THE APPLICATION AND INTERPRETATION OF THE TWO-PARAMETER ITEM RESPONSE MODEL IN THE CONTEXT OF REPLICATED PREFERENCE TESTING

by

ZACH BUTTON

B.S., Kansas State University, 2013

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
Dr. Suzanne Dubnicka

Abstract

Preference testing is a popular method of determining consumer preferences for a variety of products in areas such as sensory analysis, animal welfare, and pharmacology. However, many prominent models for this type of data do not allow different probabilities of preferring one product over the other for each individual consumer, called overdispersion, which intuitively exists in real-world situations. We investigate the Two-Parameter variation of the Item Response Model (IRM) in the context of replicated preference testing. Because the IRM is most commonly applied to multiple-choice testing, our primary focus is the interpretation of the model parameters with respect to preference testing and the evaluation of the model's usefulness in this context. We fit a Bayesian version of the Two-Parameter Probit IRM (2PP) to two real-world datasets, *Raisin Bran* and *Cola*, as well as five hypothetical datasets constructed with specific parameter properties in mind. The values of the parameters are sampled via the Gibbs Sampler and examined using various plots of the posterior distributions. Next, several different models and prior distribution specifications are compared over the *Raisin Bran* and *Cola* datasets using the Deviance Information Criterion (DIC). The Two-Parameter IRM is a useful tool in the context of replicated preference testing, due to its ability to accommodate overdispersion, its intuitive interpretation, and its flexibility in terms of parameterization, link function, and prior specification. However, we find that this model brings computational difficulties in certain situations, some of which require creative solutions. Although the IRM can be interpreted for replicated preference testing scenarios, this data typically contains few replications, while the model was designed for exams with many items. We conclude that the IRM may provide little evidence for marketing decisions, and it is better-suited for exploring the nature of consumer preferences early in product development.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to offer my special thanks to Dr. Suzanne Dubnicka for the opportunity to undertake this research and for all the guidance throughout the process. I would also like to extend my gratitude to Dr. Perla Reyes and Dr. Christopher Vahl for graciously giving their time to serve on my committee, as well as Dr. Gary Gadbury, who was always willing to help me succeed. Finally, I wish to thank my family, friends, and all those involved in the K-State Choral Program for their unending support through the years. Go State!

Chapter 1 - Background

Preference Testing

Sensory analysis is the application of statistical techniques to the evaluation of human senses, such as taste or smell, often to gain knowledge about product attributes and consumer preferences. The discipline rose from the early days of product trade: potential customers began to test small samples of the product of interest as representations of the entire product's quality. Sensory analysis has advanced alongside the economy, now generating large quantities of data and utilizing sophisticated statistical methods for analysis.

A common approach is the *preference test*, which presents some number of product samples to a group of consumers and asks for the preferred sample to be selected. In this case, the product testers may be referred to as 'panelists' or 'consumers'. The preference test is often in the form of a choice between two products, which results in binary data. Most conventional preference testing analysis methods are based on inference for p_A, the probability of preferring product A over product B. This can be accomplished by the two-sided binomial test, where the null value of the proportion of A preferences is usually 0.50. Using the binomial distribution, we can then calculate the significance of the observed proportion of preferences for product A, $\hat{p}_A$. With a large sample size, a normal approximation confidence interval is easily accessible:

$$\hat{p}_A \pm z_{\alpha/2}\sqrt{\frac{\hat{p}_A(1-\hat{p}_A)}{n}} \qquad \textbf{(1.1)}$$

where n is the number of consumers and $z_{\alpha/2}$ is the standard normal quantile with an upper tail probability of $\alpha/2$. However, these methods must often assume that the panelists experience the same probability of preferring product A, which is a difficult assertion to accept.

Bi (2003) attempted to avoid this complication by allowing p_A to change between consumers through a Bayesian approach to the Binomial model. This method does allow prior beliefs about p_A to be reflected in the final estimates, but it is still a single estimate for all consumers. To mend this difficulty, we need a model which can accommodate individual probabilities of preferring product A for all n consumers, denoted by $p_1, \ldots, p_n$. Logistic regression allows this probability to change between consumers based on individual levels of the model's covariates, but covariate information is not typically collected in preference testing.

This suggests the use of multiple test replications for each consumer, advocated by several authors. Greenberg and Collins (1966) concluded that a single taste test may not accurately determine consumer preferences and that a two-trial taste test is much more powerful. Wilke, Cochrane, and Chambers (2006) argued that multiple tests could diagnose inconsistent preferences, which were identified in their example data by increasing proportions of preferences for product A over time. Although multiple replications are useful in determining consumer preference patterns, we need a model that can account for the changing probability of preferring product A over time, known as overdispersion. Cochrane, Dubnicka, and Loughin (2005) compared the power and Type 1 error rates of several methods of analysis which adjust for overdispersion. They concluded that the generalized linear model with a Pearson adjustment was the simplest of the models related to the binomial assumption, and the normal method also performs well, even though it does not explicitly correct for overdispersion. In the process, Cochrane et al. (2005) also found that single preference tests were often too liberal or conservative, resulting in misleading conclusions about consumer preferences. In contrast, their analysis of replicated preference tests produced more stabilized results.

The beta-binomial model is a popular alternative when multiple preference tests are being considered. Ennis and Bi (1998) discuss its advantages over the binomial model, namely the ability for inter-trial probabilities to vary, thus allowing for overdispersion. Meyners (2007) also supports the beta-binomial and provides guidelines for proper interpretation.

In order to allow consumer preferences to differ among subjects and gather information about preference trends across replications, we apply the Two-Parameter Item Response Model under a Bayesian framework. The focus of this work is the interpretation of this model's parameters in the context of replicated preference testing. As a secondary objective, we would also like to investigate how useful the model could be for industry executives making product decisions based on sensory analysis results.

Bayesian Methods and Gibbs Sampling

We will assume that the reader has some familiarity with the basics of Bayesian methods and Markov chain Monte Carlo (MCMC) simulation, so this report will only provide a brief overview of their application. Carlin and Louis (2009) provide a high-quality introduction to these methods. Recall that Bayesian estimation treats model parameters as random variables and utilizes some prior knowledge about the quantities of interest, in the form of prior distributions. The prior distributions work with the data likelihood to form the posterior distribution, which represents our updated beliefs about the parameters given the observed data. This can be viewed through the relationship

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\, f(\boldsymbol{y}|\boldsymbol{\theta}) \qquad \textbf{(1.2)}$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_p)$ are the parameters of interest, $\boldsymbol{y} = (y_1, \dots, y_n)$ represents the observed data. Here, p represents the joint posterior distribution, where π is the joint prior distribution and

f is the data likelihood. Once this posterior distribution is obtained, point and interval estimates are easily calculated using characteristics of the distribution itself.

However, this joint posterior distribution may be very complex, especially with more than one parameter, preventing direct calculation of the density function. MCMC simulation is especially useful in situations with multiple parameters. In these cases, which occur often in Bayesian statistics, it is common to apply an MCMC simulation algorithm called the Gibbs Sampler to obtain the parameters' posterior densities, a method first introduced by Geman and Geman (1984). The algorithm works by iteratively sampling from each parameter's full conditional distribution $g_i(\theta_i|\boldsymbol{\theta}_{-i}, \boldsymbol{y})$, where $\boldsymbol{\theta}_{-i} = (\theta_1, \dots, \theta_{i-1}, \theta_{i+1}, \dots, \theta_p)$. These full conditional densities may be very simple to sample from. In an instance when the full conditional density is not easily accessible, an MCMC algorithm developed by Metropolis et al. (1953), called the Metropolis-Hastings algorithm, can be used to sample from that distribution. To begin the algorithm, first set $t = 0$ and $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)}, \dots, \theta_p^{(0)})$. These initial values can be easily set to the desired prior means, but the choice usually remains inconsequential. At iteration t, repeat the following steps:

1) Draw $\theta_1^{(t)} \sim g_1(\theta_1|\theta_2^{(t-1)}, \theta_3^{(t-1)}, \dots, \theta_p^{(t-1)}, \boldsymbol{y})$

2) Draw $\theta_2^{(t)} \sim g_2(\theta_2|\theta_1^{(t)}, \theta_3^{(t-1)}, \dots, \theta_p^{(t-1)}, \boldsymbol{y})$

