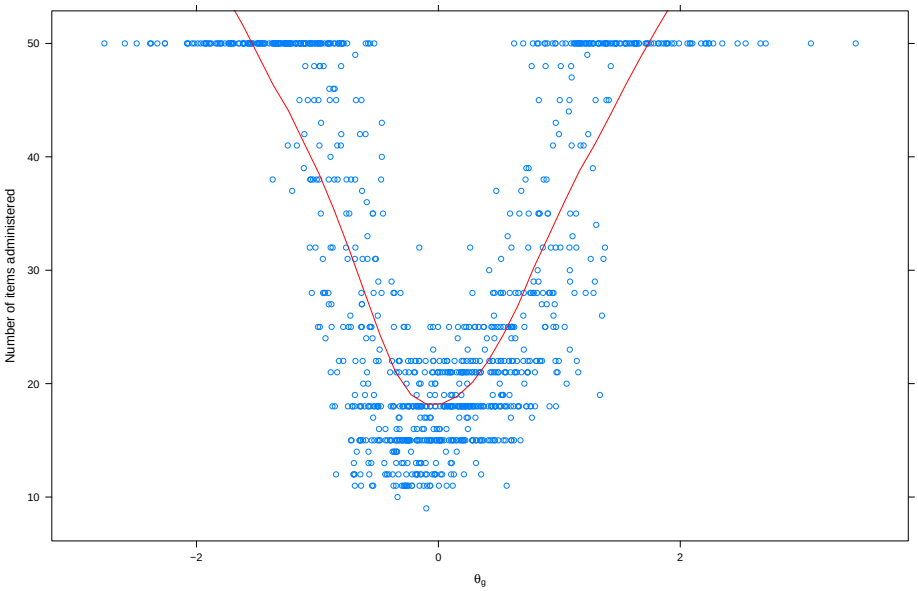

Figure 7: Number of items in dependence of θ_g.

factor by utilizing the W-rule with the weight vector $\mathbf{w} = (0.85, 0.05, 0.05, 0.05)$. The MCAT was terminated when either 50 items were administered or the general factor standard error was less than 0.2. In addition to these selection rules, a content balancing scheme was constructed to ensure that there were more M4PL items administered than MGPCMs (70% versus 30%, respectively). Finally, a Sympson-Hetter exposure control method was included such that if the general factor slopes were too large then the item would have a lower probability of being selected. Specifically, the SH expose control scheme for the i-th item was defined as

$$\text{SH} = \begin{cases} 0.3 & \text{if } a_g > 2.5, \\ 0.6 & \text{if } 2 < a_g \leq 2.5, \\ 0.9 & \text{if } 1 < a_g \leq 2, \\ 1 & \text{otherwise.} \end{cases}$$

The code used to generate and execute this simulation can be located in Appendix B.

In spite of the content balancing and exposure control effects, the MCAT design appeared to effectively recover the generated population trait values for the general factor. The final trait estimates correlated well with the population generating values ($r = 0.9751$) with little bias and variability (bias $= -0.0046$; root mean-square deviation $= 0.2213$). The MCAT was terminated after an average of 33.022 items were administered, and the empirical proportion of M4PL and MGPCM administered were 0.6863 and 0.3137, respectively. As is evident from the figure below, items were more effectively selected from the item bank when the general trait scores were within approximately ± 0.5 of the population mean. When the population values were close to the population mean, the MCATs were able to achieve the $SE(\theta_g) < 0.2$ stopping criteria; however, more extreme population values were not measured accurately by the pool of available items, and instead the MCAT sessions were terminated when the maximum number of items were administered (see Figure 7).

```
R> n_items <- laply(result, function(x) length(x$items_answered))
R> xyplot(n_items ~ Theta[, 1], xlab = expression(theta[g]),
+    panel = function(x, y) {
+      panel.xyplot(x, y)
+      panel.loess(x, y, span = 0.6, col = "red")
+    },
+    ylab = "Number of items administered")
```

6. Discussion

In this article, an R package was introduced for generating interactive graphical interfaces specifically for MCAT and non-MCAT designs. The package provided tools to generate and analyze plausible MCAT responses for use in Monte Carlo simulations, provided functions to summarize MCAT results, and included utilities to plot the estimation history for multiple latent traits. The estimation efficiency was contrasted with existent R packages, and various graphical user interfaces were constructed to demonstrate how real-time MCAT applications can be generated with the **mirtCAT** package. Using the wide selection of IRT models available in the **mirt** package, **mirtCAT** was able to support the construction of flexible unidimensional and multidimensional adaptive test designs containing a mixture of IRT models.

