University of California

Los Angeles

Identifying DIF for Latent Classes with the Dirichlet Process

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Miles Satori Chen

2015

Abstract of the Dissertation

Identifying DIF for Latent Classes with the Dirichlet Process

by

Miles Satori Chen

Doctor of Philosophy in Statistics

University of California, Los Angeles, 2015

Professor Peter Bentler, Chair

In Item Response Theory (IRT), Differential Item Functioning (DIF) occurs when individuals who have the same ability, but belong to different groups, have different probabilities of answering an item correctly. Traditional DIF methods require that the grouping variable be observable, like gender or ethnicity. Latent class IRT, on the other hand, allows for the fitting of IRT models where the grouping variable is unobserved.

Current latent class IRT methods (e.g. mixed Rasch models) require that the number of mixing components be defined in the estimation process. This dissertation proposes two latent class models, each with a Markov chain Monte Carlo algorithm, that can be used to fit IRT data without the need to specify the number of latent classes. The models employ a Dirichlet process or stick-breaking prior to allow an undefined number of mixing components to be fit.

Simulation results indicate that the models can correctly identify the latent classes without the need to specify how many unobserved groups there are. The power to correctly detect multiple latent classes, however, is quite low especially if the amount of DIF is small or if only a few items in a test exhibit DIF. The results of the proposed models are compared to those of the mixed Rasch model.

The dissertation of Miles Satori Chen is approved.

Frederic Paik Schoenberg

Qing Zhou

Li Cai

Peter Bentler, Committee Chair

University of California, Los Angeles

2015

To Lucy ...

may you learn to love learning.

Table of Contents

List of Figures

List of Tables

Acknowledgments

"Bless the LORD, O my soul, and all that is within me, bless his holy name! Bless the LORD, O my soul, and forget not all his benefits, who forgives all your iniquity... who satisfies you with good." – Psalm 103:1-3,5

I give thanks to God, who has blessed me beyond measure. He allowed me to meet so many wonderful people, without whom I could not have finished this endeavor.

Thank you, Peter Bentler, for being my advisor. It has been an honor to work under the direction of someone with such a distinguished career, yet who is still so accessible, and so patient in guiding me throughout. Your gentle counsel was always so helpful in getting me to the next step.

Thank you, Glenda Jones, for working tirelessly to support me and all the students in the department. You managed to always find me teaching opportunities and sources of funding, and you interceded on my behalf so many times to make sure I was still able to provide for my family through my many years as a student.

Thank you to my friends, my family members, and the entire Statistics department for rooting for me and cheering me on. Every one of your encouragements and inquiries into my progress kept me motivated to go on. A special thank you to my mother who babysat Lucy many times so I could make progress on this dissertation.

And finally, thank you, Krystal, for supporting me through the program. You sacrificed so much so I could stay in school and pursue this "little goal of mine." You demonstrate everyday what it means to love selflessly and generously. I could not have asked for a more excellent or beautiful companion. I love you.

Vita

2006	B.S., Industrial Engineering Kettering University Flint, Michigan
2009	M.S., Statistics University of California, Los Angeles Los Angeles, California
2010 - Present	Course Instructor – Statistics XL 10 and Statistics XL 13 University of California, Los Angeles Extension Los Angeles, California
2013, 2014	Instructor – Statistics 12 and Statistics 13 University of California, Los Angeles Los Angeles, California
2008 - 2013	Teaching Assistant – Multiple Classes Nominated: Teaching Assistant of the Year University of California, Los Angeles Los Angeles, California
2013	Distinguished Instructor Award University of California, Los Angeles Extension Los Angeles, California
2014	"Top Professors" Listing at UCLA Extension 4.8 out of 5 Rating (49 reviews) ratemyprofessors.com

CHAPTER 1

Introduction

1.1 Motivation

Measurement is a fundamental part of research for any field of study. Physical traits, like height and weight, can be measured in a straightforward process. It is easy to measure someone's height with a tape measure or his weight with a scale. On the other hand, latent traits, such as intelligence or attitude, are more difficult to measure. Measuring someone's intelligence, for instance, might require the use of a test with many questions. If the subject is able to answer nearly all of the questions correctly, he might be judged to have high intelligence. Likewise, in a survey of political attitudes, the survey may contain a series of statements that increasingly reflect a conservative political ideology. If the respondent agrees or endorses nearly all of the statements, he might be seen as having a high "conservatism score."

The quality of these types of evaluation, of course, depends greatly on the questions or statements used. Item Response Theory (IRT) is a framework for designing and analyzing such tests or questionnaires used to measure latent traits like intelligence or attitude. With a well-designed IRT test, an individual's responses to questions will provide information about that person's latent trait. As seen above, Item Response Theory can be used for evaluating ability-based traits like intelligence, as well as attitude-based traits like conservatism. For the sake of clarity and brevity, the remainder of this dissertation will use the terminology

consistent with measuring ability-based traits.

The IRT framework has a set of "rules" or assumptions that guide how test scores are to be understood. Perhaps the primary assumption is that an individual's response to each item can be modeled with functions. Each item has a function which relates the ability of the individual and the properties of the item to the individual's response on that item. The details of these functions will be further explained in the next chapter.

Another key assumption in the IRT framework is that individuals who share the same latent ability score should be truly comparable to one another with respect to the latent trait. The same set of test questions should be able to measure the latent abilities of many different people. It should not be necessary to have different sets of questions for different groups of people, but it should be possible to use different questions if the researcher decides to do so. Difficult items should be difficult for everyone who takes the test, and easy items should be easy for everyone who takes the test. In IRT, this property that items behave similarly for all test takers is known as item invariance.

If the property of item invariance is violated, that means a particular question behaves differently for test takers. For example, a question could be easy for some test takers, and difficult for others. If this were to happen, test takers may accuse the questions of being unfair or biased against certain groups of people, e.g. racial minorities. In IRT, an item that behaves differently for different groups of people is said to exhibit Differential Item Functioning (DIF) [HT88]. Differential Item Functioning is a statistical property of test items, and is preferred over the word "bias," which might imply an advantage or disadvantage is given to different social groups.

Test makers generally want to avoid using test items that exhibit DIF. The presence of DIF would suggest that the scores of individuals from different groups would not be comparable, violating one of the primary assumptions of IRT. Dif-

ferential item functioning may also exist because of multidimensionality, which challenges another assumption of IRT – that test questions are unidimensional, that is they only measure one latent trait.

To identify items that may exhibit DIF, several methods of detection have been created. Many of these methods detect DIF via the use of manifest grouping variables like race, gender, or some other easily identifiable trait. Responses to test items are analyzed for the different groups of people to see if DIF potentially exists. Using a manifest grouping variable in this way requires that the demographic information (or whatever variable is used to identify group membership) is collected for all test takers. Individuals with missing group data cannot be included in the DIF analysis. It is also not possible to perform a DIF analysis for any grouping variable for which data was not collected.

Even if all of the group information is available, using manifest groups may not be the best approach either. DIF detection with manifest groups is based on the "assumption that individuals within a manifest group are more similar to one another than the other manifest group" [DKS02]. This assumption could be problematic especially if the different groups are not homogeneous. For example, a manifest group labeled "Asian" may lump together Chinese, Japanese, Indian, Thai, and other ethnicities together. The test items may indeed behave differently between the Asian and non-Asian groups, but they may also behave differently within the Asian group. In such a case, the use of a manifest group label may be inadequate for proper DIF analysis. The use of a manifest grouping variable may also not provide an explanation as to why DIF is observed. While a DIF analysis may reveal that a certain test item behaves differently for one group versus another, it does not provide any information why that item performs differently.

To avoid some of the problems with using manifest groups, there have been some recent efforts to develop DIF detection methods for unobserved groups, or latent trait-classes. These methods try to identify classes based on the response

data alone so that the items perform most differentially between the classes. The latent class approach has been explored with mixture Rasch models [Sam05] and mixture IRT models [CB05]. One draw back with the existing latent class approaches is that the number of latent classes must be specified.

1.2 Purpose of the study

This dissertation presents two methods for DIF analysis from a latent class perspective using mixture IRT models with a Dirichlet process prior. The Dirichlet process prior allows for the fitting of mixture models without the need to specify the number of mixing components. This dissertation will present a blocked Gibbs sampler and a Metropolis-Hastings algorithm so that parameter estimates can be made from the posterior distribution. The dissertation will also compare the performance of this DIF identification method to the mixed Rasch model, which can be used to detect DIF in latent class IRT. It also explores the conditions which produce acceptable parameter and group recovery.

The next chapter will provide some background information on Item Response Theory, Differential Item Functioning, and existing methods for detecting DIF.

CHAPTER 2

Background

2.1 Item Response Theory

In IRT, an individual's responses to test items are modeled with functions called item response functions (IRF). The IRF relates the numeric value of an individual's latent trait (his ability) to the probability that he will correctly answer the item. Thus, an individual with low ability will have a lower probability of correctly answering the item, while an individual with high ability will have a higher probability of correctly answering the item. The shape of the item response function is generally a monotonically increasing S-shaped curve (frequently a logistic or normal ogive curve), starting with a probability of 0 for people with very low ability and reaching 1 for those with high ability.

A variety of IRT models has been developed to fit different situations. Unidimensional IRT models assume that the item response function is a function of only one latent trait (the ability), and is not influenced by any other traits or responses to other items. Multidimensional IRT models, on the other hand, allow the response function to depend on multiple latent traits. Dichotomous IRT models are used for tests where item responses are recorded in a dichotomy: correct or incorrect (unanswered questions are considered incorrect). In contrast, polytomous IRT models are used in situations where there are more than just two possible responses to items, e.g. Likert scale responses, and partial credit models. This dissertation will only discuss unidimensional dichotomous IRT models.

Unidimensional dichotomous models include the three-parameter logistic (3PL), two-parameter logistic (2PL), and one-parameter logistic (1PL) models. A commonly used variant of the 1PL model is known as the Rasch model. As their names suggest, the 3PL IRT model uses three parameters to define the item response functions while the 2PL and 1PL models use two and one parameters, respectively.

2.1.1 The Rasch Model

The Rasch model, introduced by Georg Rasch [Ras60], is an example of a unidimensional dichotomous IRT model. The model allows items to differ from one another only with respect to the difficulty of the item. The item response functions in the Rasch model are written in the following form:

$$p_{ij} = Pr(Y_{ij} = 1|\theta_i, b_j) = \frac{\exp(\theta_i - b_j)}{1 + \exp(\theta_i - b_j)} \tag{2.1}$$

Y_{ij} is the dichotomous response to an item (1 for correct, 0 for incorrect), θ_i is the latent ability of the individual ($i = 1, \ldots, n$), and b_j is the difficulty of the item ($j = 1, \ldots, p$). Thus, p_{ij} is the probability of a correct response by person i on item j. In figure 2.1, we see the IRFs of two Rasch model items. One item is easier than the other. As illustrated in the figure, IRFs of Rasch model items are parallel. Items that follow the Rasch model have only one parameter - the item difficulty.

From the item response functions, we can see that when the subject's ability θ is equal to the difficulty of the item b, the probability of correctly answering the item is 0.5. This is known as the item's inflection point. When the subject's ability is higher than the difficulty of the item, his probability of answering correctly is over 0.5, and when his ability is lower than the item's difficulty, his probability is less than 0.5.

A useful property of the Rasch model is that the total score of an individual

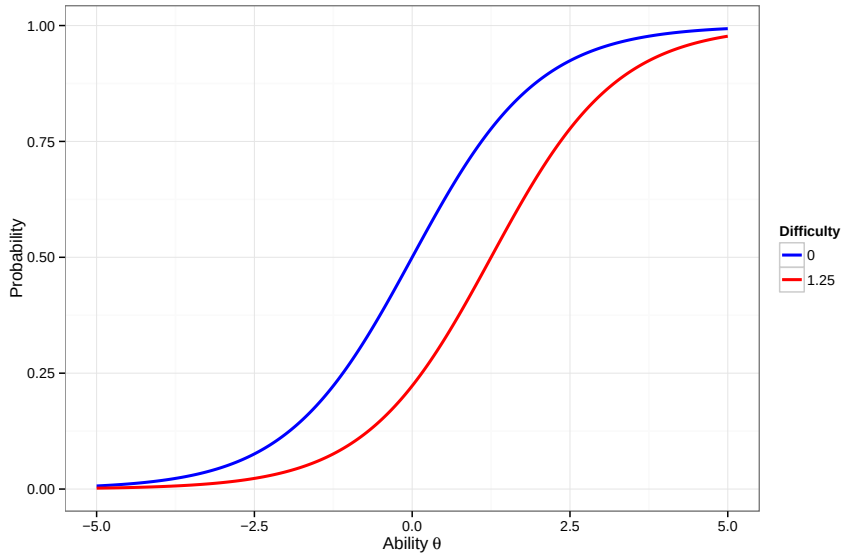

Figure 2.1: Rasch curves, with difficulty parameter b equal to 0 and 1.

is a sufficient statistic for the person's latent trait ability. The total score for an item is also a sufficient statistic for the item's difficulty parameter. In other words, the total score of a person or the total score of an item contains all of the information available to estimate that person's or item's parameter. Because of this feature, the latent trait does not need to be a part of the process to estimate item parameters, and conditional maximum likelihood estimation can be performed using the subjects' total scores instead.

The one-parameter logistic model (1PLM) is very similar to the Rasch model, except it contains a discrimination or scaling parameter which is constrained to be equal for all items. Also similar to the Rasch model is the 1-parameter normal ogive (1PNO) model. The normal ogive model uses the cumulative probability function of the normal distribution rather than a logistic curve. The logistic and normal ogive models can be made to be nearly equal if a scaling factor of 1.7 is added to the logistic model. The equation of the item response function for a normal ogive model is shown below, along with the approximately equivalent

logistic model.

$$Pr(Y_{ij} = 1|\theta_i, b_j) = \Phi(\theta_i - b_j) \approx \frac{\exp(1.7(\theta_i - b_j))}{1 + \exp(1.7(\theta_i - b_j))} \tag{2.2}$$

2.1.2 The 2-Parameter and 3-Parameter Models

In some tests, the items differ not only in difficulty but also in discriminating power. The two-parameter logistic model introduces this parameter as the item discrimination. An item with a large item discrimination will have a "steep" item response function. This means that there is a great difference in the probability of correctly answering a question between subjects whose abilities are higher than the item difficulty and that of subjects whose abilities are lower than the item difficulty. Contrast this to an item with low discrimination, where the probabilities change much less drastically between different ability levels. In the special case of having an item with a discrimination parameter of $+\infty$, the IRF becomes a step function, or a Guttman item [DL07]. In figure 2.2, a few 2-parameter model curves are shown with differing discrimination. The third parameter to appear in the 3PL

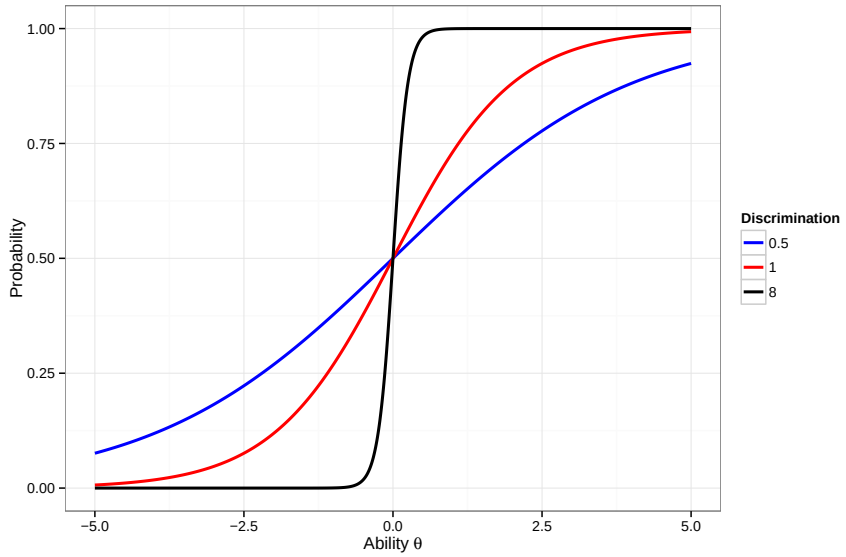

Figure 2.2: 2PL curves, with differing discrimination parameters.

model is known as the guessing parameter. This parameter frequently appears for multiple choice questions. The lower asymptote for correctly answering a question is non-zero, indicating that a subject who has very low ability may correctly answer the question simply by guessing.