3) Draw $\theta_3^{(t)} \sim g_3(\theta_3|\theta_1^{(t)}, \theta_2^{(t)}, \dots, \theta_p^{(t-1)}, \boldsymbol{y})$

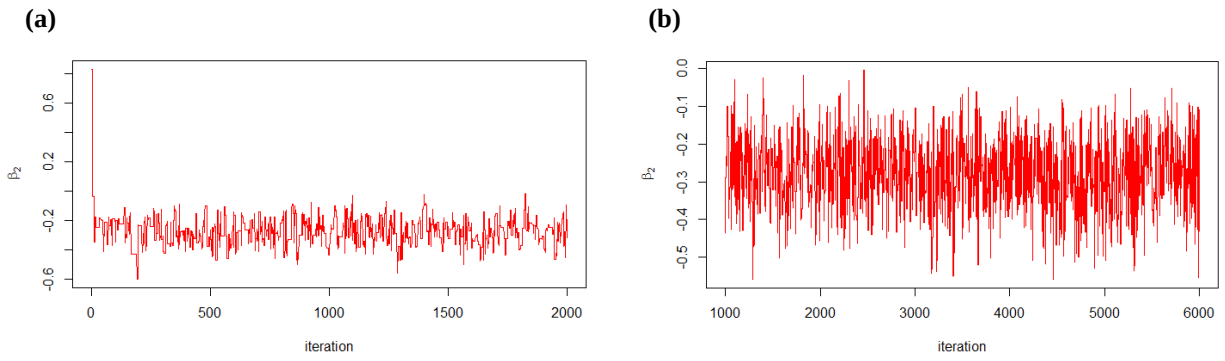
…

p) Draw $\theta_p^{(t)} \sim g_p(\theta_p|\theta_1^{(t)}, \theta_2^{(t)}, \theta_3^{(t)}, \dots, \theta_{p-1}^{(t)}, \boldsymbol{y})$

Note that the distribution in Step 2 is conditional on the most recently sampled value of θ_1, rather than $\theta_1^{(t-1)}$. This stipulation is used throughout the algorithm, as each step samples its respective parameter conditional on the most recent values of the other parameters.

It can be shown that this algorithm converges to the joint posterior distribution needed for our parameter estimation after some required number of steps t_0. Consequently, the Markov chain $\{\boldsymbol{\theta}^{(t)}; t > t_0\}$ acts as a dependent sample from $p(\boldsymbol{\theta}|\boldsymbol{y})$. The sufficient number of iterations t_0 can be investigated through plots of the sampled values; convergence is indicated by a random walk through the sample space displaying a mix of large and small jumps. Figure 1.1(a) shows the rapid convergence of an example Markov chain, and Figure 1.1(b) shows the sampled values after the first 1000 iterations were removed. This collection of removed iterations is known as the *burn-in period*. Provided that the length of the burn-in period is larger than t_0, all remaining observations serve as a sample from the parameter's posterior density. Quantities such as the sample's mean and quantiles can be easily calculated to represent posterior beliefs about the parameter in question. This method was used to sample all relevant model parameters in this report, and summary graphics of their posterior distributions are provided in the Results chapter.

Figure 1.1 Example plots showing the convergence of the Gibbs Sampling algorithm (a) and the resulting sample of 5000 parameter values (b).



Item Response Theory

Terminology

Item response models are often applied in the educational setting to describe characteristics of exams with questions that are scored as either correct or incorrect. Naturally, multiple choice tests are the primary area of application. In this familiar context, the test questions are referred to as *items*, and the individual test-takers as *examinees*. Suppose a multiple-choice exam contains k items answered by n examinees. When examinee *i* answers item *j*, his/her response is recorded as correct ($y_{ij} = 1$) or incorrect ($y_{ij} = 0$). One purpose of the item response model is to calculate the probability that a student answers each question correctly. This probability depends on the individual student's skill or knowledge in the subject matter, called *latent ability*, and characteristics of the question itself, called *item parameters*. Our primary model utilizes two item parameters, *discrimination* and *difficulty*, in addition to the students' latent ability parameters. This is predictably referred to as the Two-Parameter Item Response Model.

First, each student is assumed to possess a *latent ability*, denoted by θ_i, to describe his underlying ability with respect to the exam's content. For example, a higher *latent ability* will typically result in a higher probability of answering a given question correctly. Secondly, measures of *discrimination* and *difficulty* are included for each exam question. *Item discrimination* represents the ability of an exam question to distinguish between examinees of varying abilities. For example, if an item has low *discrimination*, there will be only slight differences in probabilities of correct answers between students with low ability and students with high ability. For an item with high *discrimination*, students with significantly higher *latent ability* will have significantly higher probabilities of answering correctly. *Item difficulty* is a

measure of the relative difficulty of a test question. Suppose a single student answers questions with a variety of difficulties. The student will have a higher probability of answering correctly on the easier questions and a lower probability of answering correctly on the more difficult questions.

Item Response Curves

The Two-Parameter Item Response Model (IRM) is used to model probabilities of students answering questions correctly – that is, quantities restricted to the interval [0,1]. In these situations, it is common practice to utilize a link function which generates only responses within such an interval. Therefore, the probability of a correct answer is modeled as some distribution function F of the parameters, written

$$P(y = 1|\theta) = F(\alpha\theta - \beta) \quad \textbf{(1.3)}$$

where α is *item discrimination* and β is *item difficulty.* Notice that *discrimination* and *difficulty* effectively act as the model's rate and location parameters, respectively. The two most commonly used link functions in these situations are *probit* and *logit*. The *probit* link uses a standard normal cumulative density function (CDF):

$$F(t) = \Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-z^2/2}\, dz \quad \textbf{(1.4)}$$

The *logit* link uses the CDF of the logistic distribution, also known as the logistic function:

$$F(t) = \frac{1}{1+e^{-t}} \quad \textbf{(1.5)}$$

Once the link function is assigned and the item parameters determined, we can create an *item response curve* for a specific exam question as a function of the students' *latent abilities* θ_i. Figure 1.2 provides example item response curves to allow for intuitive interpretations of the item parameters, much like the IRT discussion by Johnson and Albert (1999).

Figure 1.2 Item response curves (IRC), using the *probit* link function. Curves shown are the typical IRC (a), curves of varying *item difficulty* (b), curves of varying *item discrimination* (c), and curves with extreme *discrimination* values (d).