Test developers may choose to execute their MCAT GUIs locally for single computer administrations or deploy their GUIs over the Internet for remote accessibility. Executing the GUIs locally may require password protecting the questions and answer keys, disabling keyboard shortcuts in the Operating System (i.e., `Ctrl + Alt + Delete`, `Alt + F4`, `Alt + Tab`, and so on), and may require saving the relevant R objects and terminating the R session immediately after the testing interface is complete. Executing the MCAT GUIs locally will often offer a more controlled laboratory setting, and generally will help maintain the integrity of the item pool. Therefore, local execution of MCAT GUIs is the most recommended approach. Remote deployment of MCAT GUIs, on the other hand, will require the configuration of a server capable of handling the computations, and may introduce other unwanted issues (such as Internet connection problems, or slower upload and download times).

To date, the only open-source GUI-generating system that has been developed for deploying CATs in real-time has been the **Concerto** project (Kosinski and Rust 2011). **Concerto** uses the **catR** package as the back-end to perform the computations for unidimensional IRT models. However, an unfortunate complication regarding the **Concerto** project is that it is not implemented in R, and instead is executed as a standalone web application where R is used as a computational back-end. This requires the user to obtain extra knowledge regarding how to setup personal servers to collect the response data, requires learning additional web-interface tools outside of the R language, and, after these skills have been acquired, the interface is still currently limited to unidimensional IRT models. With the help of the tools available in **mirtCAT**, projects such as **Concerto** may be further extended to include unidimensional and multidimensional tests, potentially with a mixture of IRT models, in a manner similar to how the **catR** package has been adopted to perform the back-end computations in R.

Future work on **mirtCAT** will largely be driven by users who are interested in utilizing the package tools in their applied research work. However, a number of potential avenues to explore may include: better support for classification CATs, more dynamic GUI elements,

and the inclusion of interactive items responses as the **shiny** framework continues to mature. Currently, interactive items can be included by pointing to raw HTML stems, although these do not directly integrate with the responses to each item. Additionally, more holistic control over content constraints may be included by providing support for so-called shadow testing designs (e.g., Veldkamp and Linden 2002).

mirtCAT is written and manipulated entirely within R, thereby allowing seamless transitioning between data collection using GUIs and further item analysis work with packages such as **mirt**. Users only need to learn basic R code, understand how to define their item parameters with **mirtCAT** or estimate their item parameters with the **mirt** package, and learn how to manipulate the simple tools defined in **mirtCAT** to build a completely automated CAT application for their own research purposes. The package provides intuitive tools to generate interfaces for local or server-sided use, uses **mirt** to perform many of the underlying organizational and computational components, includes powerful Monte Carlo simulation design support for CATs and MCATs, and helps to facilitate the collection of respondent data using adaptive and non-adaptive tests or surveys. When used in conjunction with **mirt**, **mirtCAT** provides a fluid and organized work-flow for test developers to collect, as well as analyze, their important response data.

Acknowledgments

Special thanks to Matthew Sigal, Nicklaus Csuzdi, and two reviewers for offering helpful comments that improved the quality of this manuscript.

References

Bock R, Aitkin M (1981). "Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm." *Psychometrika*, **46**(4), 443–459. doi:10.1007/bf02293801.

Chalmers RP (2012). "**mirt**: A Multidimensional Item Response Theory Package for the R Environment." *Journal of Statistical Software*, **48**(6), 1–29. doi:10.18637/jss.v048.i06.

Chalmers RP (2015). "Extended Mixed-Effects Item Response Models with the MH-RM Algorithm." *Journal of Educational Measurement*, **52**(2), 200–222. doi:10.1111/jedm.12072.

Chalmers RP, Counsell A, Flora DB (2016). "It Might Not Make a Big DIF: Improved Differential Test Functioning Statistics that Account for Sampling Variability." *Educational and Psychological Measurement*, **76**(1), 114–140. doi:10.1177/0013164415584576.

Chalmers RP, Flora D (2014). "Maximum-Likelihood Estimation of Noncompensatory IRT Models with the MH-RM Algorithm." *Applied Psychological Measurement*, **38**(5), 339–358. doi:10.1177/0146621614520958.