The equation of the item response functions in the 3PL model can be written as follows:

$$Pr(Y_{ij} = 1|\theta_i, a_j, b_j, c_j) = c_j + (1 - c_j)\frac{\exp(a_j(\theta_i - b_j))}{1 + \exp(a_j(\theta_i - b_j))} \tag{2.3}$$

where a_j is the discrimination parameter, b_j is the difficulty parameter, c_j is the psuedo-guessing parameter of item j. Figure 2.3 shows a 3PL curve with a non-zero guessing parameter along with a 2PL curve with no guessing parameter.

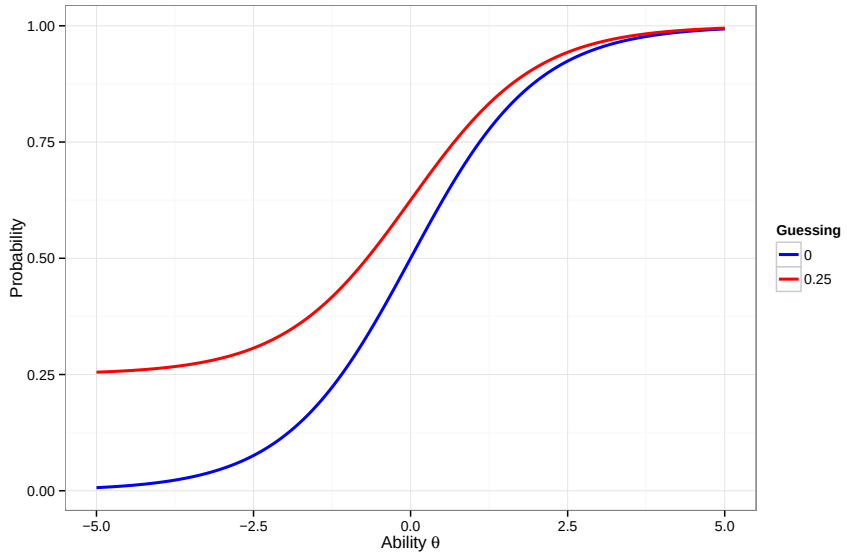

Figure 2.3: A 3PL curve with a guessing parameter of 0.25 and a 2PL curve with no guessing parameter

2.2 Maximum Likelihood Estimation for IRT Models

A large body of work has been established on methods to estimate the parameters of IRT models. The observed data that is available to the researcher is only the

responses to items. Maximum likelihood estimation (MLE) estimates parameter vales by maximizing the total likelihood of the observed responses. The likelihood of the responses of an individual, given his latent trait, θ, and the item parameters, ξ, is expressed as follows:

$$p(Y_1 = y_1, ..., Y_J = y_J|\theta, \boldsymbol{\xi}) = \prod_{j=1}^{J} Pr(Y_j = y_j|\theta, \boldsymbol{\xi})$$
$$p(\mathbf{Y}|\theta, \xi_j) = \prod_{j=1}^{J} Pr(Y_j = 1|\theta, \xi_j)^{y_j}(1 - Pr(Y_j = 1|\theta, \boldsymbol{\xi}))^{(1-y_j)} \tag{2.4}$$

where y_j is the subject's response to item j (1 for correct, and 0 for incorrect). The likelihood of the responses of all individuals is then

$$p(\mathbf{Y}|\boldsymbol{\theta}, \boldsymbol{\xi}) = \prod_{i=1}^{N}\prod_{j=1}^{J} Pr(Y_{ij} = 1|\theta_i, \xi_j)^{y_{ij}}(1 - Pr(Y_{ij} = 1|\theta_i, \xi_j))^{(1-y_{ij})} \tag{2.5}$$

where y_{ij} is the ith subject's response to item j.

2.2.1 Joint Maximum Likelihood Estimation

In Joint Maximum Likelihood (JML) estimation, both the item parameters and person parameters (latent trait abilities) are estimated jointly [DL07]. This is done in an iterative process. First, initial starting values are computed. The abilities are then estimated so that the likelihood is maximized given the current item parameter estimates. The item parameters are then estimated to maximize the likelihood given the estimated abilities. The process iterates between estimating abilities and item parameters until the changes in the likelihood is very small. For computational efficiency, the log-likelihood is used so a sum rather than a product can be maximized.

With the JML procedure, the model is not identifiable without constraints placed on the parameters [Joh07]. Mair noted that an issue with the JML procedure is that the latent traits can be considered nuisance parameters, and as the

number of subjects grows, so does the number of parameters to estimate. This could lead to estimates that are not consistent [MH07]. The joint maximum likelihood method is implemented in the function `rasch.jml()` available in the R package *sirt* [Rob15].

2.2.2 Conditional Maximum Likelihood Estimation

Conditional maximum likelihood (CML) can be implemented only for Rasch models. For the Rasch model, an individual's total score $\sum_{j=1}^{J} y_j$ is a sufficient statistic for his latent trait. The CML estimation method takes advantage of this, and conditions the likelihood on the total scores rather than the latent trait abilities [MH07]. This change eliminates the need to estimate the person parameters in order to get the item parameters. Constraints still need to be applied to some of the item parameters to ensure identifiability. The CML method is used to estimate parameters in the package *eRm* [MHM14].

2.2.3 Marginal Maximum Likelihood Estimation

In Marginal Maximum Likelihood (MML), the person parameters are removed from the likelihood by integrating them out [BA81]. The person parameters are assumed to come from a distribution $g(\theta)$, and are considered to be random effects, while the item parameters are treated as fixed effects. The ability distribution $g(\theta)$ is often selected to be the normal distribution, or for the sake of computational simplicity, a discrete distribution similar in shape to the normal distribution can be substituted. Other distributions can be selected and will have a significant impact on the estimated parameters [Joh07].

An expectation-maximization (EM) algorithm is applied to the estimation of parameters. MML is implemented in the R package *ltm* to estimate parameters of the Rasch, 2PL and 3PL models [Riz06].

2.3 Monte Carlo Estimation Methods

Monte Carlo methods are frequently used to approximate a distribution without the need to solve for it analytically. Rather than solve a possibly intractable integral, we could instead generate many random draws from the distribution. After generating enough of these random draws, we will have an idea of the properties of the desired distribution. Markov chain Monte Carlo (MCMC) methods employ a Markov chain that has a stationary distribution that matches the desired posterior distribution [PJ99]. One method to define such a Markov chain is via the Gibbs sampler.

2.3.1 The Gibbs Sampler

The Gibbs sampler draws values for each parameter from its conditional distribution assuming the other parameters are known. Even if simulating random draws from the joint posterior distribution is difficult, simulating draws for each parameter's conditional distribution may be possible. In this situation, Gibbs sampling can still simulate random draws from the posterior distribution [Alb92].

Each "loop" of the sampler will draw values for each of the parameters. Beginning with some initial values, the sampler draws a value from the conditional distribution for one parameter. Once drawn, the conditional distribution of the next parameter is updated. A value is then drawn for the next parameter, and again the conditional distribution for the following parameter is updated. This continues until random draws have been made for all of parameters, and then the loop repeats. It should be noted that the conditional distribution of each parameter takes into account the observed data, the values of the other parameters, and the prior distribution of that parameter. Also important to note is that each random draw comes from the conditional distribution based on the other parameters drawn, so the samples are not independent of one another.

The sampling continues in a loop for a selected number of cycles. Initial draws are expected to vary greatly, and to be heavily influenced by the starting values of the chain. This is considered the "burn in" period of the Markov chain. After the "burn in" period, the draws are expected to stabilize near the stationary distribution, which matches the desired posterior distribution. By studying the simulated draws generated in the stable portion of the chain, we can gain an understanding of the distribution of the parameter values.

The Gibbs sampler for IRT will draw values for the person parameters and the item parameters in the same cycle. So in a sense it creates estimates for both sets of parameters simultaneously, as is the case with JML. Patz and Junker discuss how MCMC estimates may face the same issues of JML in that the estimates of parameters may not be consistent for the true parameter values. They explain that if the person parameter estimates are "thrown away" and inference is only performed on the item parameters, then problems of inconsistency can be expected to go away [PJ99].

Patz and Junker created a Gibbs sampler algorithm in S-Plus to estimate parameters in the 2PL model [PJ99]. Albert introduced a missing variable (data augmentation) into the Gibbs sampler to estimate parameters of the 2PNO model [Alb92]. Albert's algorithm was written in MATLAB.

Gibbs sampling can also be performed using the BUGS language, which has been implemented through the freely available programs, WinBUGS, OpenBUGS [LST09], and JAGS [Plu03]. Implementation of Gibbs sampling is different with the BUGS language as it only requires that the relationships between variables in the model be defined. The BUGS system then determines how to sample the parameters on its own. This is in contrast to manual implementations of Gibbs sampling, which requires each step of the sampling algorithm to be established (often requiring that conditional distributions for each parameter be found). Curtis published a variety of IRT models written in the BUGS language so parameters

could be estimated [Cur10].

2.3.2 The Metropolis-Hastings Algorithm

The Metropolis-Hastings (M-H) Algorithm is another mechanism to generate samples from a desired distribution. The M-H algorithm can draw samples from any distribution $P(x)$, as long as one can compute the value of another function $f(x)$ that is proportional to $P(x)$.

The M-H algorithm works as follows. Given a current value of x at time t, $x^{(t)}$, we propose the next value, $x^{(t+1)}$. This value, $x^{(t+1)}$, is drawn from a proposal distribution, $g(x)$. This proposal distribution could be any distribution, but the algorithm will perform best if it is similar to the target distribution. The proposed value $x^{(t+1)}$ is accepted with a "coin flip," with a probability of acceptance of

$$\min\left(1, \frac{f(x^{(t+1)})g(x^{(t)}|x^{(t+1)})}{f(x^{(t)})g(x^{(t+1)}|x^{(t)})}\right)$$

If the proposal distribution $g(x)$ is symmetric, then $g(x^{(t+1)}|x^{(t)}) = g(x^{(t)}|x^{(t+1)})$, and the $g()$ terms in the acceptance probability cancel out. The probability simplifies to the ratio of the values of the function at the current state and the proposed state. A Gibbs step is actually a special case of M-H, where $f(x^{(t)}) = g(x^{(t+1)}|x^{(t)})$. In this situation, all of the terms cancel out, and the acceptance probability is always 1. An MCMC algorithm may incorporate Gibbs steps along with M-H steps. This type of algorithm is called, Metropolis-Hastings within Gibbs.

The choice of the proposal distribution has a significant effect on the performance of the algorithm. A common choice for a proposal distribution is the normal distribution with mean set to be $\mu = x^{(t)}$. The standard deviation, σ, selected for this proposal distribution influences how far the proposed values are from the current value. If the proposed value is too far away from the current value, the probability that it will be accepted is often small as the proposed values frequently ends up in regions of low probability. In this case, the chain will get

"stuck," and it will take a long time to generate enough samples from the target distribution. On the other hand, with a σ too small, the proposed values are very close to the current value. These values usually have a high probability of being accepted, but it also means that the generated samples are moving very slowly around state space. Here, too, it will take a long time to converge to the target distribution.

2.4 Differential Item Functioning

With IRT, inferences made about subjects' abilities based on their test scores should be equally valid across the entire population. Even if the population consists of different subgroups, the interpretation of a subject's latent ability should be the same for members of all subgroups. In a similar vein, item parameters should be invariant across groups of subjects. Estimates of the item parameters based on data from one sample of test takers should match the estimates of the item parameters based on data from another sample of test takers. Of course, sampling error will mean that parameter estimates won't be exactly the same, but the estimates should be similar.

The presence of differential item functioning violates these assumptions. Items with DIF will have different response functions for different groups of subjects. The presence of DIF means that scores of individuals from different groups should not be compared to each other.

Researchers have found that items can exhibit DIF if the items in question depend on more than one dimension [FRG09]. The test may be designed to measure what is called the primary dimension, but some of the items may actually be multidimensional. If the additional dimensions affect the probability of a correct response to the item, the additional dimensions may cause the items to exhibit DIF. These secondary dimensions may be considered to be auxiliary dimensions if

researchers intend to measuring them or they may be considered nuisance dimensions if researchers are not interested in measuring them [RS96]. This dissertation focuses on unidimensional IRT models, and will treat secondary dimensions as nuisance dimensions should any be identified during analysis.

In order to make comparative statements between groups, DIF analysis will generally label the groups. One group is called the reference group, which serves as a baseline, and the other group is called the focal group. DIF analyses may consider more than one focal group at a time. Some of the available DIF detection methods can compare multiple focal groups at once, while others can accommodate only one focal group at a time.

If DIF is identified in an item, the effect of DIF can be classified as either uniform or non-uniform. If an item has uniform DIF, then it is uniformly more difficult for all members of one group than it is for members of the other group. In the IRT framework, this almost always implies that the difficulty parameter b_j of the item is different for the two groups. Figure 2.1 can also be viewed as an item exhibiting uniform DIF between two groups. Non-uniform DIF, on the other hand, exists for items where the difference in difficulty between the groups is dependent on the value of the latent trait ability. In IRT, this usually means that the discrimination a_j parameter is different for the groups, with the possibility that the difficulty parameter is also different. Figure 2.2 can be viewed as an item exhibiting non-uniform DIF for three groups.