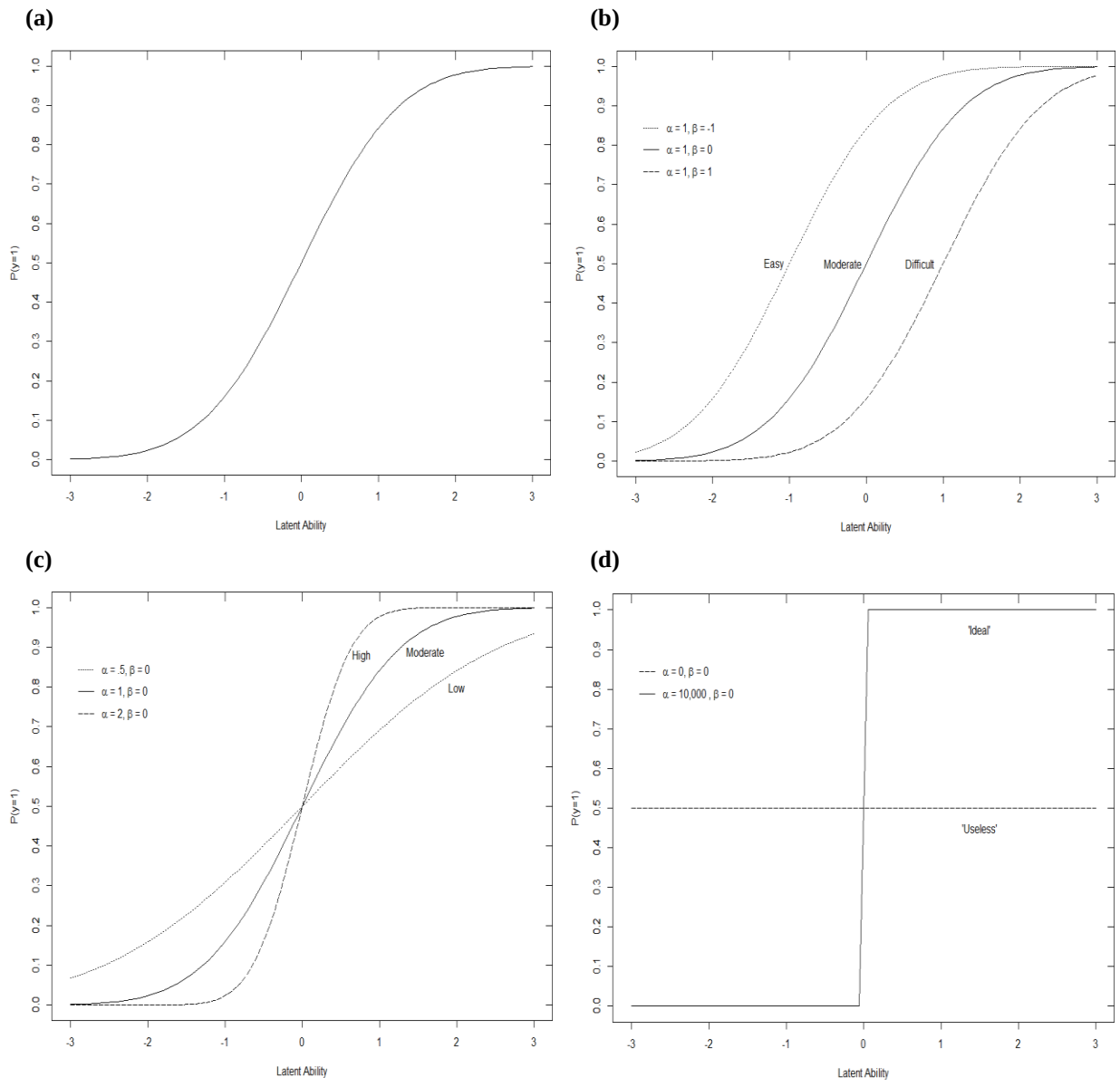


Figure 1.2(a) shows the typical item response curve with $\alpha = 1$ and $\beta = 0$. It is common to assign the prior distributions $\theta_i \sim N(0,1)$, so that, a priori, most values are within the interval (-3, 3) and the average *latent ability* is represented by $\theta = 0$. Note that the probability of answering correctly is always in the interval [0, 1], and the probability of a student with average

ability giving a correct response is equal to 0.5. Also, since the *item discrimination* is effectively a rate parameter, setting $\alpha > 0$ gives a curve in which the probability of a correct answer increases as a student's *latent ability* increases. Since the positive relationship between ability and performance is usually expected, many IRMs justifiably restrict α to this support.

In Figure 1.2(b), we can view the effect of changing the *difficulty* parameter β. As β increases, for fixed α, the item response curve shifts to the right, indicating that higher latent abilities are required to achieve the same probabilities of answering correctly. Therefore, the curve corresponding to $\beta = -1$ represents an "easy" item, the curve corresponding to $\beta = 0$ represents an item of "moderate difficulty", and the curve corresponding to $\beta = 1$ represents a "difficult" item.

Figure 1.2(c) shows the effect of changing the *discrimination* parameter α, keeping β fixed. Here, only the slope of the curve is affected as α changes from 0.5 to 2. With a low *discrimination* of 0.5, the probabilities of correct answers change slowly as *latent ability* is increased. This means that the students of varying abilities would generally be more similar in their responses. In contrast, a high *discrimination* of 2 creates a curve which alters the probabilities of correct answers very quickly. In this case, we would likely see a separation between students of high ability and students of low ability, indicating that this item "discriminates" well. The effect of *item discrimination* is further illustrated in Figure 1.2(d), which displays the curves of what could be considered "ideal" and "useless" exam questions. The "ideal" item would assure that all students of at least some given *latent ability* answer correctly, and all students with lower *ability* answer incorrectly. The curve is achieved by applying a very high *discrimination*, $\alpha = 10{,}000$, and setting *difficulty* to the required *latent ability* to answer correctly, in this case $\beta = 0$. On the other hand, a "useless" item would

provide no distinction between students of different *latent abilities*, and thus all students have the same probability of answering correctly.

An Example

Johnson and Albert (1999) walk through a typical example of item response modeling, using data from a sociological experiment. In this study, $n = 120$ students were given personality ratings by a subset of $k = 107$ of those students in the categories of likeability, aggressiveness, and shyness. The example materials can be found at http://www-math.bgsu.edu/ ~albert/ ord_book/Chapter6/, with the dataset called "ratings.dat". In the text's example, the shyness classifications act as item response data, where the 120 students are treated as *examinees*, the 107 ratings of each student as *items*, and the shyness ratings as responses. Here, each student was perceived as either "shy" or "not shy". If "shy" answers are assigned ones and "not shy" answers zeroes, then this dataset can be expressed as a 120 x 107 matrix of binary data. The relationship between the parameters and the probability of a student being perceived as shy is expressed via the Two-Parameter Probit IRM (2PP),

$$P\left(y_{ij} = 1 \middle| \theta_i\right) = \Phi\left(\alpha_j \theta_i - \beta_j\right);\ i = 1, \ldots, 120;\ j = 1, \ldots, 107 \qquad \textbf{(1.6)}$$

Therefore, using the *probit* link, we can write the probability of a response y_{ij} as

$$P\left(y_{ij} \middle| \theta_i, \alpha_j, \beta_j\right) = \Phi(\alpha_j \theta_i - \beta_j)^{y_{ij}}[1 - \Phi\left(\alpha_j \theta_i - \beta_j\right)]^{1 - y_{ij}};\ y_{ij} = 0, 1 \qquad \textbf{(1.7)}$$

Once the assumption is made that all responses given by an examinee are independent, called *conditional independence*, the probability of a student's sequence of responses $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{ik})$ is

$$P(y_i | \theta_i, \alpha, \beta) = \prod_{j=1}^{k} \Phi(\alpha_j \theta_i - \beta_j)^{y_{ij}}[1 - \Phi\left(\alpha_j \theta_i - \beta_j\right)]^{1 - y_{ij}} \qquad \textbf{(1.8)}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)$. Next, we must assume that all examinees' responses are also independent of each other, which means that the data likelihood is simply the product of the above probability:

$$L(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{i=1}^{n} \prod_{j=1}^{k} \Phi(\alpha_j \theta_i - \beta_j)^{y_{ij}} [1 - \Phi(\alpha_j \theta_i - \beta_j)]^{1-y_{ij}} \quad \textbf{(1.9)}$$

Johnson and Albert (1999) assumed a priori that the *latent abilities* $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ and the *item difficulties* $\boldsymbol{\beta}$ are drawn from Standard Normal distributions, and the *item discriminations* $\boldsymbol{\alpha}$ are drawn from *N(1,1)*. This implies that the joint posterior distribution is

$$g(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta} | \boldsymbol{y}) \propto L(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \prod_{i=1}^{n} \phi(\theta_i; 0,1) \prod_{j=1}^{k} \phi(\alpha_j; 1,1)\ \phi(\beta_j; 0,1) \quad \textbf{(1.10)}$$

where $\phi(x; \mu, \sigma^2)$ denotes a normal density with parameters μ and σ^2. Full conditionals based on (1.10) are not all easy to sample from. Thus, they also implement a latent variable structure introduced by Albert and Chib (1993) for this example, defining the unobservable quantity Z_{ij} to represent the underlying cause for examinee responses. This process simplifies the Gibbs Sampling algorithm, making it easier to sample from the desired joint posterior distribution. The latent variables are simulated from truncated normal distributions, and then the posterior distributions can be calculated using standard results from normal linear models. More details on this process are included in the Methods chapter. Defining $\boldsymbol{Z} = (Z_{11}, \ldots Z_{nk})$, the final joint posterior density of all model parameters is

$$g(\boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta} | \boldsymbol{y}) \propto \prod_{i=1}^{n} \prod_{j=1}^{k} \left[\phi(Z_{ij}; m_{ij}, 1) Ind(Z_{ij}, y_{ij})\right]$$

$$x\ \prod_{i=1}^{n} \phi(\theta_i; 0,1) \prod_{j=1}^{k} \phi(\alpha_j; 1,1)\ \phi(\beta_j; 0,1) \quad \textbf{(1.11)}$$

where $Ind(Z_{ij}, y_{ij})$ equals 1 when $\{Z_{ij} > 0, y_{ij} = 1\}$ or $\{Z_{ij} \leq 0, y_{ij} = 0\}$ and equals 0 otherwise.