Chang HH, Ying Z (1996). "A Global Information Approach to Computerized Adaptive Testing." *Applied Psychological Measurement*, **20**(3), 213–229. doi:10.1177/014662169602000303.

Choi SW, King D (2014). ***MAT**: Multidimensional Adaptive Testing*. R package version 2.2, URL https://CRAN.R-project.org/package=MAT.

Eggen T (1999). "Item Selection in Adaptive Testing with the Sequential Probability Ratio Test." *Applied Psychological Measurement*, **23**(3), 249–261. doi:10.1177/01466219922031365.

Fisher R (1925). "Theory of Statistical Estimation." *Mathematical Proceedings of the Cambridge Philosophical Society*, **22**(5), 700–725. doi:10.1017/s0305004100009580.

Gibbons R, Darrell R, Hedeker D, Weiss D, Segawa E, Bhaumik DK, Kupfer D, Frank E, Grochocinski V, Stover A (2007). "Full-Information Item Bifactor Analysis of Graded Response Data." *Applied Psychological Measurement*, **31**(1), 4–19. doi:10.1177/0146621606289485.

Gibbons R, Hedeker D (1992). "Full-Information Item Bi-Factor Analysis." *Psychometrika*, **57**(3), 423–436. doi:10.1007/bf02295430.

Kingsbury G, Zara A (1989). "Procedures for Selecting Items for Computerized Adaptive Testing." *Applied Measurement in Education*, **2**, 359–375. doi:10.1207/s15324818ame0204_6.

Kingsbury G, Zara A (1991). "A Comparison of Procedures for Content-Sensitive Item Selection in Computerized Adaptive Tests." *Applied Measurement in Education*, **4**(3), 241–261. doi:10.1207/s15324818ame0403_4.

Kosinski M, Rust J (2011). *The Development of **Concerto**: An Open Source Online Adaptive Testing Platform*. Paper presented at the International Association for Computerized Adaptive Testing, Pacific Grove.

Kullback S, Leibler R (1951). "On Information and Sufficiency." *The Annals of Mathematical Statistics*, **22**(1), 79–86.

Linden W (1999). "Multidimensional Adaptive Testing with a Minimum Error-Variance Criterion." *Journal of Educational and Behavioral Statistics*, **24**, 398–412. doi:10.3102/10769986024004398.

Linden W, Glas C (eds.) (2010). *Elements of Adaptive Testing*. Springer-Verlag.

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

Magis D, Raîche G (2012). "Random Generation of Response Patterns under Computerized Adaptive Testing with the R Package **catR**." *Journal of Statistical Software*, **48**(8), 1–31. doi:10.18637/jss.v048.i08.

Maydeu-Olivares A, Hernández A, McDonald RP (2006). "A Multidimensional Ideal Point Item Response Theory Model for Binary Data." *Multivariate Behavioral Research*, **41**(4), 445–471. doi:10.1207/s15327906mbr4104_2.

McBride J, Martin J (1983). "Reliability and Validity of Adaptive Ability Tests in a Military Setting." In DJ Weiss (ed.), *New Horizons in Testing*, pp. 223–236. Academic Press, New York.

Mulaik S (2010). *Foundations of Factor Analysis*. 2nd edition. Chapman & Hall, Boca Raton.

Mulder J, Linden W (2009). "Multidimensional Adaptive Testing with Optimal Design Criteria for Item Selection." *Psychometrika*, **74**(2), 273–296. doi:10.1007/s11336-008-9097-5.

Muraki E (1992). "A Generalized Partial Credit Model: Application of an EM Algorithm." *Applied Psychological Measurement*, **16**, 159–176. doi:10.1177/014662169201600206.

Muraki E, Carlson E (1995). "Full-Information Factor Analysis for Polytomous Item Responses." *Applied Psychological Measurement*, **19**, 73–90. doi:10.1177/014662169501900109.

Nydick S (2014). ***catIrt**: An R Package for Simulating IRT-Based Computerized Adaptive Tests*. R package version 0.5-0, URL https://CRAN.R-project.org/package=catIrt.

Reckase MD (2009). *Multidimensional Item Response Theory*. Springer-Verlag, New York. doi:10.1007/978-0-387-89976-3.

Revuelta J (1998). "A Comparison of Item Exposure Control Methods in Computerized Adaptive Testing." *Journal of Educational Measurement*, **35**(4), 311–327. doi:10.1111/j.1745-3984.1998.tb00541.x.