2.5 Existing DIF detection methods

2.5.1 Manifest group methods

The Mantel-Haenszel (MH) procedure for DIF detection, developed by Holland and Thayer [HT88], is useful for detecting uniform DIF with manifest groups. It

does not require the use of an IRT model. It involves the creation of two-by-two tables which sum up the correct and incorrect responses for the reference and focal groups. The MH procedure calculates an odds-ratio statistic based on the values from the tables that are used to decide if the item in question exhibits a significant amount of DIF. The MH procedure is frequently used to detect DIF, and it is one of the methods employed by Educational Testing Service (ETS) in their evaluation of standardized testing.

The Breslow-Day (BD) test, originally from an epidemiological application was applied to the detection of non-uniform DIF by Penfield [Pen03]. It is used to see if there is an association between item response and group membership, and to see if that association remains constant across different total test scores. If the association does not remain constant across the range of total test scores, then non-uniform DIF is likely to be present. Like the MH procedure, it does not depend on the use of IRT models. A recent study found that the Breslow-Day test performed better than the MH procedure and logistic regression with regard to committing type I error.

Logistic regression has also been used as a method for detecting both uniform and non-uniform DIF [Zum99]. A logistic regression model is fitted for the probability of correctly answering an item based on the total test score, group membership, and the interaction between the two. Uniform DIF can be detected by testing the effect of the group membership while non-uniform DIF can be detected by testing the interaction effect. Logistic regression uses total test score as a proxy for the latent trait ability and can be seen as a method that is between non-IRT dependent methods and IRT methods for detecting DIF.

Other methods try to detect DIF by fitting IRT models to the data. Fitting a 1PL or Rasch model can be used to detect uniform DIF, while 2PL and 3PL models can detect uniform and non-uniform DIF. The Likelihood ratio test (LRT) is one method that uses IRT models to detect DIF. It involves fitting two models

to the data of responses. The compact model uses the same item parameter estimates for both groups. The larger model has additional item parameters for the different groups. The significance of the additional parameters are tested using a likelihood ratio test. Cohen, Kim, and Wollack [CKW96] used a likelihood ratio test in Multilog to detect DIF in two-parameter and three-parameter IRT models. They found that type I error rates were similar to alpha levels for two-parameter models, but not for three-parameter models.

Another method for detecting DIF using IRT models is the Lord's chi-square test [KC95]. It tests a null hypothesis that item parameters for one group of subjects is equal to the item parameters for another group of subjects. A test statistic based on the estimates of the item parameters and variance-covariance matrices has an asymptotic chi-squared distribution, which can be used to determine if the DIF is significant.

One more method used to detect DIF is Raju's area method [KC95]. It takes fitted IRT models and calculates the area between the estimated response functions for the two groups. A statistic is calculated based on the null hypothesis that the area between the two curves is zero.

2.6 DIF analysis with mixture IRT models

Mixture models allow us to model data that may be produced by different groups or classes, where the group membership is unknown. An IRT mixture model can be used in the context of investigating DIF. If a mixture model fits the response data, it may indicate the presence of DIF. The different mixing components may be considered to be subgroups of the population, and the subgroups may have different item parameters.

Generically, the overall likelihood $f(\cdot)$ of a mixture model is the weighted sum of the likelihoods of each of the components $f_k(\cdot)$ using component weight π_k,

$(k = 1, \ldots, K)$,

$$f(y_i) = \sum_{k=1}^{K} \pi_k f_k(y_i)$$

where $0 < \pi_k < 1$ and $\sum_k \pi_k = 1$. For IRT mixture models, $f(y_i)$ is the overall item response function, and $f_k(y_i)$ would be the item response function for members of subgroup k. If the $f_k(y_i)$ are different, that would suggest the presence of DIF.

2.6.1 Mixture Rasch Model

Rost [Ros90] extended the Rasch model into a Mixture Rasch Model to allow for the possibility of latent classes. These latent classes are subgroups of the population that might differ in response strategies to the items. The latent class could be related to a manifest grouping variable, but it is also possible that different classes exist solely based on the response patterns exhibited. Certain items may have different difficulties for members of different latent classes, and could ultimately result in a different ordering of item difficulties. Members within a latent class will have responses that reflect the same ordering of item difficulties, while members of other classes will have responses that reflect a different ordering of item difficulties [Emb07].

In the Mixture Rasch Model, each latent class will have its own difficulty parameters for each of the items. The probability of a correct response for an individual is a mixture of the item response functions for each of the latent classes and the probability that the individual is in the latent class. Let p_{ijk} be the probability of a correct response by person i on item j and belonging to class k. The probability of a correct response by person i on item j would then be the

weighted sum of a correct response

$$\begin{aligned} p_{ij} &= \sum_{k=1}^{K} \pi_k p_{ijk} \\ &= \sum_{k=1}^{K} \pi_k \frac{\exp(\theta_{ik} - b_{jk})}{1 + \exp(\theta_{ik} - b_{jk})} \end{aligned}$$

2.6.2 Estimating parameters of the Mixed Rasch Model

Estimating parameters in a mixed Rasch model can be done with an EM-type algorithm, suggested by Rost [Ros90]. The group membership is treated as unknown, and optimization is done on the complete data log likelihood, including group membership. During the Expectation step, the expected frequencies of the different response patterns are estimated for each latent class. The frequency estimates are found by weighting the observed frequency with a proportion indicating the probability of observing this pattern within a class. These frequency counts are not necessarily integers as they are expected frequencies. The weighting proportion depends only on the model parameters and latent class proportion. Two patterns, the vector of 0s and vector of 1s, are excluded from the estimation procedure as they do not reveal information about the latent classes.

In the Maximization step of the algorithm, the maximum likelihood estimates of the parameters are recalculated based on the expected pattern frequencies found in the E-step. The score parameters and item parameters are estimated for each of the latent classes. They are found by maximizing the log-likelihood of the data for each of the classes. Through an iterative process, estimates of the item and person parameters can be found. Preinerstorfer [PF12] uses this EM algorithm suggested by Rost in his R package `mRm` to estimate the parameters in a Mixed Rasch Model. The EM algorithm for mixed Rasch models was also adapted by Frick, Strobl et.al. [FSL12] in the R package `psychomix`.

In the EM algorithm, the number of latent classes is not estimated. The

number of latent classes must be set prior to the estimation of parameters. If the number of latent classes is unknown, several models can be fitted to the data, each with a different number of classes. A model will then need to be selected based on different criteria, such as total log likelihood, Akaike information criterion (AIC), or Bayesian information criterion (BIC). Both packages, mRm and psychomix, provide the information criterion of the fitted model to aid model selection.

Parameters of the Mixed Rasch Model can also be estimated via MCMC sampling.

2.6.3 Mixture 2- and 3-parameter logistic model

Additional parameters can be added to the mixture Rasch model to produce the mixture 2- and 3-parameter logistic models (M2PLM and M3PLM). In a M3PLM, the items use the 3-parameter logistic model, which uses the difficulty, discrimination, and guessing parameters and allows the parameters to differ for each of the latent classes. Like the mixture Rasch model, each latent class will have a set of parameters for each of the items. Similarly, the probability that an individual correctly responds to an item is the resulting sum product of the item response function and the probability of latent class membership.

$$f(y_i) = \sum_{k=1}^{K} \pi_k f_k(y_i|\theta_i, a_{kj}, b_{kj}, c_{kj}) \tag{2.6}$$

Estimating parameters of finite mixture models generally require that the number of mixing components be specified. For latent classes, the number of mixing components may be unknown. Using a Dirichlet process mixture model can avoid this problem as they do not require the number of mixing components to be specified.

2.7 The Dirichlet Process

A Dirichlet process [Fer73] can be described as a distribution over probability distributions. In other words, the Dirichlet process is a distribution, and a random draw from it will produce another probability distribution [BJ06].

The Dirichlet process is defined by H_0, a base distribution, and α, the concentration parameter. A draw from the Dirichlet process will produce a discrete probability distribution. This discrete output distribution will have several values, each of which can be drawn from the base distribution H_0. Even if the base distribution H_0 is continuous, the draws from the Dirichlet process will be a discrete distribution, meaning the values drawn from this output distribution will repeat. The concentration parameter α determines the extent to which the values repeat, with lower values indicating more repetition and a lower variety of possible values. Thus, as α approaches zero, the Dirichlet process will produce a distribution with a single possible value, and as α approaches infinity, the Dirichlet process becomes nearly continuous.

The generation of a Dirichlet process can be explained using a stick breaking process. The stick breaking process requires us to imagine a stick of length one. A proportion, V_1 is drawn from the distribution $Beta(1, \alpha)$. A piece of the stick is broken off with size equal to $\pi_1 = V_1$, and placed at a value θ_1 which is drawn from the base distribution H_0. This leaves $(1 - V_1)$ of the stick behind. Another proportion, V_2 is also drawn from the distribution $Beta(1, \alpha)$, and is broken off from the remaining portion of the stick. This piece, which is of size $\pi_2 = V_2(1-V_1)$ is placed at value θ_2, also drawn from the base distribution H_0. This process continues. Each time, a piece is broken off from the remaining portion of the stick, and placed at a value drawn from the base distribution. The size of each piece indicates the probability of a value i in the discrete distribution that results from the Dirichlet process.

The Dirichlet Process creates a discrete distribution, with potentially an infinite number of components. As such, it is suitable for use in mixture models with an unspecified number of mixing components [Nea00].

CHAPTER 3

The Proposed Models and Samplers

3.1 Two Mixture Models

In this chapter, I present two IRT mixture models and samplers that can be used to recover their parameters. One model and sampler is based on the semi-parametric Dirichlet process mixture model proposed by Miyazaki and Hoshino [MH09]. The other is based on the MCMC sampler for a 2PL IRT model by Patz, Junker, and VanHoudnos [JV] [Van].

3.2 Mixture model with a Dirichlet Process prior

Let θ_i represent the latent trait ability of individual i where $i \in 1, \ldots, N$. We will assume that the distribution of θ_i is normal, with mean $\mu = 0$ and standard deviation $\sigma = 1$. That is

$$\theta_i \sim N(0, 1)$$

Thus, we have N individuals in our data, and each individual i belongs to latent class k_i, where $k_i \in 1, \ldots, K < N$. Each latent class has mixing proportion π_k where $\sum_{k=1}^{K} \pi_k = 1$. The mixing proportion of each latent class is generated by a finite-dimensional Dirichlet Process. Sethuraman's stick breaking process [Set94] has been shown to be equivalent to the Dirichlet Process and will be the method used to find the mixing proportions. Following Ishwaran and James in their 2001

paper [IJ01], we have

$$\begin{aligned}
\pi_1 &= V_1 \\
\pi_2 &= (1 - V_1)V_2 \\
\pi_k &= (1 - V_1)(1 - V_2)\dots(1 - V_{k-1})V_k,\ 3 \leq k < K
\end{aligned} \tag{3.1}$$

We set $V_K = 1$, to ensure that $\sum_{k=1}^{K} \pi_k = 1$. V_k are independently drawn from Beta(a_k, b_k), where $a_k, b_k > 0$. This is the stick-breaking procedure, where at each step we break off a portion of the remaining stick V_k and use the length of this piece as π_k.

We will use the two-parameter logistic (2PLM) IRT model as our base model, which has item response functions that are very similar to the two-parameter normal ogive model.

$$\begin{aligned}
Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}) &= \frac{\exp(1.7a_{k_ij}(\theta_i - b_{k_ij}))}{1 + \exp(1.7a_{k_ij}(\theta_i - b_{k_ij}))} \\
&\approx \Phi(a_{k_ij}(\theta_i - b_{k_ij}))
\end{aligned} \tag{3.2}$$

Here, Y_{ij} is the dichotomous response to item j (where $j \in 1, \dots, J$) by subject i and θ_i is the latent trait ability of subject i. a_{k_ij} and b_{k_ij} are the discrimination and difficulty parameters of item j for latent class k_i, respectively.

For the Gibbs sampler, it is convenient to re-parameterize the model as follows. Let $\alpha_{k_ij} \equiv a_{k_ij}$ and $\beta_{k_ij} \equiv a_{k_ij}b_{k_ij}$. The item response model can then be rewritten as

$$Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}) = \frac{\exp(1.7(\alpha_{k_ij}\theta_i - \beta_{k_ij}))}{1 + \exp(1.7(\alpha_{k_ij}\theta_i - \beta_{k_ij}))} \tag{3.3}$$

Albert [Alb92] suggested introducing the latent variable Z_{ij} with realization z_{ij}. Z_{ij} is normally distributed with mean $\eta_{ik_ij} = \alpha_{k_ij}\theta_i - \beta_{k_ij}$ and standard

deviation equal to 1.

$$z_{ij} \sim N(\eta_{ik_ij}, 1), \text{ where } \eta_{ik_ij} = \alpha_{k_ij}\theta_i - \beta_{k_ij}$$

Which is equivalent to

$$z_{ij} = \alpha_{k_ij}\theta_i - \beta_{k_ij} + \epsilon_i, \text{ where } \epsilon_i \sim N(0, 1)$$

Suppose $X_{ij} = 1$ when $z_{ij} > 0$ and $X_{ij} = 0$ when $z_{ij} \leq 0$. It can then be shown that X_{ij} are independent Bernoulli random variables with probability $\Phi(\eta_{ik_ij})$. The 2PLM model that is used approximates the normal ogive model very closely, so we can say that the responses Y_{ij} can also be treated as independent Bernoulli random variables with probability $\Phi(\eta_{ik_ij})$.

This is useful, because when we want to sample from the distribution of Z, the probability of Z_{ij} given $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}$, and $\mathbf{Y}$ can be expressed in terms of the normal distribution, as follows

$$Z_{ij}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{Y} \sim \begin{cases} N(\eta_{ij}, 1) & \text{truncated at the left by 0 if } y_{ij} = 1 \\ N(\eta_{ij}, 1) & \text{truncated at the right by 0 if } y_{ij} = 0 \end{cases} \tag{3.4}$$

To account for the mixture components, we want to be able to sample from the joint posterior distribution of α, β, θ, z, and k. The complete posterior can be written as:

$$\begin{aligned} Pr(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{z}, \mathbf{k}|\mathbf{Y}) &= Pr(\mathbf{z}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{k}, \mathbf{Y}) \times Pr(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{k}|\mathbf{Y}) \\ &= Pr(\mathbf{z}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{k}, \mathbf{Y}) \times Pr(\boldsymbol{\alpha}, \boldsymbol{\beta}|\mathbf{k}) \times Pr(\boldsymbol{\theta}) \times Pr(\mathbf{k}) \\ &= \prod_{i=1}^{N} \Bigg\{ \phi(\theta_i) \times Pr(k_i) \times \prod_{j=1}^{J} Pr(z_{ij}|\alpha_{k_ij}, \beta_{k_ij}, \theta_i, Y_{ij}) \times \\ &\qquad Pr(\alpha_{k_ij}, \beta_{k_ij}|k_i) \Bigg\} \end{aligned} \tag{3.5}$$

The discrimination and difficulty parameters of the different mixture components are generated with a finite-dimensional Dirichlet Process. The base distribution of the discrimination parameter is the multivariate normal distribution centered at the hyperparameter $\boldsymbol{\alpha}_0$ with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\alpha}}$, truncated on the left at 0. The base distribution of the difficulty parameter is the multivariate normal distribution centered at the hyperparameter $\boldsymbol{\beta_0}$ with covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$.