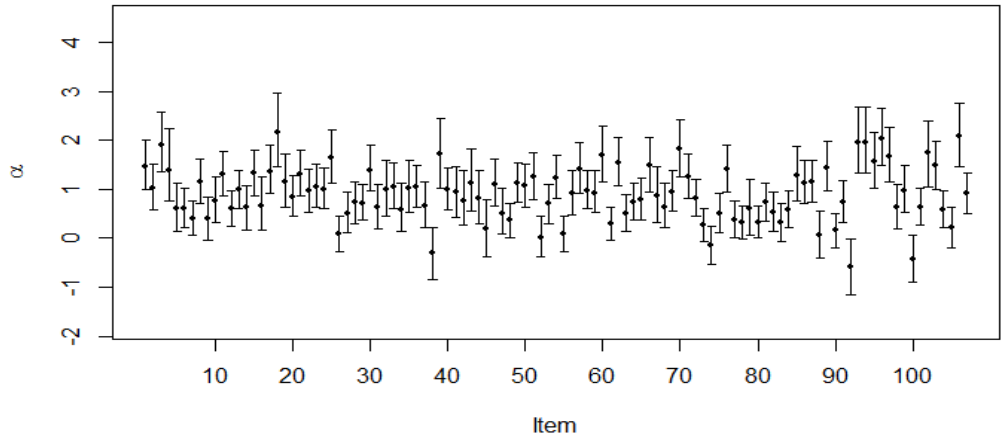
Each iteration of the Gibbs sampling algorithm for this example draws $\left\{Z_{ij}^{(t)}\right\}$ from truncated normal distributions with means m_{ij} and variance 1. The truncation occurs at 0: from the left when $y_{ij} = 1$, and from the right when $y_{ij} = 0$. Next, the latent abilities $\left\{\theta_i^{(t)}\right\}$ are simulated from normal distributions with means m_{θ_i} and variances v_{θ_i}. Finally, the item parameters $\left\{\alpha_j^{(t)}, \beta_j^{(t)}\right\}$ are drawn from a multivariate normal density with mean vector m_j and covariance matrix v_j. The means and variances for the conditional posterior distributions of the model parameters are explained further in the Methods chapter of this report. Although the choice of initial values for these parameters is usually inconsequential, Johnson and Albert (1999) also provides strategies for determining reasonable starting values, which may lead to faster convergence.

We recreated this Gibbs sampling example using the dataset posted online at *http://www-math.bgsu.edu/~albert/ord_book/*. For this reproduction, we used 1500 iterations after burn-in, rather than the text's choice of 1000, and the Gibbs sampler was programmed in R, (R Core Team, 2014). It is important to verify that the Gibbs algorithm converged during the burn-in period, a characteristic which can be visualized in the Simulation Sequence vs. Iteration plot for any parameter of interest. The text provides such a plot for the specific parameter α_{25}, where no trend is apparent across iterations, and only a moderate correlation between successive iterates is present. Johnson and Albert (1999) found the posterior standard errors to be consistent across parameters, so they felt justified in using this MCMC sample to summarize their posterior knowledge about the 120 examinees and 107 items. The posterior distributions of all $n + 2k = 334$ parameters are summarized via whisker plots in Figures 1.3(a-c).

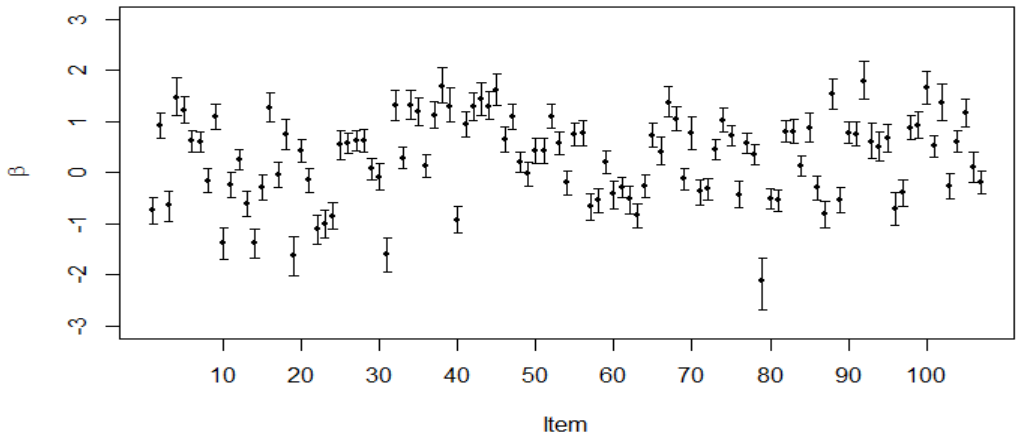
The *Shyness* example illustrates all of the concepts employed in our research: the application of Item Response Theory to a non-standard setting, the subsequent parameter interpretations, the determination of the likelihood, prior distributions, and resulting joint posterior density, the implementation of the Gibbs sampling algorithm, the rough verification of algorithm convergence, and brief summaries of the model parameters' posterior distributions.

Figure 1.3 Side-by-side whisker plots for all *discrimination* (a), *difficulty* (b), and *latent shyness* (c) parameters. The ends of the 'whiskers' represent a 90% credible interval for each posterior distribution, and the center dots signify posterior means.

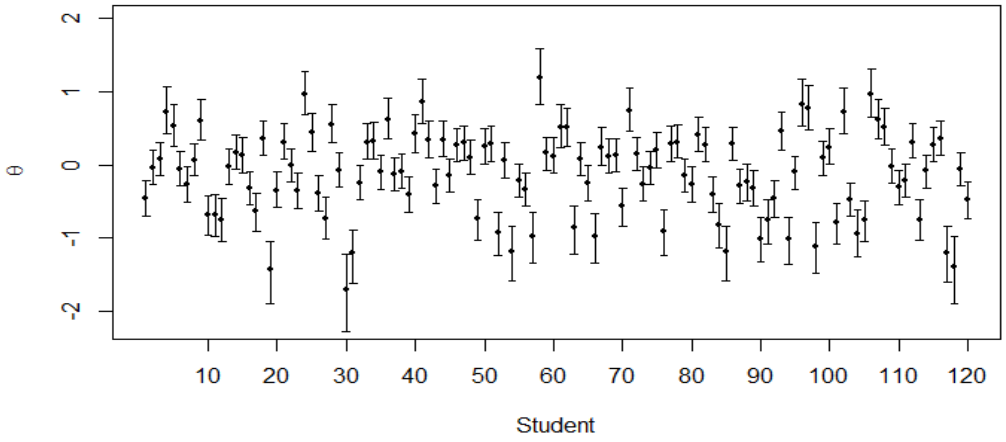
(a)



(b)



(c)



Chapter 2 - Methods

Preference Testing Data

The two primary datasets used to evaluate the IRM were taken from Wilke et al. (2006), who advocated the use of multiple replications in preference testing. The products chosen were raisin bran and cola, selected for their "ease of preparation, similar appearance within a product type and small but noticeable differences during tasting." Several self-reported consumers of each product were chosen as panelists: 305 people for raisin bran and 296 for cola. For both products, samples of two national brands were given to each panelist in random order, in order to decrease the effect of testing position. The panelists were then instructed to choose which product they preferred. After a seven-minute waiting period, the next test was conducted; four preference tests were completed for each consumer.

Preference testing data can be expressed as a collection of response patterns, the series of responses across replications for each consumer. With four replications and two possible choices, product A and product B, there are sixteen possible response patterns. To apply the Two-Parameter IRM, we formatted the data into an $n\ x\ k$ matrix of binary data, where n is the number of panelists, and k is the number of replications. The responses are denoted by $\{y_{ij};\ i = 1, \dots n, j = i, \dots, k\}$, where $y_{ij} = 1$ indicates that panelist i preferred product A on replication j, and $y_{ij} = 0$ indicates a preference for product B for the same consumer and replication. The product labels are duplicated from the original paper, in which the specific product names were not disclosed; regardless, the assignment of labels 'A' and 'B' is arbitrary for our purposes.