RStudio Inc (2014). ***shiny**: Web Application Framework for R*. R package version 0.10.0, URL https://CRAN.R-project.org/package=shiny.

Samejima F (1969). *Estimation of Latent Ability Using a Response Pattern of Graded Scores*. Number 17 in Psychometric Monographs. Psychometric Society, Richmond.

Segall D (1996). "Multidimensional Adaptive Testing." *Psychometrika*, **61**(2), 331–354. doi:10.1007/bf02294343.

Suh Y, Bolt D (2010). "Nested Logit Models for Multiple-Choice Item Response Data." *Psychometrika*, **75**(3), 454–473. doi:10.1007/s11336-010-9163-7.

Sympson J (1977). "A Model for Testing with Multidimensional Items." In D Weiss (ed.), *Proceedings of the 1977 Computerized Adaptive Testing Conference*, pp. 82–98. Computerized Adaptive Testing Laboratory, Minneapolis.

Thissen D (1982). "Marginal Maximum Likelihood Estimation for the One-Parameter Logistic Model." *Psychometrika*, **47**(2), 175–186. doi:10.1007/bf02296273.

Thissen D, Cai L, Bock R (2010). "The Nominal Categories Item Response Model." In M Nering, R Ostini (eds.), *Handbook of Polytomous Item Response Theory Models: Development and Applications*, pp. 43–75. New York: Taylor & Francis.

Thissen D, Pommerich M, Billeaud K, Williams V (1995). "Item Response Theory for Scores on Tests Including Polytomous Items with Ordered Responses." *Applied Psychological Measurement*, **19**(1), 39–49. doi:10.1177/014662169501900105.

Veldkamp B, Linden W (2002). "Multidimensional Adaptive Testing with Contsraints on Test Content." *Psychometrika*, **67**(4), 575–588. doi:10.1007/bf02295132.

Veldkamp B, Linden W (2008). "Implementing Sympson-Hetter Item-Exposure Control in a Shadow-Test Approach to Constrained Adaptive Testing." *International Journal of Testing*, **8**(3), 272–289. doi:10.1080/15305050802262233.

Wainer H, Dorans N (eds.) (2000). *Computerized Adaptive Testing: A Primer*. 2nd edition. Lawerence Erlbaum Associates.

Warm T (1989). "Weighted Likelihood Estimation of Ability in Item Response Theory." *Psychometrika*, **54**(2), 427–450. doi:10.1007/bf02294627.

Weiss D (1982). "Improving Measurement Quality and Efficiency with Adaptive Testing." *Applied Psychological Measurement*, **6**(4), 479–492. doi:10.1177/014662168200600408.

Wickham H (2011). "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software*, **40**(1), 1–29. doi:10.18637/jss.v040.i01.

A. Adaptive GUI definitions

Code used to define the GUI for a multidimensional adaptive math ability test with two correlated factors ($r = 0.5$) and 120 items. Intercept parameters (i.e., "easiness" values) were selected based an on arbitrary spacing effect, where if the response options were closer to the true answer then the item was understood to be more difficult.

```
R> library("mirtCAT")
R> set.seed(1234)
R> nitems <- 120
R> itemnames <- paste0("Item.", 1:nitems)
R> a <- matrix(c(rlnorm(nitems/2, 0.2, 0.3), rnorm(nitems/4, 0, 0.3),
+    numeric(nitems/2), rnorm(nitems/4, 0, 0.3),
+    rlnorm(nitems/2, 0.2, 0.3)), nitems)
R> d <- matrix(rnorm(nitems))
R> pars <- data.frame(a, d)
R> colnames(pars) <- c("a1", "a2", "d")
R> trait_cov <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
R> mod <- generate.mirt_object(pars, itemtype = "2PL",
+    latent_covariance = trait_cov)
R> questions <- answers <- character(nitems)
R> options <- matrix("", nitems, 5)
R> spacing <- floor(d - min(d)) + 1
R> for (i in 1:nitems) {
+    if (i < 31) {
+      n1 <- sample(1:100, 1)
+      n2 <- sample(101:200, 1)
+      ans <- n1 + n2
+      questions[i] <- paste0(n1, " + ", n2, " = ?")
+    } else if (i < 61) {
+      n1 <- sample(1:50, 1)
+      n2 <- sample(51:100, 1)
+      m1 <- sample(1:10, 1)
+      m2 <- sample(1:10, 1)
+      ans <- n1 + n2 + m1 * m2
+      questions[i] <- paste0(n1, " + ", n2, " + ", m1, " * ", m2, " = ?")
+    } else if (i < 91) {
+      n1 <- sample(1:10, 1)
+      n2 <- sample(1:10, 1)
+      m1 <- sample(1:25, 1)
+      m2 <- sample(1:25, 1)
+      ans <- n1 + n2 + m1 * m2
+      questions[i] <- paste0(m1, " * ", m2, " + ", n1, " + ", n2, " = ?")
+    } else {
+      m1 <- sample(1:50, 1)
+      m2 <- sample(1:50, 1)
+      ans <- n1 + n2 + m1 * m2
```

```
+      questions[i] <- paste0(m1, " * ", m2, " = ?")
+    }
+    answers[i] <- as.character(ans)
+    ch <- ans + sample(c(-5:-1, 1:5) * spacing[i, ], 5)
+    ch[sample(1:5, 1)] <- ans
+    options[i,] <- as.character(ch)
+  }
R> df <- data.frame(Question = questions, Option = options, Answer = answers,
+    Type = "radio")
```