Dirichlet process mixture models are often treated as being potentially infinite dimensional. Ishwaran and Zarepour [IZ02] proposed a finite-dimensional Dirichlet process mixture model by limiting the number of mixture components. Ishwaran and James [IJ01] demonstrated that a finite-dimensional Dirichlet process mixture model will accurately approximate an infinite dimensional Dirichlet process mixture model, given that the number of mixing components is large enough.

3.2.1 Description of the sampler for this model

The blocked Gibbs sampler can be used for drawing values from the posterior distribution of this model. This blocked Gibbs sampler is based heavily on the work of Miyasaki and Hoshino [MH09], who in turn based their model on Ishwaran and James's work [IJ01], and Albert's work [Alb92].

1. Sample α and β

   The first step of the blocked Gibbs sampler is to draw parameter values for α and β. In our model, the latent variable z_{ij} is equal to $\alpha_{k_ij}\theta_i - \beta_{k_ij} + \epsilon_i$. We must draw values of the parameters for each latent class. For each class l, we have $z_j = \alpha_j\theta_i - \beta_j + \epsilon_i$. In matrix notation, $\mathbf{Z} = \boldsymbol{\alpha\theta} - \boldsymbol{\beta} + \boldsymbol{\epsilon}$. As suggested by Albert, we can set matrix $\boldsymbol{\xi} = [\boldsymbol{\alpha}, \boldsymbol{\beta}]$ and $\mathbf{X} = [\boldsymbol{\theta}, -\mathbf{1}]$. We now have

$$\mathbf{Z} = \mathbf{X}\boldsymbol{\xi} + \boldsymbol{\epsilon} \tag{3.6}$$

where $\boldsymbol{\epsilon}$ comes from $N(0, 1)$. The ordinary least squares estimator for $\boldsymbol{\xi}$ is

$$\hat{\boldsymbol{\xi}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Z} \tag{3.7}$$

The conditional distribution of $\boldsymbol{\xi}$ given $\mathbf{Z}$ and $\boldsymbol{\theta}$, which will be used to generate the draws, is then

$$N(\hat{\boldsymbol{\xi}}, (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1})I(\alpha > 0) \tag{3.8}$$

where $I(\alpha > 0)$ indicates that only positive draws of α will be accepted. In this step of the Gibbs sampler, values of α and β must be drawn for each of the latent classes where at least one subject has been assigned.

2. Sample latent class assignments k_i

   With the draws of the item parameters α and β now available, the sampler will produce draws of the latent class assignments from its conditional distribution. Subject i belongs to latent class k_i, which can take on values 1 through K. The latent class assignments are contained in the vector $\mathbf{k}$. The distribution of $\mathbf{k}$ comes from a discrete probability measure. The conditional distribution of each k_i is then the following discrete distribution

   $$\sum_{k=1}^{K} \pi_{k_i}\delta_k(\cdot) \tag{3.9}$$

   where δ_k is a discrete measure concentrated at k, and π_{k_i} is the corresponding probability for each latent class for individual i. A Metropolis-Hastings step is used for the acceptance probability of the sampled k.

3. Sample latent class proportions π with a stick-breaking process

The probabilities π_k are drawn from the generalized Dirichlet distribution.

$$\begin{aligned}
\pi_1 &= V_1 \\
\pi_k &= (1 - V_1)(1 - V_2) \dots (1 - V_{k-1})V_k,\ 2 \leq k < K \\
V_k &\sim Beta\left(a_k + M_k, b_k + \sum_{m=k+1}^{K} M_m\right)
\end{aligned} \tag{3.10}$$

In our case we have a finite dimensional Dirichlet process. According to Ishwaran and James, this corresponds to having $a_k = \alpha/N$ for all k and $\mathbf{b} = (\sum_{k=2}^{N} a_k, \sum_{k=3}^{N} a_k, \cdots, a_N)$. To clarify, the α in the a_k term is the Dirichlet process concentration parameter, and not the discrimination parameter in the IRT model. M_k is equal to the number of subjects assigned to latent class k.

4. Sample latent values z

   The next step in the Gibbs sampler is to generate draws of z_{ij}. The conditional distribution of z_{ij} based on the other parameters, as explained earlier, is

$$p(Z_{ij}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{Y}) \sim \begin{cases} N(\eta_{ij}, 1) & \text{truncated at the left by 0 if } y_{ij} = 1 \\ N(\eta_{ij}, 1) & \text{truncated at the right by 0 if } y_{ij} = 0 \end{cases}$$

5. Sample latent abilities θ

   The last step of the Gibbs sampler is to draw values of θ_i. The conditional distribution of θ_i is

$$N\left(\frac{\sum_{j=1}^{J} \alpha_{k_i j}(z_{ij} + \beta_{k_i j})}{\sum_{j=1}^{J} \alpha_{k_i j}^2 + 1}, \frac{1}{\sqrt{\sum_{j=1}^{J} \alpha_{k_i j}^2 + 1}}\right) \tag{3.11}$$

   Once all parameters have been updated with their random draws, the sampler runs through the loop again.

3.3 Mixture model with a stick-breaking prior

Another mixture IRT model is presented here. This model with a stick-breaking prior is similar to the mixture model presented earlier. A stick-breaking process is used to generate the latent class probabilities. In the other model and sampling algorithm, the item parameters came from a conditional distribution based on the least-squares estimates of the parameters based on the responses and latent class assignments. The item parameters in this model are sampled from their respective prior distributions, and accepted with a Metropolis-Hastings probability.

We specify the model as follows. The latent trait ability of each individual is θ_i. $\theta_i \sim N(0,1)$. Each individual i belongs to latent class k_i. Each latent class has mixing proportion π_k where $\sum_{k=1}^{K} \pi_k = 1$. Like the other model, π_k is generated by a finite-dimensional stick-breaking process.

$$\begin{aligned}
\pi_1 &= V_1 \\
\pi_2 &= (1 - V_1)V_2 \\
\pi_k &= (1 - V_1)(1 - V_2)\dots(1 - V_{k-1})V_k,\ 3 \leq k < K
\end{aligned} \tag{3.12}$$

We set $V_K = 1$, and we have $\sum_{k=1}^{K} \pi_k = 1$. V_k are independently drawn from Beta(a_k, b_k), where $a_k, b_k > 0$.

We will use the two-parameter logistic (2PLM) IRT model as our base model.

$$Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}) = \frac{\exp(a_{k_ij}(\theta_i - b_{k_ij}))}{1 + \exp(a_{k_ij}(\theta_i - b_{k_ij}))} \tag{3.13}$$

Again, Y_{ij} is the dichotomous response to item j by subject i and θ_i is the latent trait ability of subject i. a_{k_ij} and b_{k_ij} are the discrimination and difficulty parameters of item j for latent class k_i, respectively. The difficulty parameters b have a normal prior. The discrimination parameters have a prior that is a log-normal distribution.

We want to be able to sample from the joint posterior distribution of a, b, θ,

and k. The complete posterior can be written as:

$$\begin{aligned} Pr(\mathbf{a}, \mathbf{b}, \boldsymbol{\theta}, \mathbf{k}|\mathbf{Y}) &= \frac{Pr(\mathbf{Y}|\mathbf{a}, \mathbf{b}, \boldsymbol{\theta}, \mathbf{k})}{Pr(\mathbf{a}, \mathbf{b}, \boldsymbol{\theta}, \mathbf{k})} \\ &\propto Pr(\mathbf{Y}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{k}) \\ &\propto \prod_{i=1}^{N} \prod_{j=1}^{J} Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij})^{y_{ij}} \times \\ &\quad (1 - Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}))^{(1-y_{ij})} Pr(a_{k_ij}, b_{k_ij}|k_i) Pr(\theta_i) \end{aligned} \tag{3.14}$$

The Metropolis-Hastings algorithm will be used to draw samples from the posterior distribution, so it is not an issue that we only know the distribution up to a normalizing constant.

3.3.1 Description of the sampler

For this model, we will use a Metropolis-Hastings algorithm. Every iteration of the sampling loop will draw values for each of the parameters in the model.

The sampling steps are described here.

1. Sample the latent trait values θ

   The posterior distribution of the latent ability parameters come from the following distribution. The $\ldots$ in $Pr(\theta_i|\ldots)$ represent all of the other parameters in the model.

   $$\begin{aligned} Pr(\theta_i|\ldots) \propto \prod_{j=1}^{J} & Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij})^{y_{ij}} \\ & \times (1 - Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}))^{(1-y_{ij})} N(\theta_i; \mu = 0, \sigma = 1) \end{aligned}$$

   The M-H algorithm is used. A normal distribution can be used as the proposal distribution. The proposed value, $\theta^\star$ is drawn from a normal distribution centered at the current value of $\theta \sim N(\theta, \sigma_{MH})$, where σ_{MH} is the

standard deviation of the proposal distribution. The value of σ_{MH} can be tuned so that the acceptance rates are desirable. The proposal distribution is symmetric, i.e. $N(\theta; \theta^\star, \sigma_{MH}) = N(\theta^\star; \theta, \sigma_{MH})$. These terms cancel out in the acceptance probability.

The proposed value $\theta^\star$ is accepted with the following probability:

$$Pr(accept_i) = \min\left\{1, \frac{Pr(\theta_i^\star|\ldots)}{Pr(\theta_i|\ldots)}\right\}$$

A proposed value $\theta^\star$, and the acceptance probability must be calculated for all N subjects in the data.

2. Sample the discrimination parameter a

   The posterior distribution of the discrimination parameters come from the following distribution.

$$Pr(a_j|\ldots) \propto \prod_{j=1}^{J} Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij})^{y_{ij}} \times (1 - Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}))^{(1-y_{ij})} LogN(a_j; \mu_a, \sigma_a)$$

   $\mu_a and \sigma_a$ are the hyperparameters of the prior distribution. For the M-H algorithm, we will use a normal distribution centered at the current value of a as our proposal distribution.

3. Sample the difficulty parameter b

   The difficulty parameters can be sampled from the following distribution.

$$Pr(b_j|\ldots) \propto \prod_{j=1}^{J} Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij})^{y_{ij}} \times (1 - Pr(Y_{ij} = 1|\theta_i, a_{k_ij}, b_{k_ij}))^{(1-y_{ij})} N(b_j; \mu = 0, \sigma = 1)$$

   For the M-H algorithm, we will use a normal distribution centered at the current value of b as our proposal distribution.

4. Sample the latent class assignments k

   The latent class assignments are sampled from the following distribution.

$$\begin{aligned} Pr(k_i|\ldots) \propto \prod_{j=1}^{J} & Pr(Y_{ij}=1|\theta_i, a_{k_ij}, b_{k_ij})^{y_{ij}} \\ & \times (1 - Pr(Y_{ij}=1|\theta_i, a_{k_ij}, b_{k_ij}))^{(1-y_{ij})} \\ & \times \pi_{k_i} \end{aligned}$$

   For the M-H algorithm, we will use a generate k_i as a single draw from the multinomial distribution with probabilities $\boldsymbol{\pi}$.

5. Sample the latent class proportions π

   The last parameter to be sampled is the latent class proportions. The latent class proportions π are generated by the stick breaking process.

$$\begin{aligned} \pi_1 &= V_1 \\ \pi_2 &= (1 - V_1)V_2 \\ \pi_k &= (1 - V_1)(1 - V_2)\ldots(1 - V_{k-1})V_k,\ 3 \leq k < K \end{aligned} \tag{3.15}$$

   The proposed values of v are drawn from a beta distribution. For v_k, the beta distribution's first shape parameter is proportional to the count of subjects assigned in the latent class k. The second shape parameter in the beta distribution is proportional to the count of subjects in the remaining latent classes $k+1$ to K. The sampled v values are used to construct the values of pi.

   The proposed $\pi^\star$ is accepted with the MH acceptance probability. The posterior probability used to calculate the acceptance is as follows:

$$Pr(\pi^\star|\ldots) \propto \prod_{k=1}^{K}\prod_{i=1}^{N} \pi_{k_i}^{I(k_i=k)} \times Dir(\pi^\star; \alpha = \boldsymbol{\pi}) \tag{3.16}$$

This completes the steps for the M-H sampling algorithm.

CHAPTER 4

Simulations

In this chapter, we run a series of simulations and discuss the results. We generate data with different parameters and use the proposed algorithms to recover the parameters of the models. We will refer to the proposed Gibbs sampling algorithm with Dirichlet process prior based on the work of Hoshino and Miyasaki as the Gibbs sampler. We will refer to the Metropolis-Hastings algorithm with a stick breaking prior, based on the work of Patz, Junker, and VanHoudnos as the MH algorithm.

4.1 A first simulation

In our first simulation, the responses were generated by a mixture Rasch model (MRM). The first set simulated the responses of $N = 2000$ subjects on $J = 30$ items. Two latent classes were simulated, with approximate proportions of 78% for one latent class, and 22% for the other. No manifest grouping variable was specified.

The difficulty parameters of the two latent classes were made to be additive inverses of each other. For the first latent class, the difficulty parameters started at -3 and progress to +3 across the 30 items. The second latent class was -1 times the parameters of the first latent class. Being a Rasch model, the discrimination parameters were set to be equal for all items. The values of some of the parameters used in the simulation are shown in table 4.1.

Both of the proposed algorithms, the Gibbs sampler and MH algorithm, were run without specifying the number of latent classes or manifest grouping variable. Although the number of latent classes are not specified, both sampling algorithms require that we set a maximum number of latent classes. For the algorithms, we specify that a maximum of 8 latent classes are to be estimated. The algorithms can accommodate more latent classes, but for computational efficiency 8 was arbitrarily chosen.

4.1.1 Results of the Gibbs Sampler

The Markov chain from the Gibbs sampler seemed to stabilize after 200 samples. The chain ran for an additional 7500 cycles with a thinning factor of 3, keeping 2500 samples. The chain displayed a significant amount of posterior variance, but remained stable around the true parameter values.

The Gibbs sampler correctly identified two latent classes. The estimated proportions (based on the mean of the samples) of the latent classes were 0.768, and 0.222, with the remaining empty latent classes having a non-zero but negligible probability associated with them. The traceplot of the proportion of latent class 1 can be seen in Figure 4.1.

The estimates for the difficulty parameters matched closely with the true values. One exception was the easiest item for the second latent class. The true difficulty for this item was set to be -3, but the algorithm estimated the item to have a difficulty of -7.6. Nearly all of the respondent in the second latent class answered this item correctly, and thus there was little information available to estimate the difficulty of this item. Representative traceplots of one of the items (item 14) for both latent classes can be seen if Figure 4.2.