Hypothetical Datasets

To assist with model parameter interpretation and investigate how the IRM behaves for certain preference patterns, we generated five datasets with the same number of replications as the *Raisin Bran* and *Cola* data but designed with specific properties in mind. Rather than conducting a simulation study by setting parameter values and randomly generating data, we manually created case studies of possible response matrices. To explore the resulting parameter values when all possible response patterns are equally present, the *Balanced* dataset was created. To contrast the *Balanced* data, we created the *Extreme* dataset, in which all consumers consistently chose either one product or the other. Next, to explore the effects of each model parameter when the other is held constant, we created the *Increasing Discrimination* and *Increasing Difficulty* datasets. To achieve *Increasing Discrimination*, a collection of response patterns had to be adjusted carefully to portray low *item discrimination* in early replications and high *item discrimination* in late replications. Likewise, since posterior *item difficulty* is related to the observed proportion of consumers choosing product A, an *Increasing Difficulty* dataset will contain more preferences for A in early replications and fewer preferences for A in late replications. *Increasing Difficulty* was achieved via two patterns: *Monotone* and *Non-Monotone*. More details on the creation of these hypothetical datasets may be found in the Results chapter.

Model Specification and Gibbs Sampling

Recall that the general Two-Parameter IRM can be represented by

$$P(y_{ij} = 1 | \theta_i) = F(\alpha_j \theta_i - \beta_j),$$

where θ_i is the *latent ability* for consumer i, and α_j and β_j are the *discrimination* and *difficulty* parameters for test j, respectively. With a *probit* link function, this model becomes

$$P(y_{ij} = 1|\theta_i) = \Phi(\alpha_j\theta_i - \beta_j),$$

and a *logit* link produces

$$P(y_{ij} = 1|\theta_i) = \frac{1}{1+e^{-(\alpha_j\theta_i-\beta_j)}} \quad \textbf{(2.1)}$$

After specification of the prior distributions, we are free to collect data and calculate the joint posterior density from which we must sample values. However, the full conditional densities used by the Gibbs sampler are difficult to sample from. These issues could be overcome by incorporating a Metropolis-Hastings step, but we will instead discuss a clever alternative.

The application of the 2PP follows directly from the *Shyness* data example presented by Johnson and Albert (1999). Because of the consistent data structure, the specific methods described here are directly applicable to all seven preference testing datasets. Each dataset is an $n\ x\ k$ matrix of binary data, where $y_{ij} = 1$ indicates a preference for product A over product B for consumer i in replication j. Albert and Chib (1993) introduced a data augmentation mechanism which simplifies the Gibbs sampler for situations with such high complexity. For each observation, there exists $Z_{ij} \sim N(m_{ij}, 1)$, where $m_{ij} = \alpha_j\theta_i - \beta_j$. We then assume that

$$y_{ij} = \begin{cases} 0\,, & Z_{ij} \leq 0 \\ 1\,, & Z_{ij} > 0. \end{cases}$$

Thus,

$$P(y_{ij} = 1) = P(Z_{ij} > 0) = 1 - \Phi(-m_{ij}) = \Phi(m_{ij}), \quad \textbf{(2.2)}$$

which produces the familiar form of the 2PP shown above.

It is standard practice to assume the *latent preferences* are drawn from a Standard Normal distribution, so that an "average" ability is zero. More than just expressing our prior beliefs about θ_i, this should also prevent identifiability issues, according to Johnson and Albert (1999).

We still encountered minor problems in this regard, which are discussed in the Computation chapter. We also assume that an "average" difficulty will be zero, for ease of interpretation. However, the prior distributions for α and β can reasonably be changed for varying purposes. It is intuitive to restrict α to positive numbers, which implies that an increase in underlying preference for product A leads to an increase in the probability of selecting A. To allow for the most freedom, we specified $\alpha_j \sim N(\mu_\alpha, s_\alpha^2)$, where $\mu_\alpha > 0$ assumes a priori that the *discrimination* is positive without restricting the posterior values to be so. In addition, we specified $\beta_j \sim N(0, s_\beta^2)$ to obtain a distribution symmetric about zero but flexible with respect to prior variance.

Based on these prior distributions, the joint posterior density of the parameters conditional on the data is

$$g(\boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta} | \boldsymbol{y}) \propto \prod_{i=1}^{n} \prod_{j=1}^{k} \left[\phi(Z_{ij}; m_{ij}, 1) Ind(Z_{ij}, y_{ij})\right]$$

$$x \ \prod_{i=1}^{n} \phi(\theta_i; 0,1) \prod_{j=1}^{k} \phi(\alpha_j; \mu_\alpha, s_\alpha^2) \ \phi(\beta_j; 0, s_\beta^2) \quad \textbf{(2.3)}$$

To employ the Gibbs Sampling algorithm, each parameter is then iteratively sampled from its full conditional posterior density:

- $g_{\boldsymbol{Z}}(\boldsymbol{Z}|\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{y})$
- $g_{\boldsymbol{\theta}}(\boldsymbol{\theta}|\boldsymbol{Z}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{y})$
- $g_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{\alpha}, \boldsymbol{\beta}|\boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{y})$

Therefore, at iteration *t*, the parameters are drawn as follows:

1) $\left\{Z_{ij}^{(t)}\right\} \sim \begin{cases} TN(m_{ij}, 1; 0, \infty)\,, & y_{ij} = 1 \\ TN(m_{ij}, 1; -\infty, 0)\,, & y_{ij} = 0 \end{cases}$ **(2.4)**

where $TN(\mu, \sigma^2; c, d)$ denotes a Normal distribution with mean μ and variance σ^2 truncated over the interval (c, d), and $m_{ij} = \alpha_j^{(t-1)}\theta_i^{(t-1)} - \beta_j^{(t-1)}$.

2) $\left\{\theta_i^{(t)}\right\} \sim N\left(m_{\theta_i}, v_{\theta_i}\right)$ **(2.5)**

where $m_{\theta_i} = \frac{\sum_{j=1}^{k} \alpha_j^{(t-1)}(Z_{ij}^{(t)} + \beta_j^{(t-1)})}{\sum_{j=1}^{k} \alpha_j^{2(t-1)} + 1}$

and $v_{\theta_i} = \frac{1}{\sum_{j=1}^{k} \alpha_j^{2(t-1)} + 1}$

3) $\left\{\alpha_j^{(t)}, \beta_j^{(t)}\right\} \sim N_2\left(m_j, v_j\right)$ **(2.6)**

where $m_j = [X'X + \Sigma_0^{-1}]^{-1}\left[X'Z_j^{(t)} + \Sigma_0^{-1}\begin{pmatrix}\mu_\alpha \\ 0\end{pmatrix}\right]$ is the 2-dimensional mean vector,

$v_j = [X'X + \Sigma_0^{-1}]^{-1}$ is the *2 x 2* covariance matrix,

$\Sigma_0 = \begin{bmatrix} s_\alpha^2 & 0 \\ 0 & s_\beta^2 \end{bmatrix}$ and X is the design matrix with columns $(\theta_i^t, -1)$.

This algorithm was first programmed in R, to verify the *Shyness* data results from Johnson and Albert (1999). The R code for our 2PP IRM is provided in the Appendix. Similarly to the *Shyness* example, the parameters' starting values were set to their respective prior means. Where the authors used 1000 iterations and no burn-in period for their MCMC sample, we iterated the algorithm 2000 times and removed the first 500. Although this was an adequate method for working through the text's example, the R code is too computationally costly for

repeated use with different specifications. To expedite computation, we utilized OpenBUGS software to run the algorithm for all remaining portions of this research. OpenBUGS was introduced by Thomas et al. (2006) as an updated version of WinBUGS, the popular Gibbs Sampling software developed by Lunn et al. (2000). Our model specification in OpenBUGS does not use the data augmentation approach given by Albert and Chib (1993), but rather uses the model first described in this chapter. This required only slight modifications to the BUGS code given by Curtis (2010) for the Two-Parameter Logistic model (2PL).