B. Code to generate MCAT simulation

Code used to generate and analyze a multidimensional mixed-item simulation with **mirtCAT**.

```
R> library("mirtCAT")
R> library("plyr")
R> library("mvtnorm")
R> nitems <- 360
R> N <- 1000
R> a <- matrix(c(rlnorm(nitems, 0.2, 0.3),
+    rlnorm(nitems/3, -1, 0.2), numeric(nitems),
+    rlnorm(nitems/3, -1, 0.2), numeric(nitems),
+    rlnorm(nitems/3, -1, 0.2)), nitems, 4)
R> d <- data.frame(d = rnorm(nitems/2))
R> g <- c(rep(0.2, nitems/2), rep(NA, nitems/2))
R> u <- c(rep(0.95, nitems/2), rep(NA, nitems/2))
R> ds <- data.frame(matrix(rnorm(nitems*2, 0, 2), nitems/2, 4))
R> ints <- rbind.fill(d, ds)
R> pars <- data.frame(a, g, u, ints)
R> colnames(pars) <- c(paste0("a", 1:4), "g", "u", "d", paste0("d", 1:4))
R> itemtype <- c(rep("4PL", nitems/2), rep("gpcm", nitems/2))
R> mo <- generate.mirt_object(pars, itemtype)
R> Theta <- rmvnorm(N, sigma = diag(4))
R> resp <- generate_pattern(mo, Theta)
R> SH <- rep(1, nitems)
R> SH <- ifelse(a[, 1] > 1, 0.9, SH)
R> SH <- ifelse(a[, 1] > 2, 0.6, SH)
R> SH <- ifelse(a[, 1] > 2.5, 0.3, SH)
R> content <- rep(c("4PL", "gpcm"), each = nitems/2)
R> content_prop <- c("4PL" = 0.75, "gpcm" = 0.25)
R> design <- list(min_SEM = c(.2, Inf, Inf, Inf),
+    weights = c(0.85, 0.05, 0.05, 0.05), max_items = 50,
+    exposure = SH, content = content, content_prop = content_prop)
R> library("parallel")
R> cl <- makeCluster(detectCores())
R> result <- mirtCAT(mo = mo, local_pattern = resp, start_item = "Wrule",
```

```
+    criteria = "Wrule", design = design, cl = cl)
R> est.Theta1 <- laply(result, function(x) x$thetas[1])
R> ave_nans <- mean(laply(result, function(x) length(x$items_answered)))
R> empirical_contents <- ldply(result, function(x, content)
+    table(content[x$items_answered]) / length(x$items_answered),
+    content = content)
R> empirical_props <- colMeans(empirical_contents)
R> r <- cor(Theta[,1], est.Theta1)
R> bias <- mean(Theta[,1] - est.Theta1)
R> RMSD <- sqrt(mean((Theta[,1] - est.Theta1)^2))
```

Affiliation:

R. Philip Chalmers
Department of Psychology
York University
4700 Keele St.
Toronto, ON, M3J 1P3, Canada
E-mail: rphilip.chalmers@gmail.com

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
July 2016, Volume 71, Issue 5 *Submitted:* 2014-08-15
doi:10.18637/jss.v071.i05 *Accepted:* 2015-05-25