Correlation between the true difficulty values and the estimated values (using the mean of the samples) is 0.957. If the erroneously estimated item is removed,

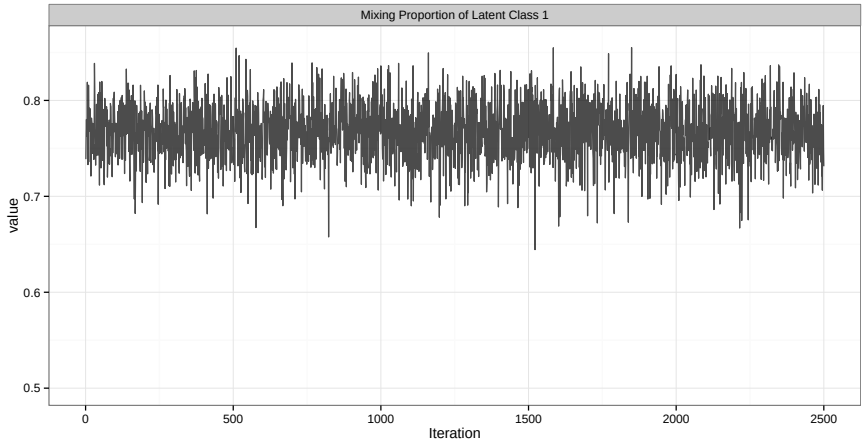

Figure 4.1: Traceplot of the MCMC chain of the Gibbs sampler for the proportion of latent class 1.

the correlation is 0.996. Estimated values of some of the difficulty parameters can be seen in the middle columns of Table 4.1.

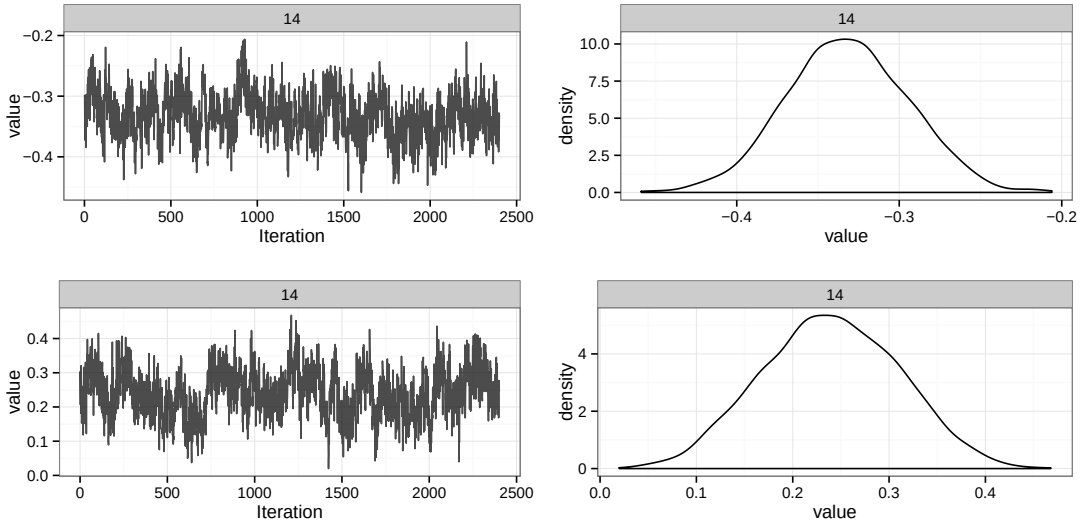

Figure 4.2: Traceplots of the MCMC chains of the Gibbs sampler and corresponding densities for the difficulty parameters of item 14 in both latent classes.

4.1.2 Results of the MH Algorithm

The Metropolis-Hastings algorithm with a stick-breaking prior was also used to estimate the parameters of the model. The Markov chain of the MH algorithm converged fairly quickly. The chain was allowed to run for another 2500 iterations to generate samples.

The algorithm correctly identified two latent classes. The mean value of the samples for the proportion parameters were .778 and .216. Again, the two proportions do not add up to 1 because the other latent classes, despite having no subjects assigned to them, have non-zero probabilities. The traceplot for the proportion of one of the latent classes can be seen in Figure 4.3.

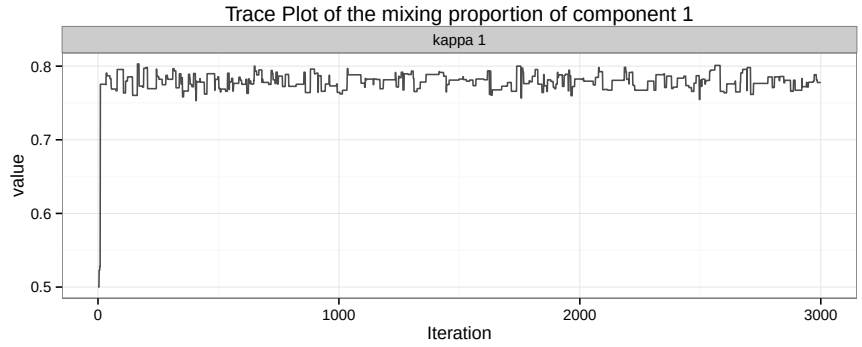

Figure 4.3: Traceplot of the MCMC chain for the mixing proportion of latent class 1. The Burn-in period (first 500 draws) has been left in the trace plot.

The estimates of difficulty parameters were aligned with the true values of the parameter. Correlation between the true values and the estimated values (mean of samples) is 0.999. A traceplot for the difficulty parameter of one of the items can be seen in Figure 4.4. Estimated values of some of the difficulty parameters can be seen in the right-most columns of Table 4.1.

The proposed algorithm also estimates person parameters. A plot of the estimated person parameters against the true values can be seen in Figure 4.5. The estimated parameters aligned closely to the true values.

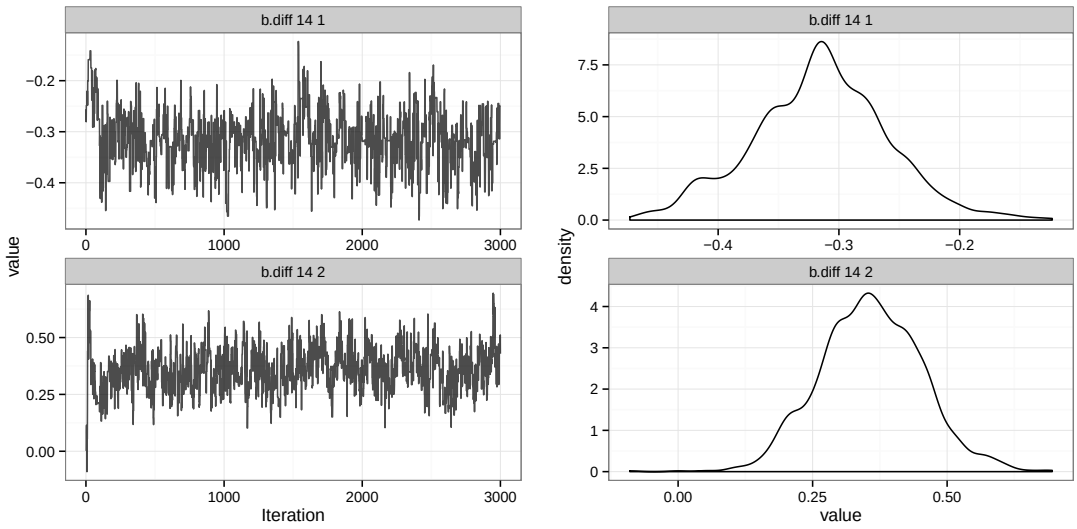

Figure 4.4: Traceplot of the MCMC chain of the MH algorithm and corresponding density for the difficulty parameters of item 14 in both of the latent classes.

4.1.3 Comparison to MRM

To compare the performance of this algorithm with existing methods, I used the `mrm()` function from package `mRm`. I fit a mixed Rasch model to the data, specifying two latent classes. The `mRm` algorithm correctly identified the proportions of latent classes to be very close to the true proportions. The estimates of the parameters were also accurate, with a correlation of 0.999 to the true parameter estimates.

I also used `mRm` to fit mixed Rasch models with the incorrect number of latent classes specified. The function converged to solutions despite having searching for the wrong number of latent classes. The BIC for these models were higher (1-class model: 77845, 3-class model: 49752, 4-class model: 49702) than the BIC for the model with 2 latent classes (BIC: 49486). This suggests that if one were to use `mRm` to fit a mixed Rasch model, he would conclude 2 is the correct number of latent classes.

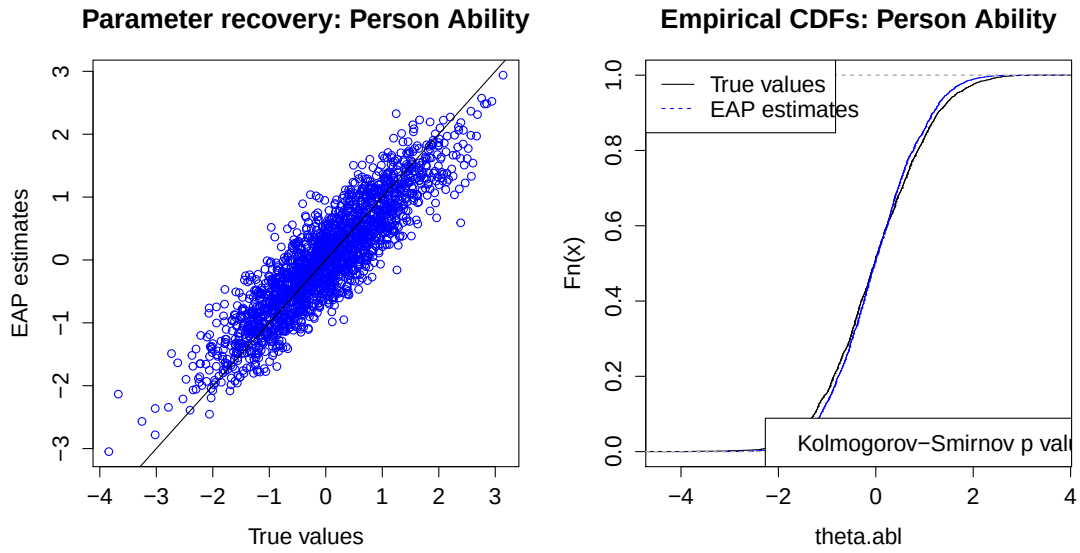

Figure 4.5: Plot comparing the true value of latent ability parameters θ and the estimated values produced by the sampler.

4.2 Additional simulations of the Mixed Rasch Model

The population was changed to have different proportions of latent classes. Additional simulations were run with latent class proportions of a 50/50 split and 60/40. Other settings were left unchanged. In these simulations, both the Gibbs sampler and the MH algorithm were correctly able to identify two latent components with the proportion estimates within one percentage point of the true value. Parameter estimates were also consistent with the true parameter values.

The `mrm()` function also successfully identified the latent class proportions and parameter estimates when 2 latent classes were specified. Additional mixed Rasch models with an improper number of latent classes were also fitted, but returned BIC values that were higher than the model with only 2 latent classes, as expected.

The population was further changed to have three latent classes, with proportions 0.5, 0.3, and 0.2. The difficulty parameters of the first two classes remained additive inverses, while the third class alternated negative and positive difficulty

values. Both the Gibbs sampler and the MH algorithm were able to correctly detect three latent classes with the estimated proportions within two percentage point of the true proportion values. Estimates of the parameters were aligned with the true values, with the exception of the items with the lowest difficulties, which were estimated to have even lower difficulty levels than the actual parameter values. The values of some of the difficulty parameters for the three latent classes and their estimates are shown in Table 4.2.

For the 3-class model, `mRm` was also able to find the correct proportions. Fitting the wrong number of latent classes with `mRm` returned BIC values that were higher than that of the 3-class model, suggesting that the 3-class model is the best fitting one.

4.3 Simulations of mixture 2PL models

In further simulations, the discrimination parameters were allowed to vary, thus generating data from a mixture 2PL model.

Multiple simulations were run, with two and three latent classes. For simulations with two latent classes, difficulty parameters were left as additive inverses. For simulations with three latent classes, the first two classes had difficulty parameters that were additive inverses, and the third class had difficulties alternating negative and positive. Discrimination parameters were randomly sampled from a uniform distribution from 0.5 to 2.5.

For these simulations, both the Gibbs sampler and the MH algorithm were able to consistently recover the correct number of latent classes, and were also able to estimate the proportions of the latent classes accurately to within 2 percentage points of the true latent class proportions.

For both the Gibbs sampler and the MH algorithm, the correlation between the true and estimated values of the difficulty parameters was found to be above

0.97 in the simulated cases. Correlation between the true and estimated values of the discrimination parameters averaged around 0.80. The true and estimated values of some of the discrimination parameters for one of these simulations are shown in table 4.3. Scatter plots showing the correlation between the true and estimated values of the discrimination parameters can be seen in Figure 4.6.

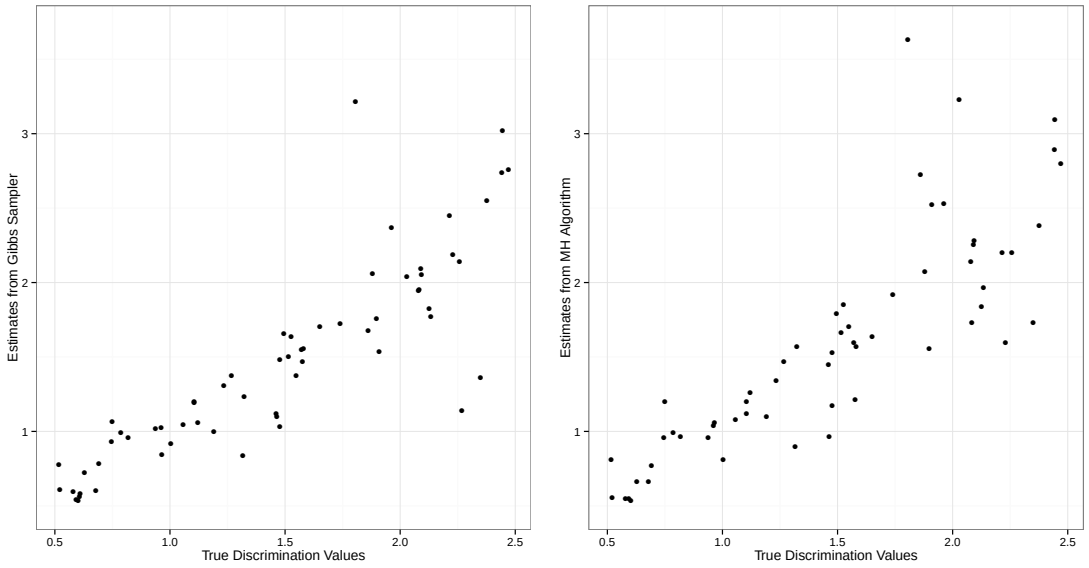

Figure 4.6: Scatter plots comparing the correlation between true and estimated values from the Gibbs Sampler and MH algorithm of the discrimination parameters in the 2PL model.