The models were fit by OpenBUGS software through the R interface, using the R package 'BRugs', introduced by Ligges (2013). 'BRugs' gives access to convergence plots and the entire list of sampled parameter values through the functions 'samplesHistory' and 'samplesSample', respectively, allowing further flexibility for output storage and plot generation. In particular, the R function 'BRugsFit' requires the input of a model file and dataset in OpenBUGS syntax, and it outputs posterior information for the parameters of interest, including the mean and the 95% credible interval. This function allows the user to specify initial values, parameter(s) to follow, the number of desired MCMC chains, the number of iterations, and the duration of the burn-in period, among other settings. For all seven preference testing datasets, we specified one MCMC chain of 6000 iterations, with the first 1000 constituting the burn-in period, and the initial values were randomly generated by OpenBUGS. All computations were done with random number seeds for future reproducibility. More information regarding the computation of these values, convergence, and potential difficulties can be found in the Discussion chapter. The parameters saved were $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\theta}$, and the probability of preferring product A for each replication, denoted by $\boldsymbol{p} = (p_1, \ldots, p_4)$. The probability of preferring product A on replication j is calculated by

$$p_j = \Phi\left(\frac{-\beta_j}{\sqrt{1+\alpha_j^2}}\right) \quad \textbf{(2.7)}$$

p_j may be used to indicate which of two products was preferred across all subjects, hence the formula's absence of any *latent ability* parameters, and it can be thought of as an alternate measure of *item difficulty*. This probability can be expected to adhere closely to the observed proportions of consumers preferring product A at each replication, depending on the choice of prior distributions.

Chapter 3 - Results

Overview

We applied the 2PP to the *Raisin Bran* and *Cola* datasets described and used by Wilke et al. (2006). These datasets originally illustrated the importance of replication in sensory analysis tests due to panelists' inconsistent preferences, and they now serve as useful real-world demonstrations of the Two-Parameter IRM. To verify this model's capabilities in controlled situations and assist in parameter interpretation, we constructed five replicated preference test datasets with specific properties. Each of these datasets features four replications, to imitate the real-world data described above, ranging from 60 to 135 consumers. These hypothetical datasets will be referred to as *Balanced*, *Extreme*, *Increasing Discrimination*, and *Increasing Difficulty (Monotone* and *Non-Monotone)*. The same prior distributions were used for all seven datasets, as described in the *Shyness* data example: $\theta_i \sim N(0,1)$, $\alpha_j \sim N(1,1)$, and $\beta_j \sim N(0,1)$.

The plots included in this section are used to summarize the posterior distributions of several quantities. The *latent preference* parameter θ_i describes each consumer's true preference for one of two products, with positive values representing preferences for product A. Included in this section are side-by-side whisker plots of the posterior distributions of each *latent preference* parameter, where the ends of the whiskers represent a 95% credible interval for θ_i. The *discrimination* and *difficulty* parameters are quantities that describe characteristics about each preference test replication. First, the *discrimination* parameter, denoted by α_j, determines the relative distinction between consumers of varying latent preferences for product A at a given replication. For example, high *discrimination* indicates that the consumers' recorded preferences on a given replication align with a relatively strong latent preference for that product, and low *discrimination* indicates virtually no relationship between true *latent preferences* and the

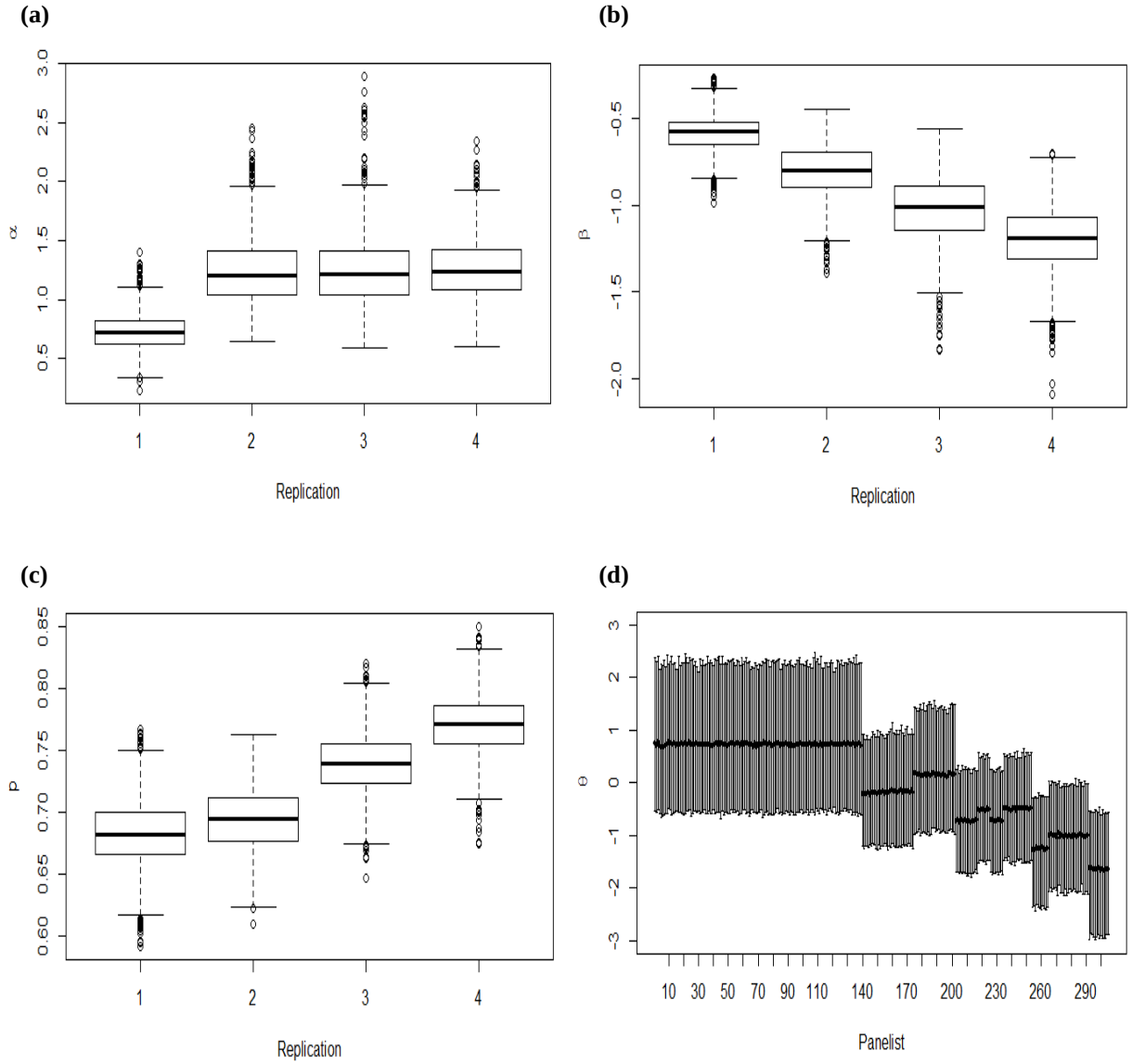
preferences observed in the data. Secondly, the *difficulty* parameter β_j explains the relative probability of selecting product A on a given replication for consumers with no preference for either product. A "difficult" replication would somehow invoke a lower probability of choosing A than an "easy" replication, for any given *latent preference* and *discrimination*. Naturally, all of these characteristics will change with different product and consumers. Finally, the probability of preferring product A, denoted by p_j, was calculated as described in the Methods section. This probability can be thought of as an alternate measure of *item difficulty*. Posterior summaries for the probability of preferring A were included for each dataset, but we observe that the mean of each of these distributions is approximately equal to the observed proportion of A selections for every dataset.

Raisin Bran Data

Wilke et al. (2006) utilized raisin bran preference tests to argue the importance of replication in sensory analysis studies. According to their research, consumers may not consistently record the same preferred product when the products in question are not highly discriminable and/or when the consumers' preferences are not particularly strong. The *Raisin Bran* dataset consists of 305 self-reported consumers of the popular cereal, with each of four replications forcing a reported preference between two national brands.

The posterior summary plots show interesting trends in the *Raisin Bran* test example. First, in Figure 3.1(a), the posterior distributions of the *discrimination* parameter for replications 2, 3, and 4 are very similar, but replication 1 shows a unique distribution. This means that responses were more similar between panelists of high and low preferences in the first test, whereas the next three tests did a better job of discriminating between panelists of different

Figure 3.1 Summary plots from the *Raisin Bran* data for the posterior distributions of *item discrimination* (a), *item difficulty* (b), *probability of preferring A* (c), and *latent preference* (d).



underlying preferences. In addition, we also observe decreasing *difficulty* parameters over time in Figure 3.1(b), due to the fact that more panelists chose product A in each successive preference test. Note that the distributions in Figure 3.1(c) are consistently greater than 0.5, indicating overall preference for product A at each replication. *Latent preferences* are associated

with the number of times a consumer chose product A across the four replications and the order in which they did so. Therefore, the small number of possible response patterns causes the side-by-side whisker plots in Figure 3.1(d) to segregate into ten distinct groups. The credible intervals for all but the last 52 consumers contain zero, indicating that their *latent preferences* for product A are not significantly different from zero. On the contrary, it appears that the last 52 respondents preferred product B to a significant degree.

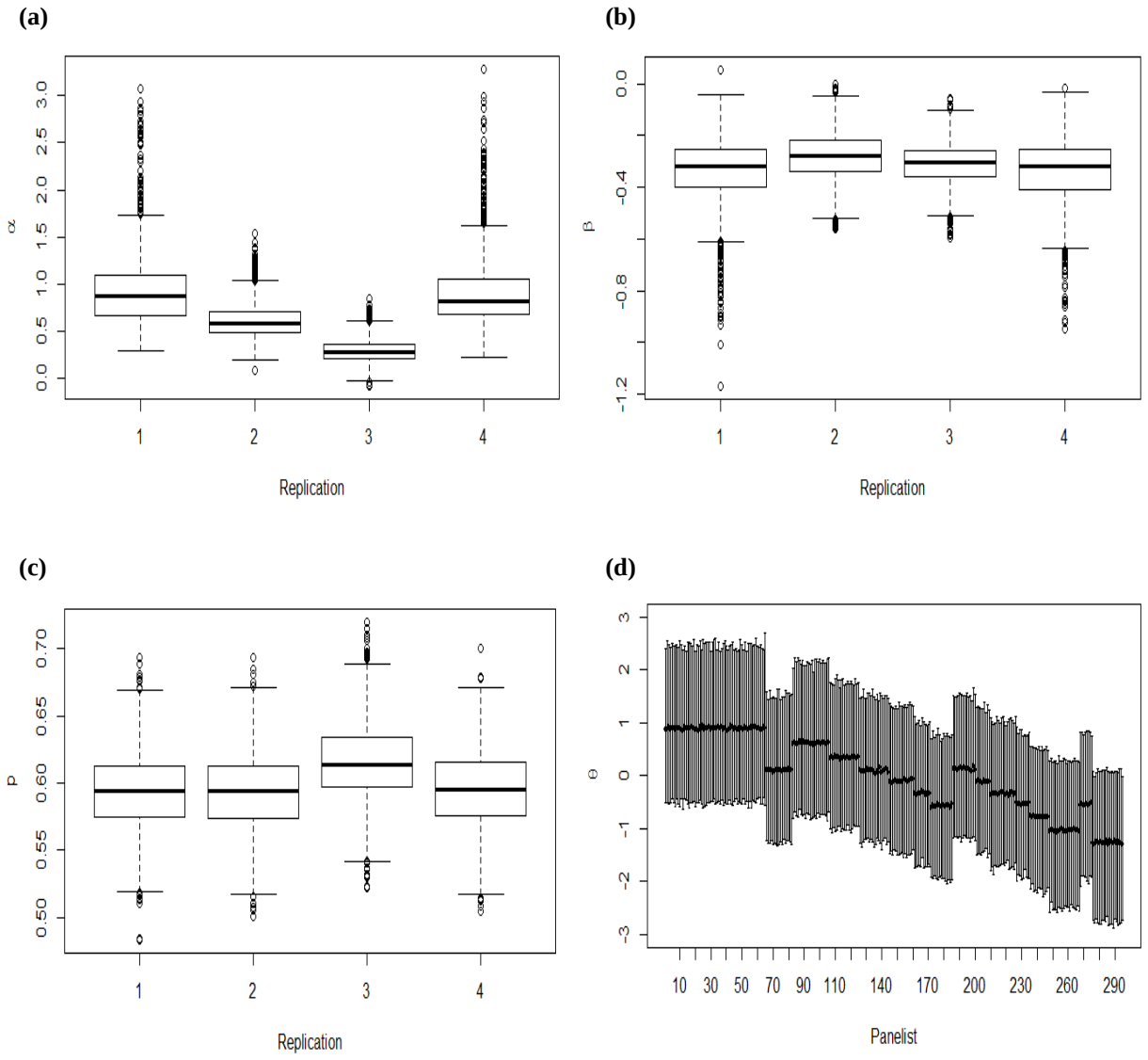
Cola Data

The *Cola* tests included 296 self-reported acceptors of the product, again requiring a choice between two national brands for each of four replications. Wilke et al. (2006) noted that nearly 71% of consumers changed their preference at least once throughout the four replications, evidence that replicated preference tests may be needed when differences between the products are not discernible, relative to personal preferences. The *Cola* example is a second test of the Two-Parameter IRM's functionality when the data is based on real humans' recorded preferences.

Much like the *Raisin Bran* dataset, we can visually separate groups of panelists' total A selection counts from their *latent preference* whisker plots in Figure 3.2(d). The posterior distributions for the *difficulty* parameters β_j in Figure 3.2(b) almost entirely consist of negative values, indicating that there was some degree of preference for product A in all four tests. In this example, the *discrimination* posterior distributions in Figure 3.2(a) decrease over the first three replications but take a positive turn in the final test. Returning to the interpretation of *item discrimination*, this means that Replications 1 and 4 more clearly distinguish between consumers of varying latent preferences, whereas consumers tend to respond contrary to their true

preferences more often in the middle replications. As in the *Raisin Bran* data, the distributions of the probability of selecting product A at each replication in Figure 3.2(c) show clear preference for product A.

Figure 3.2 Summary plots from the *Cola* data for the posterior distributions of *item discrimination* (a), *item difficulty* (b), *probability of preferring A* (c), and *latent preference* (d).



Balanced Data

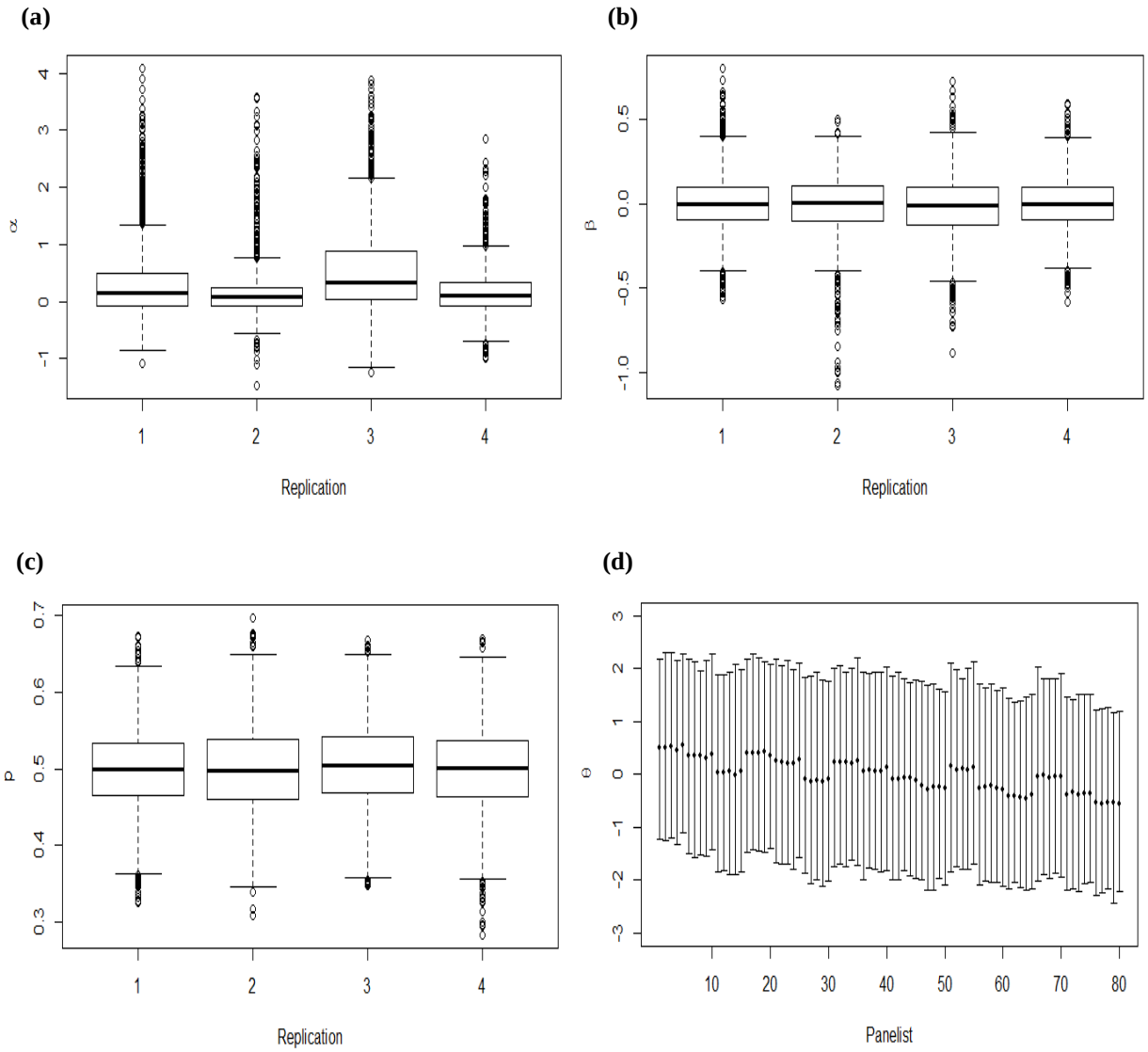
With four replications, sixteen possible response patterns exist for any given panelist. The *Balanced* dataset consists of five panelists for each of these sixteen patterns, outlined in Table 3.1. This balanced design ensures that a 50% preference for product A is observed in each preference test, and those preferences are spread out among the different panelists; this means that few panelists hold extreme prejudice for one product or the other, whereas the majority of subjects do not show great preference. This could be due to either a lack of preference for a product or an inability to distinguish the two products.