Discrimination parameter estimates for the least difficult items were consistently estimated to have a very high discrimination values. A closer inspection of the data revealed that nearly all subjects in that particular latent class answered those items correctly. The estimates are based on the likelihood of the responses, so it is reasonable that extremely high values of the discrimination parameter would be estimated.

For these simulations, MRM models were also fit. For each dataset, I ran the `mrm()` function specifying between one and four latent classes. The mixed Rasch model with the correct number of latent classes consistently had the lowest BIC

value, so it is reasonable to conclude that `mRm` was able to identify the correct number of latent classes.

The estimated proportions of the mixed Rasch model also consistently matched the true proportions (to within two-percentage points) of each latent class. The mixed Rasch model does not estimate discrimination parameters, so the estimated difficulty parameters can only be expected to be proportional to the true difficulty parameters.

Here, the fitted mixed Rasch models exhibited odd behavior. For the two-class models, the true and estimated values of the parameters lined up fairly well. Correlation between estimated values and true values were around 0.95, indicating that the fit was almost as good as the parameter estimates from the MH algorithm. Scatter plots comparing the correlation between the true difficulty values and the estimated values from the MH algorithm and those from the MRM can be seen in Figure 4.7.

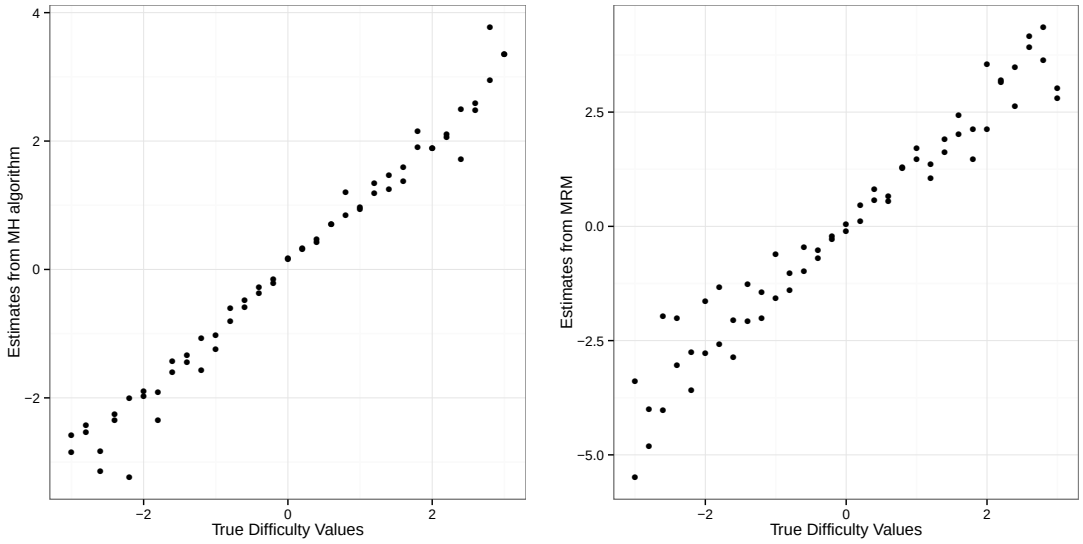

Figure 4.7: Scatter plots comparing the correlation between true and estimated values from the MH algorithm and Mixed Rasch Model of the difficulty parameters in the 2PL model with two latent classes.

On the other hand, estimates with the `mrm()` function for a three-class model did not produce expected results. Values for two of the latent classes had a correlation of 0.963 with the true values of the difficulty parameters. For the remaining latent class, however, the correlation between the estimated values and the true values was -0.976. This strong negative correlation could actually be a result of an estimation artifact, or perhaps an alternative parameterization of the model, rather than an indicator of a shortcoming of the estimation method. Repeated simulations produced similar results with the `mrm()` function. Further exploration of the `mrm()` function may need to be done to investigate why the parameter estimates are nearly perfect additive inverses of the true values for that latent component. Unfortunately, the function `mrm()` does not calculate person parameters or reveal the latent classes to which people are assigned. As such, it is not possible to know why the introduction of varying discrimination parameters led to the incorrect estimation of difficulty parameters for one of the latent classes. Scatter plots comparing the correlation between the true and estimated values from the MH algorithm and MRM in the three class model can be seen in Figure 4.8.

4.4 Difficulties

Both the Gibbs sampler and the MH algorithm had trouble identifying latent classes when only a few of the items exhibited DIF. For example, a simulation was performed where 27 items shared the same parameters across latent classes, and 4 of the items had DIF for two latent classes. Despite running a few different chains, both the Gibbs sampler and the MH algorithm consistently only found one latent class.

The parameter estimates for the 27 items with no DIF were accurate. The parameter estimates for the items with DIF were inaccurate. Correlation between

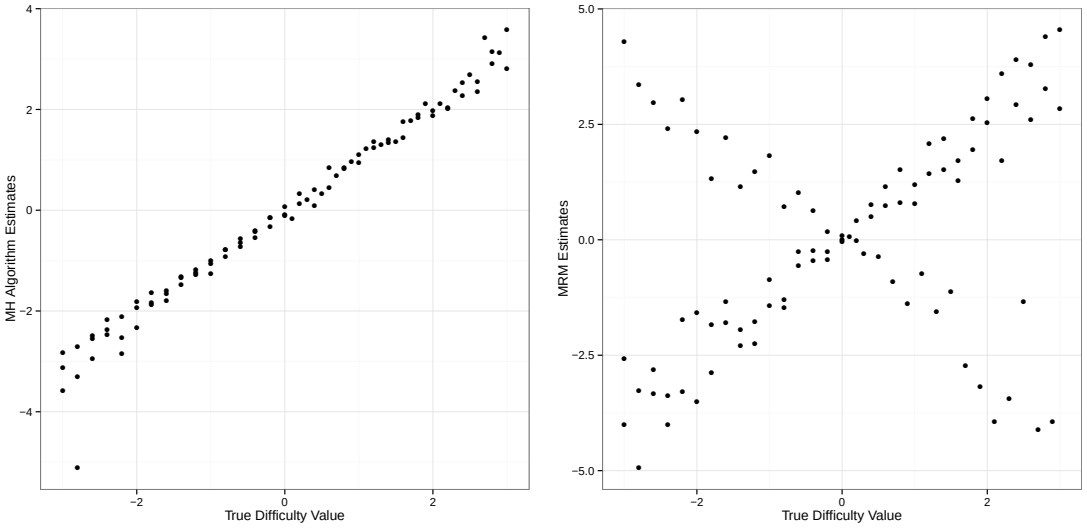

Figure 4.8: Scatter plots comparing the correlation between true and estimated values from the MH algorithm and Mixed Rasch Model of the difficulty parameters in the 2PL model with three latent classes.

the true and estimated values of the 27 items without DIF was 0.997. The items with DIF were estimated to have difficulties close to the average of the difficulties across latent classes, weighted by the proportion of the latent class.

For these models, the MRM function also struggled to correctly identify the proportions of latent classes, despite explicitly telling the function to search for two latent classes. For a model with a 70/30 split between latent classes, the MRM estimated that the proportion of latent classes was closer to a 95/5 split. The parameter estimates for the smaller component were also wildly off for all of the items.

A few other simulations produced surprising results. Data was simulated where the difficulty for one latent class was a sequence of items in increasing difficulty. For the other latent class, the difficulty of the items were in a random order. In some of these simulation, the MH algorithm correctly identified the latent classes and parameter estimates. In others, however, the algorithm would incorrectly identify only one latent class, or in some cases too many latent classes. As the

order of the items' difficulties for the second latent class is random, it is hard to know under which conditions the algorithm is correctly able to identify the appropriate number of latent classes. In many of these simulations, the MRM function outperformed the algorithm by correctly identifying the proportions of the two latent classes. It should be noted, however, the MRM function was given the correct number of latent classes to estimate.

	True Values		Gibbs Sampler		MH Algorithm	
	Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
Item 1	-3.00	3.00	-3.81	2.54	-2.83	3.11
Item 2	-2.79	2.79	-2.99	2.75	-2.74	2.78
Item 3	-2.59	2.59	-2.34	2.22	-2.49	2.62
Item 4	-2.38	2.38	-2.31	2.06	-2.26	2.27
Item 5	-2.17	2.17	-2.11	2.03	-2.17	1.96
Item 6	-1.97	1.97	-1.96	2.09	-1.82	1.70
Item 7	-1.76	1.76	-1.66	1.73	-1.73	1.79
Item 8	-1.55	1.55	-1.60	1.39	-1.60	1.27
Item 9	-1.34	1.34	-1.34	1.26	-1.34	1.23
Item 10	-1.14	1.14	-1.11	1.13	-0.96	1.19
Item 11	-0.93	0.93	-0.90	0.87	-0.85	0.83
Item 12	-0.72	0.72	-0.69	0.67	-0.67	0.69
Item 13	-0.52	0.52	-0.47	0.53	-0.49	0.54
Item 14	-0.31	0.31	-0.33	0.24	-0.26	0.32
Item 15	-0.10	0.10	-0.13	0.06	-0.10	0.03
Item 16	0.10	-0.10	0.09	-0.17	0.09	-0.09
Item 17	0.31	-0.31	0.31	-0.26	0.29	-0.34
Item 18	0.52	-0.52	0.48	-0.54	0.50	-0.56
Item 19	0.72	-0.72	0.70	-0.78	0.69	-0.54
Item 20	0.93	-0.93	0.97	-0.92	0.84	-0.83
...	...	...	...	...	...	...
Item 27	2.38	-2.38	2.62	-2.28	2.19	-2.16
Item 28	2.59	-2.59	2.78	-2.74	2.47	-2.56
Item 29	2.79	-2.79	3.00	-2.71	2.64	-2.69
Item 30	3.00	-3.00	3.15	-7.60	2.79	-2.99

Table 4.1: Comparison between true parameter values and estimates produced by the Gibbs sampler and MH algorithm

	True Values			Gibbs Sampler			MH Algorithm		
	Cl 1	Cl 2	Cl 3	Cl 1	Cl 2	Cl 3	Cl 1	Cl 2	Cl 3
Item 1	-3.00	3.00	0.00	-2.85	3.14	0.00	-5.62	3.23	-0.04
Item 2	-2.80	2.80	0.10	-3.00	2.95	0.13	-3.36	2.50	0.03
Item 3	-2.60	2.60	-0.20	-2.98	2.75	-0.25	-2.39	2.65	-0.15
...	...	...	...	...	...	...	...	...	...
Item 13	-0.60	0.60	-1.20	-0.56	0.71	-1.14	-0.58	0.59	-1.16
Item 14	-0.40	0.40	1.30	-0.39	0.43	1.05	-0.42	0.40	1.20
Item 15	-0.20	0.20	-1.40	-0.16	0.23	-1.48	-0.25	0.21	-1.31
...	...	...	...	...	...	...	...	...	...
Item 31	3.00	-3.00	-3.00	2.72	-3.58	-2.77	3.75	-3.42	-9.20

Table 4.2: Comparison of the true values and estimates of the difficulty parameters of a model with 3 latent classes.

	True Values			MH Algorithm		
	Class 1	Class 2	Class 3	Class 1	Class 2	Class 3
1	2.50	0.80	1.50	2.72	0.85	1.80
2	1.00	2.50	0.80	0.87	2.19	0.51
3	1.50	1.00	2.50	1.30	0.97	2.61
...	...	...	...	...	...	...
13	2.50	0.80	1.50	2.61	0.68	1.62
14	1.00	2.50	0.80	1.02	2.72	0.98
15	1.50	1.00	2.50	1.63	1.00	2.31
...	...	...	...	...	...	...
31	1.50	1.00	2.50	1.82	0.87	6.35

Table 4.3: Comparison of the true values and estimates from the MH algorithm of some of the discrimination parameters of a model with three latent classes.

CHAPTER 5

Discussion

The proposed models and algorithms can be used to identify latent classes in a mixture IRT model without the need to specify the number of latent classes present in the model. Both algorithms, however, seem to have fairly low power in identifying multiple latent classes, especially if the differences between the latent classes are not large.

The primary difficulties in the sampling algorithms are the steps where the latent class assignments for the individuals are sampled. In the algorithms, individual persons have no prior information associated with them regarding their latent class assignments. Both algorithms employ a categorical distribution to randomly assign an individual to a latent class. The new latent class assignment is accepted or rejected based on the likelihood of the individual's response, the current parameter estimates, and the probability of the class assignments. This relative lack of information leads to an algorithm that has low power to identify latent classes.

The use of a Dirichlet process or a stick-breaking prior for latent class assignments in theory should be able to allow for the identification of latent classes in IRT mixtures without the need to specify the number of components. The practical implementation, however, reveals some of the weaknesses of semi-parametric and non-parametric methods. With too few constraints or assumptions placed on the model, the parameter space can be too large to effectively find an optimal solution.

To be most effective, the algorithm needs to be strengthened by placing stronger prior beliefs on the number of latent classes in the model. A stick-breaking prior can accommodate up to an infinite number of latent classes. In reality, a dataset is generally not expected to have very many latent classes. The Dirichlet process incorporates the parameter α to account for how likely additional "sticks" are broken off.

The current implementations, however, seems to struggle with identifying the correct number of components. With the original parameters, the algorithm often breaks off 'too large' of a piece and assigns all of the individuals to one latent component, especially when the differences between latent classes were not extreme. On the other hand, modifying the proposal parameter of the sampler associated with the latent class mixing proportions led to solutions with too many latent classes, or chains that did not converge. Of course, if the goal is to identify the correct number of latent classes, the dependence of the solution on the adjustment of this prior-belief parameter is problematic.

At some level, the algorithm proposed here is not much more efficient than running multiple mixture Rasch models and comparing BIC values to choose the one with the optimal number of components. The proposed models, however, can fit 2PL models to the data, which is not possible with mixed Rasch models. When evaluating the fit of the model, it should be noted that unlike fitting the Mixed Rasch Model with a different number of latent classes, the proposed methods does not try to fit multiple models. Thus, model fit statistics that take into account the number of free parameters to be estimated, like AIC or BIC, do not provide more information than the likelihood of the data.

The lack of power to detect multiple latent classes in tests with only a few items exhibiting DIF was noted a 2011 study by DeMars and Lau [DL11]. That study looked at how accurate DIF detection is with a mixture Rasch model. The study found that IRT mixture models did not estimate effect sizes well for

cases where DIF affected only a small number of items. The study found that parameter recovery with IRT mixture models may not be very accurate in a variety of situations.

Further studies could be done with the proposed models in this dissertation. A more thorough examination of conditions can be explored to see when the model fails to correctly identify the latent classes. Computational performance of the Gibbs sampling algorithm can also be improved.

These models and sampling algorithms were proposed as a potential method to detect DIF. Based on the results of the simulations, however, I cannot recommend that they be used in a real-world scenario. It is unlikely that the kind of extreme DIF used in the simulations would occur in an actual setting. Both models failed to detect multiple latent classes for tests in which only a few items exhibit DIF. As I believe it is far more likely scenario to encounter tests where only a few items exhibit DIF, the proposed models are unlikely to perform well these situations.

The results of this dissertation add to the existing DIF literature a brief exploration of one of the limitations of latent class IRT analysis, while also demonstrating that a Dirichlet process prior can be used to detect latent classes in extreme cases.

APPENDIX A

R Code for the Gibbs Sampler

```
### For Printing


library(mvtnorm)
library(MASS)
library(truncnorm)


#generate random draws from a multivariate normal
multinor <- function(mu,sigma){
  siz = nrow(as.matrix(mu))
  SIG = chol(sigma)
  rm = rnorm(siz) #N(0,1)
  RV1 = mu + t(SIG)%*%as.matrix(rm)
  return(RV1)
}


N.persons <- 2000 #number of persons
L.components <- 5  #number of max components
J.items <- 30 #number of items


##########################################
####### Generate Response Data ###########
```

```
########################################
set.seed(2222)
delta <- matrix(0,nrow=2,ncol=J.items)
delta[1,] <- c(seq(-3,3, len=30))
delta[2,] <- c(seq(3,-3, len=30))
alpha <- matrix(0.5+runif(2*30),nrow=2,ncol=J.items)

theta <- rnorm(N.persons,0,1)
group <- c(rep(1,.77*N.persons), rep(2,.23*N.persons))
#latent class
dif <- matrix(0,nrow=N.persons,ncol=J.items) #difficulty
prob <- matrix(0,nrow=N.persons,ncol=J.items) #probability
y <- matrix(0,nrow=N.persons,ncol=J.items) #response matrix
delta.mat <- t(delta[1,]%o%(group==1))+ t(delta[2,]%o%
(group==2))
dif <-t(t(theta%o%rep(1,J.items)*alpha - alpha*delta.mat))
prob <- 1/(1+exp(-1.702*dif))
out <- matrix(runif(prod(dim(prob))),nrow=N.persons)
y <- (out<prob)+0
############## end data generation ###########

i <- 0 #counter for person
l <- 0 #counter for component
j <- 0 #counter for item

alpha.dp <- 1 #concentration parameter alpha for the dirichlet
process mixture
```

```
k <- rep(1,N.persons) #vector of component assignments
# k[i] is the component assignment of person i

kappa.component <- rep(1/L.components, L.components) #mixing
# proportions for each of the l components. initialized to 1/L

#initial values for alpha = 2,
#initial beta = inv.norm of the proportion correct
alpha.discrimination <- matrix(2, nrow=L.components,
ncol=J.items) #discrimination
beta.difficulty <- matrix(rep(-2*qnorm(colSums(y)/N.persons),
each=L.components), nrow=L.components, ncol=J.items)
#difficulty

xi <- list(0)
xi.hat <- list(0)
theta.ability <- rnorm(N.persons,0,1) #ability parameter

# reparameterized p(u=1) = 1/(1+exp(-1.702(alpha*theta-beta))
# eta = alpha*theta - beta
# z = eta + epsilon
eta.latent.ijk <- array(0, c(N.persons, J.items, L.components))
z.latent.ijk <- array(0, c(N.persons, J.items, L.components))
epsilon <- rnorm(N.persons)
eta.latent.ijk <- theta.ability %o% t(alpha.discrimination) -
rep(1,N.persons) %o% t(beta.difficulty)
z.latent.ijk <- eta.latent.ijk + epsilon
```

```
eta.latent.ij <- matrix(0,nrow=N.persons,ncol=J.items)
z.latent.ij <- matrix(0,nrow=N.persons,ncol=J.items)
for(i in 1:N.persons){
  eta.latent.ij[i,] <- eta.latent.ijk[i,,k[i]]
  z.latent.ij[i,] <- z.latent.ijk[i,,k[i]]
}

thin <- 3
MCMC <- 2600*thin
alpha.discrimination.mcmc <- array(0,c(L.components, J.items,
MCMC))
beta.difficulty.mcmc <- array(0,c(L.components, J.items, MCMC))
theta.ability.mcmc <- matrix(0,nrow=N.persons,ncol=MCMC)
kappa.component.mcmc <- matrix(0,nrow=L.components,ncol=MCMC)

pb <- txtProgressBar(max=MCMC, style=3)
for(counter in 1:MCMC){

################################################
##### part 1 of the sampler - generate xi ######
################################################

# find the subjects in the lth component
for(l in 1:L.components){
  subjects.in.l <- k==l
  if(sum(subjects.in.l)>1){ #skip for empty components
    x.sub.l = cbind(theta.ability[subjects.in.l],
rep(-1,sum(subjects.in.l)))
```

```
    z.sub.l = z.latent.ij[subjects.in.l,]
    xi.hat[[l]] = solve(t(x.sub.l)%*%x.sub.l)%*%
t(x.sub.l)%*%z.sub.l #XtX^(-1) XtZ
    for(j in 1:J.items){
      alpha.discrimination[l,j] <- 0
      while(alpha.discrimination[l,j] <=0){
#force alpha positive
        #generate random mv normal
        xi[[l]] <- mvrnorm(n=1, xi.hat[[l]][,j],
solve(t(x.sub.l)%*%x.sub.l))
        alpha.discrimination[l,j] <- xi[[l]][1]
        beta.difficulty[l,j] <- xi[[l]][2]
      }
    }
  }
  if(sum(subjects.in.l)<=1){
    for(j in 1:J.items){
      alpha.discrimination[l,j] <- rtruncnorm(1, a=0,
mean=2, sd=.5)
      beta.difficulty[l,j] <- rnorm(1,0,.5)
    }}
}


################################################
# part 2 - generate latent class proportions ##
################################################
```

```
k.star   <- colSums(rmultinom(N.persons,1,kappa.component)*
   (1:L.components))
k.old <- k

epsilon <- rnorm(N.persons)
eta.latent.ijk <- theta.ability %o% t(alpha.discrimination) -
   rep(1,N.persons) %o% t(beta.difficulty)
z.latent.ijk <- eta.latent.ijk + epsilon
prob.y <- 1/(1 + exp(-1.702*z.latent.ijk))

old.prob <- 0
new.prob <- 0
for(i in 1:N.persons){
old.prob[i] <- log(kappa.component[k[i]]) +
sum(log(prob.y[i,,k[i]]^y[i,]) +
log((1-prob.y[i,,k[i]])^(1-y[i,])))
new.prob[i] <- log(kappa.component[k.star[i]]) +
sum(log(prob.y[i,,k.star[i]]^y[i,]) +
log((1-prob.y[i,,k.star[i]])^(1-y[i,])))
}

accept <- pmin(0,new.prob-old.prob)
k <- ifelse(log(runif(N.persons)) < accept, k.star, k)

#####################################################
##### part 3 of the sampler - generate kappa #####
#####################################################
# sample kappa and v from the beta -
```

```
# the stick breaking construction of dirichlet process
ak <- rep(alpha.dp/L.components,L.components-1)
bk <- 0
vk <- 0
mk <- 0
for(l in 1:(L.components-1)){
  bk[l] = alpha.dp - l*alpha.dp/L.components
}
for(l in 1:L.components){
  mk[l] = sum(k == l)
}

for(l in 1:(L.components-1)){
  vk[l] <- rbeta(1, max(1,ak[l] + mk[l]*alpha.dp/L.components),
max(1,bk[l] + *alpha.dp/L.components*sum(mk[(l+1):
L.components])) )
}

vk[L.components] = 1

kappa.component = 0
kappa.component[1] = vk[1]
for(l in 2:L.components){
  kappa.component[l] = prod(1-vk[1:(l-1)])*vk[l]
}

#####################################################
######## part 4 of the sampler - Sample Z ########
```

```
##################################################

for(i in 1:N.persons){
  eta.latent.ij[i,] <- eta.latent.ijk[i,,k[i]]
}

for(i in 1:N.persons){
  for(j in 1:J.items){
    if(y[i,j] == 1){
      z.latent.ij[i,j] <- rtruncnorm(1,a=0,
mean=eta.latent.ij[i,j],sd=1)
    }
    if(y[i,j] == 0){
      z.latent.ij[i,j] <- rtruncnorm(1,b=0,
mean=eta.latent.ij[i,j],sd=1)
    }
  }
}

## Sample theta
for(i in 1:N.persons){
  denom = sum(alpha.discrimination[k[i],]^2)+1
  num = sum(alpha.discrimination[k[i],] * (z.latent.ij[i,] +
beta.difficulty[k[i],]) )
  theta.ability[i] <- rnorm(1,mean=num/denom, sd=1/sqrt(denom))
}

#### data dump ####
```

```
if(counter%%thin==0){
  alpha.discrimination.mcmc[,,counter/thin] <-
alpha.discrimination
  beta.difficulty.mcmc[,,counter/thin] <- beta.difficulty
  theta.ability.mcmc[,counter/thin] <- theta.ability
  kappa.component.mcmc[,counter/thin] <- kappa.component
}
  setTxtProgressBar(pb, counter)
}
```

APPENDIX B

R Code for the Metropolis-Hastings Algorithm

```
# Generate 3 latent class
require(MCMCpack)
require(truncnorm)
require(coda)

set.seed(333333)
N.persons = 3000
L.components <- 3  # number of components for data generation
M.components <- 8  # max number of components allowed in the
                   # sampler
J.items <- 31      # number of items
kappa <- c(.5,.3,.2)

a.disc <- matrix(.5+2*runif(J.items*L.components),
    nrow=J.items,ncol=L.components)
alt.seq <- rep(c(-1,1),length=31)*rep(seq(0,3,length=31))
b.diff <- matrix(c(seq(-3,3,length=J.items),seq(3,-3,
    length=J.items),alt.seq), nrow=J.items, ncol=L.components)
k <- rmultinom(N.persons,1,kappa) #class assignments
k <- k[order(rowSums(k),decreasing=TRUE),]
# rearrange the random k, so the largest class is first
```

```
## generate the person ability parameters
mean.theta <- 0
theta.abl  <- rnorm(N.persons, mean=mean.theta, sd=1)

## the N x J matrix of response probabilities
discrimination <- a.disc%*%k
    #discrimination for the appropriate class
difficulty <- b.diff%*%k
term.1 <- t(discrimination)*theta.abl
term.2 <- t(discrimination*difficulty)
P.prob <- plogis(term.1-term.2)
    # 1/(1 + exp(term.2 - term.1))

## generate the 0/1 responses U as a matrix of Bernoulli draws
U <- ifelse(runif(J.items*N.persons) < P.prob,1,0)

#####################################################################
## Define a single iteration of the MH within Gibbs sampler
#####################################################################
blocked.mcmc.update  <- function( U.data, cur){
  cur <-  sample.th(U.data, cur)
  cur <-  sample.a(U.data, cur)
  cur <-  sample.b(U.data, cur)
  cur <-  sample.k(U.data, cur)
  cur <-  sample.kappa(U.data, cur)
  return(cur)
}
```

```
# hyper parameters
hyperpars <- list( mu.a  = 1.185, s2.a  = 1.185, s2.b  = 100,
    alpha.th = 1, beta.th = 1 )

# Define a complete run of the MH within Gibbs sampler
run.chain.2pl.mix <- function(M.burnin, M.keep, M.thin,
    U.data, hyperpars, th.init, a.init, b.init,
    k.init, kappa.init, MH.th, MH.a, MH.b, MH.kappa,
    verbose=FALSE) {

  # list of things to keep track of in the "current state"
  # of the chain
  cur <<- list(th=th.init, a=a.init, b=b.init, k = k.init,
  kappa = kappa.init, hyperpars=hyperpars,
  MH = list(th=MH.th, a=MH.a, b=MH.b, kappa=MH.kappa),
  ACC= list(th=0,th.n=0, a=0,a.n=0,  b=0,b.n=0, k=0, k.n=0 ))

  ## Define matrix to store MCMC results...
  chain.keep <- matrix( NA, ncol = M.keep, nrow =
    length(th.init) + length(a.init) + length(b.init) +
    length(k.init) + length(kappa.init))
  rownames(chain.keep) <- c(
    paste('theta.abl', 1:length(th.init)),
    paste('a.disc', 1:dim(a.init)[1],
          rep(1:dim(a.init)[2],each=dim(a.init)[1])),
    paste('b.diff', 1:dim(b.init)[1],
          rep(1:dim(b.init)[2],each=dim(b.init)[1])),
```

```
  paste('k.class', 1:dim(k.init)[1], rep(1:dim(k.init)[2],
      each=dim(k.init)[1])),
  paste('kappa',  1:length(kappa.init))
)

# Burn-in phase
if( M.burnin > 0 ) {
  for(ii in 1:M.burnin ) {
    cur <- blocked.mcmc.update(U.data, cur)
  }}

# Keep these results after thinning
for (m in 1:M.keep) {
  # Skip the "thinned" pieces of the chain
  if( M.thin > 1 ) {
    for( ii in 1:(M.thin-1) ) {
      cur <- blocked.mcmc.update(U.data, cur)
    }}
  # Generate a "kept" update and save its results
  cur <<- blocked.mcmc.update(U.data, cur)
  chain.keep[,m] <- c( cur$th, as.numeric(cur$a),
      as.numeric(cur$b), as.numeric(cur$k), cur$kappa)

  if ( m %% 100 == 0) {
    print(m) # displays progress of the chain
  }
}
```

```
  if (verbose) {
    cat(paste("Average acceptance rates:",
    "\n theta.abl:", round(cur$ACC$th / cur$ACC$th.n,3),
    "\n a.disc:   ", round(cur$ACC$a  / cur$ACC$a.n,3),
    "\n b.diff:   ", round(cur$ACC$b  / cur$ACC$b.n,3),
    "\n k.class:  ", round(cur$ACC$k  / cur$ACC$k.n,3),
    "\n"))
  }
  return( chain.keep )
}

# log.prob function
log.prob <- function(U.data, th, a, b, k) {
  # Build the matrix of probabilities
  discrimination <- a%*%k
    #discrimination for the appropriate class
  difficulty <- b%*%k
  term.1  <- t(discrimination)*th
  term.2 <- t(discrimination*difficulty)
  P.prob <- plogis(term.1-term.2)
    # 1/(1 + exp(term.2 - term.1))
  ## Build the likelihood
  log.bernoulli  <- log(P.prob^U.data) +
    log((1-P.prob)^(1-U.data))
  return(log.bernoulli)
}

sample.th <- function(U.data, old) {
```

```
th.old    <- old$th
MH.th     <- old$MH$th
th.star   <- rnorm(N.persons,th.old,MH.th)

## Calculate the probability of accepting the proposal draw
log.cc.star <- (
  apply(log.prob(U.data, th.star, old$a, old$b, old$k),1,sum)
  + dnorm(th.star,0,1,log=TRUE)
)
log.cc.old  <- (
  apply(log.prob(U.data, th.old, old$a, old$b, old$k),1,sum)
  + dnorm(th.old,0,1,log=TRUE)
  )
log.prop.star <- 0
log.prop.old <- 0

## acceptance probability on the log scale
log.alpha.star <- pmin(log.cc.star + log.prop.old -
  log.cc.old - log.prop.star,0)

## Accept or reject the proposals with the
## calculated acceptance probabilities
acc.new <- ifelse(log(runif(N.persons)) <= log.alpha.star,
  1, 0)

## Update the chain
cur          <- old
cur$th       <- ifelse(acc.new==1, th.star, th.old)
```

```
  ## Calculate acceptance probabilities
  cur$ACC$th   <- old$ACC$th + mean( acc.new )
  cur$ACC$th.n <-  cur$ACC$th.n + 1
  return(cur)
}


sample.b <- function(U.data, old) {
  b.old    <- old$b
  MH.b     <- old$MH$b
  classes  <- dim(b.old)[2]
  b.star   <- matrix(rnorm(length(b.old),b.old,old$MH$b),
    ncol=dim(b.old)[2])

  ## Calculate the probability of accepting the proposal draw
  log.cc.star <- (
    t(old$k%*%log.prob(U.data, old$th, old$a, b.star, old$k))
    + dnorm(b.star, 0, sqrt(old$hyperpar$s2.b),log=TRUE)
  )
  log.cc.old  <- (
    t(old$k%*%log.prob(U.data, old$th, old$a, b.old, old$k))
    + dnorm(old$b, 0, sqrt(old$hyperpar$s2.b),log=TRUE)
  )
  log.prop.star <- 0
  log.prop.old <- 0

  ## acceptance probability on the log scale
```

```
  log.b.star <- pmin(log.cc.star + log.prop.old - log.cc.old -
        log.prop.star,0)

  ## Accept or reject the proposals with the
  ## calculated acceptance probabilities
  acc.new <- ifelse(log(runif(length(b.old)))<=log.b.star,
        1, 0)

  ## Update the chain
  cur         <- old
  cur$b       <- ifelse(acc.new==1, b.star, b.old)

  ## Calculate acceptance probabilities
  cur$ACC$b   <- old$ACC$b + mean( acc.new )
  cur$ACC$b.n <-  cur$ACC$b.n + 1

  return(cur)
}

sample.a <- function(U.data, old) {
  a.old    <- old$a
  MH.a     <- old$MH$a
  classes  <- dim(a.old)[2]
  a.star   <- matrix(rnorm(length(a.old),mean=a.old,
    sd=old$MH$a), ncol=classes)

  ## Calculate the probability of accepting the proposal draw
  log.cc.star <- (
```

```
  t(old$k%*%log.prob(U.data, old$th, a.star, old$b, old$k))
  + dlnorm(a.star, old$hyperpar$mu.a,
  sqrt(old$hyperpar$s2.a), log=TRUE)
)
log.cc.old  <- (
  t(old$k%*%log.prob(U.data, old$th, a.old, old$b, old$k))
  + dlnorm(old$a, old$hyperpar$mu.a,
  sqrt(old$hyperpar$s2.a), log=TRUE)
)
log.prop.star <- log(dnorm(a.star,a.old,MH.a))
log.prop.old  <- log(dnorm(a.old,a.star,MH.a))

## ... Calculate the acceptance probability on the log scale
log.alpha.star <- pmin(log.cc.star + log.prop.old -
  log.cc.old - log.prop.star,0)

## Accept or reject the proposals with the
## calculated acceptance probabilities
acc.new <- ifelse(log(runif(length(a.old)))<=
  log.alpha.star, 1, 0)

## Update the chain
cur <- old
cur$a <- ifelse(acc.new==1, a.star, a.old)

## Calculate acceptance probabilities (for tuning)
cur$ACC$a   <- old$ACC$a + mean( acc.new )
cur$ACC$a.n <-  cur$ACC$a.n + 1
```

```
  return(cur)
}

sample.k <- function(U.data, old) {
  k.old    <- old$k
  MH.k     <- old$MH$k
  classes  <- dim(k.old)[1]
  kappa.old    <- old$kappa

  k.star   <- rmultinom(N.persons,1,kappa.old)
    #class assignments

  ## Calculate the probability of accepting the proposal draw
  log.cc.star <- (
    as.matrix(rowSums(log.prob(U.data, old$th, old$a,
        old$b, k.star)))
      + t(log(t(kappa.old)%*%k.star))
  )
  log.cc.old  <- (
    as.matrix(rowSums(log.prob(U.data, old$th, old$a,
        old$b, k.old)))
      + t(log(t(kappa.old)%*%k.old))
  )

  log.prop.star <- 0
  log.prop.old  <- 0
```

```
  ## Calculate the acceptance probability on the log scale
  log.alpha.star <- pmin(log.cc.star + log.prop.old -
    log.cc.old - log.prop.star,0)

  ## Accept or reject the proposals with the
  ## calculated acceptance probabilities
  decision <- ifelse(log(runif(N.persons))<=log.alpha.star,
    1, 0)
  acc.new <- matrix(rep(decision, classes),ncol=classes)

  ## Update the chain
  cur          <- old
  cur$k       <- ifelse(t(acc.new)==1, k.star, k.old)

  ## Calculate acceptance probabilities
  cur$ACC$k   <- old$ACC$k + mean( acc.new )
  cur$ACC$k.n <- cur$ACC$k.n + 1

  return(cur)
}

sample.kappa <- function(U.data, old) {
  kappa.old    <- old$kappa
  classes  <- dim(old$k)[1]
  persons <- dim(old$k)[2]

  counts <- apply(old$k,1,sum)
  Mm <- 0
```

```
for(l in 2:length(counts)){
  Mm[l] <- sum(counts[l:length(counts)])
}

v <- 0
for(l in 1:(length(counts)-1)){
  v[l] <- rbeta(1,max(1,10*counts[l]/persons,na.rm=TRUE),
      max(1,10*Mm[l+1]/persons,na.rm=TRUE))
}
v[length(counts)] <- 1
v <- ifelse(is.na(v),rbeta(1,1,1),v)

kappa.star <- 0
kappa.star[1] <- v[1]  #stick breaking construction
for(l in 2:M.components){
  kappa.star[l] = prod(1-v[1:(l-1)])*v[l]
}

## Calculate the probability of accepting the proposal draw
log.cc.star <- (
  sum(log(kappa.star%*%old$k)) +
  log(ddirichlet(kappa.star,kappa.old))
)
log.cc.old  <- (
  sum(log(kappa.old%*%old$k)) +
  log(ddirichlet(kappa.old,kappa.old))
)
```

```
  log.prop.star <- 0
  log.prop.old  <- 0

  ## Calculate the acceptance probability on the log scale
  log.alpha.star <- log.cc.star + log.prop.old - log.cc.old -
                            log.prop.star

  ## Accept or reject the proposals with the
  ## calculated acceptance probabilities
  acc.new <- ifelse(log(runif(1))<=log.alpha.star, 1, 0)

  ## Update the chain
  cur <- old
  cur$kappa <- if(acc.new==1 && sum(is.na(kappa.star))==0 ){
      cur$kappa <- kappa.star
      }else{
      cur$kappa <- kappa.old}

  return(cur)
}

set.seed(333333)
cur <- list()
run.A <- run.chain.2pl.mix(
  M.burnin = 0, M.keep  = 2000, M.thin = 1,
  U.data  = U,
  hyperpars = hyperpars,
  th.init = rep(0,N.persons),
```

```
a.init  = matrix(1.5,nrow=J.items,ncol=M.components),
b.init  = matrix(0,nrow=J.items,ncol=M.components),
kappa.init = rep(1/M.components, M.components),
k.init = rmultinom(N.persons,1,rep(1/M.components,
  M.components)),
MH.th=.85, MH.a=.15, MH.b=.15, MH.kappa=0.25, verbose=TRUE)
```

References

[Alb92] James H Albert. "Bayesian estimation of normal ogive item response curves using Gibbs sampling." *Journal of Educational and Behavioral Statistics*, **17**(3):251–269, 1992.

[BA81] R Darrell Bock and Murray Aitkin. "Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm." *Psychometrika*, **46**(4):443–459, 1981.

[BJ06] David M Blei, Michael I Jordan, et al. "Variational inference for Dirichlet process mixtures." *Bayesian analysis*, **1**(1):121–143, 2006.

[Cam92] Gregory Camilli. "A conceptual analysis of differential item functioning in terms of a multidimensional item response model." *Applied Psychological Measurement*, **16**(2):129–147, 1992.

[CB05] Allan S Cohen and Daniel M Bolt. "A mixture model analysis of differential item functioning." *Journal of Educational Measurement*, **42**(2):133–148, 2005.

[CGC11] Seung W Choi, Laura E Gibbons, and Paul K Crane. "Lordif: An R package for detecting differential item functioning using iterative hybrid ordinal logistic regression/item response theory and Monte Carlo simulations." *Journal of Statistical Software*, **39**(8):1, 2011.

[CKW96] Allan S Cohen, Seock-Ho Kim, and James A Wollack. "An investigation of the likelihood ratio test for detection of differential item functioning." *Applied Psychological Measurement*, **20**(1):15–26, 1996.

[Cur10] S McKay Curtis et al. "BUGS code for item response theory." *Journal of Statistical Software*, **36**(1):1–34, 2010.

[DKS02] Ralph J De Ayala, Seock-Ho Kim, Laura M Stapleton, and C Mitchell Dayton. "Differential item functioning: A mixture distribution conceptualization." *International Journal of Testing*, **2**(3-4):243–276, 2002.

[DL07] Dato NM De Gruijter, J Th Leo, et al. *Statistical test theory for the behavioral sciences*. CRC Press, 2007.

[DL11] Christine E DeMars and Abigail Lau. "Differential Item Functioning Detection With Latent Classes: How Accurately Can We Detect Who Is Responding Differentially?" *Educational and Psychological Measurement*, p. 0013164411404221, 2011.

[Emb07] Susan E Embretson. "Mixed Rasch models for measurement in cognitive psychology." In *Multivariate and mixture distribution Rasch models*, pp. 235–253. Springer, 2007.

[Fer73] Thomas S Ferguson. "A Bayesian analysis of some nonparametric problems." *The annals of statistics*, pp. 209–230, 1973.

[FRG09] Carolyn F Furlow, Terris Raiford Ross, and Phill Gagné. "The impact of multidimensionality on the detection of differential bundle functioning using simultaneous item bias test." *Applied Psychological Measurement*, **33**(6):441–464, 2009.

[FSL12] Hannah Frick, Carolin Strobl, Friedrich Leisch, and Achim Zeileis. "Flexible Rasch mixture models with package psychomix." 2012.

[HT88] Paul W Holland and Dorothy T Thayer. "Differential item performance and the Mantel-Haenszel procedure." *Test validity*, pp. 129–145, 1988.

[IJ01] Hemant Ishwaran and Lancelot F James. "Gibbs sampling methods for stick-breaking priors." *Journal of the American Statistical Association*, **96**(453), 2001.

[IZ02] Hemant Ishwaran and Mahmoud Zarepour. "Exact and approximate sum representations for the Dirichlet process." *Canadian Journal of Statistics*, **30**(2):269–283, 2002.

[Joh07] Matthew S Johnson. "Marginal maximum likelihood estimation of item response models in R." *Journal of Statistical Software*, **20**(10):1–24, 2007.

[JV] Patz R. J. Junker, B. W. and N. VanHoudnos. *Markov Chain Monte Carlo for Item Response models*. Chapman & Hall/CRC.

[KC95] Seock-Ho Kim and Allan S Cohen. "A comparison of Lord's chi-square, Raju's area measures, and the likelihood ratio test on detection of differential item functioning." *Applied Measurement in Education*, **8**(4):291–312, 1995.

[LST09] David Lunn, David Spiegelhalter, Andrew Thomas, and Nicky Best. "The BUGS project: evolution, critique and future directions." *Statistics in medicine*, **28**(25):3049–3067, 2009.

[MH07] Patrick Mair and Reinhold Hatzinger. "Extended Rasch modeling: The eRm package for the application of IRT models in R." 2007.

[MH09] Kei Miyazaki and Takahiro Hoshino. "A Bayesian semiparametric item response model with Dirichlet process priors." *Psychometrika*, **74**(3):375–393, 2009.

[MHM14] P. Mair, R. Hatzinger, and Maier M.J. *eRm: Extended Rasch Modeling*, 2014. R package version 0.15-4.

[Nea00] Radford M Neal. "Markov chain sampling methods for Dirichlet process mixture models." *Journal of computational and graphical statistics*, **9**(2):249–265, 2000.

[Pen03] Randall D Penfield. "Applying the Breslow-Day test of trend in odds ratio heterogeneity to the analysis of nonuniform DIF." *Alberta Journal of Educational Research*, **49**(3):231–243, 2003.

[PF12] David Preinerstorfer and Anton K Formann. "Parameter recovery and model selection in mixed Rasch models." *British Journal of Mathematical and Statistical Psychology*, **65**(2):251–262, 2012.

[PJ99] Richard J Patz and Brian W Junker. "A straightforward approach to Markov chain Monte Carlo methods for item response models." *Journal of educational and behavioral Statistics*, **24**(2):146–178, 1999.

[Plu03] Martyn Plummer et al. "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." In *Proceedings of the 3rd international workshop on distributed statistical computing*, volume 124, p. 125. Vienna, 2003.

[Ras60] Georg Rasch. "Probabilistic models for some intelligence and achievement tests." *Copenhagen: Danish Institute for Educational Research*, 1960.

[Riz06] Dimitris Rizopoulos. "ltm: An R package for Latent Variable Modelling and Item Response Theory Analyses." *Journal of Statistical Software*, **17**(5):1–25, 2006.

[Rob15] Alexander Robitzsch. *sirt: Supplementary Item Response Theory Models*, 2015. R package version 1.3.

[Ros90] Jürgen Rost. "Rasch models in latent classes: An integration of two approaches to item analysis." *Applied Psychological Measurement*, **14**(3):271–282, 1990.

[RS96] Louis A Roussos and William F Stout. "Simulation Studies of the Effects of Small Sample Size and Studied Item Parameters on SIBTEST and Mantel-Haenszel Type I Error Performance." *Journal of Educational Measurement*, **33**(2):215–230, 1996.

[Sam05] Karen M Samuelsen. *Examining differential item functioning from a latent class perspective*. PhD thesis, University of Maryland, 2005.

[Set94] Jayaram Sethuraman. "A Constructive Definition of Dirichlet Priors." *Statistica Sinica*, **4**:639–650, 1994.

[Van] N. VanHoudnos. "Tips, tricks, and tuning advice for implementing a Metropolis-Hastings within Gibbs sampler for a 2PL IRT model with R.".

[Zum99] Bruno D Zumbo. "A handbook on the theory and methods of differential item functioning (DIF): Logistic regression modeling as a unitary framework for binary and Likert-type (ordinal) item scores." 1999.