The consistent location of the posterior distributions of α and β in Figure 3.3(a-b) reflects the intended balance of this dataset. Because the distribution of preferences for A has no

Table 3.1 The collection of possible response patterns in data with four replications. 'Count' and 'Total' are specific to the responses observed in the *Balanced* dataset.

	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	1	4	5
	1	1	1	0	3	5
	1	1	0	1	3	5
	1	1	0	0	2	5
	1	0	1	1	3	5
	1	0	1	0	2	5
	1	0	0	1	2	5
	1	0	0	0	1	5
	0	1	1	1	3	5
	0	1	1	0	2	5
	0	1	0	1	2	5
	0	1	0	0	1	5
	0	0	1	1	2	5
	0	0	1	0	1	5
	0	0	0	1	1	5
	0	0	0	0	0	5
Total	40	40	40	40		

Figure 3.3 Summary plots from the *Balanced* data for the posterior distributions of *item discrimination* (a), *item difficulty* (b), *probability of preferring A* (c), and *latent preference* (d).



discernible pattern across the range of *latent preferences*, the four tests do not discriminate well between those who prefer A and those who prefer B. This results in low posterior means for the four α parameters. In addition, with half of the consumers preferring A on each occasion, the mean *difficulty* at each time point is approximately zero. The posterior distributions of the *latent preferences* θ_i in Figure 3.3(d) also reflect a secondary intention of this example – to vary latent

preferences gradually while *discrimination* and *difficulty* remain nearly constant. Although ten consumers in the *Balanced* dataset responded consistently across replications, the 95% credible intervals for their respective *latent preferences* contain zero, indicating a lack of preference. This results from a relatively small sample size and a wide variety of observed response patterns.

Extreme Data

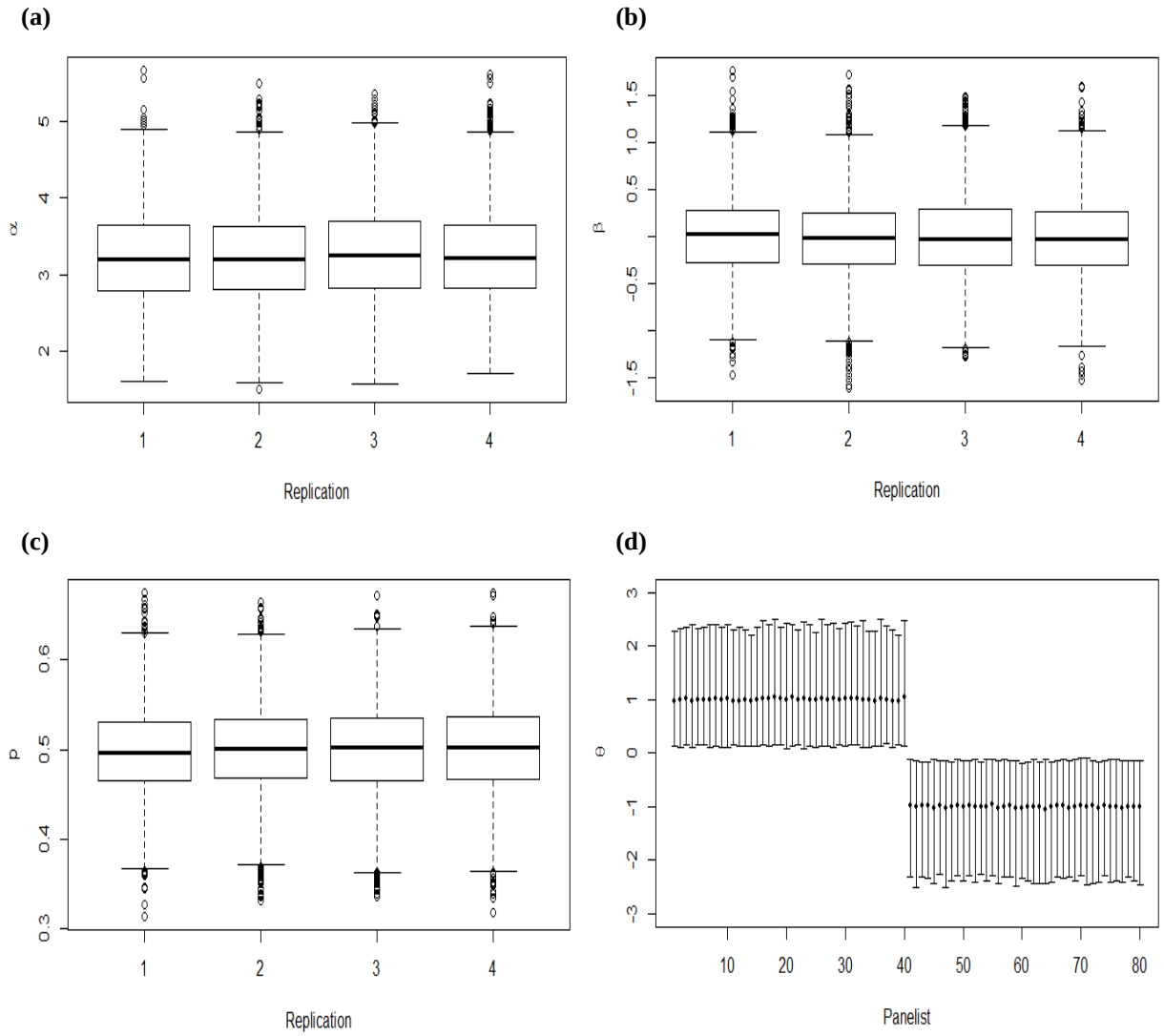
To illustrate the result of very strong preferences, we generated a dataset in which all panelists exclusively preferred either product A or product B. Like the *Balanced* example, the *Extreme* dataset also uses 80 panelists, with half choosing product A and half choosing product B in every test. This means that the observed proportion of panelists preferring product A is 0.50 at each replication; however, all respondents display a consistent preference for one product or the other. These response patterns are shown in Table 3.2.

Table 3.2 The collection of response patterns present in the *Extreme* dataset.

	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	1	4	40
	0	0	0	0	0	40
Total	40	40	40	40		

Figure 3.4(a) shows the substantial increase in the posterior means for α compared to the *Balanced* data, and the clear distinction between consumers' preferences is displayed in Figure 3.4(d). However, note that the *difficulty* distributions in Figure 3.4(b) are very similar to the previous dataset. The *Balanced* and *Extreme* examples demonstrate how wildly α and θ can vary without affecting β. That is to say, the posterior distributions of the β_j for the two datasets

Figure 3.4 Summary plots from the *Extreme* data for the posterior distributions of *item discrimination* (a), *item difficulty* (b), *probability of preferring A* (c), and *latent preference* (d).



were nearly identical, while the posterior distributions for α_j and θ_i differed substantially. In the *Extreme* case, every panelist responded consistently over time, and the model responds by assigning appropriately extreme θ_i values. In addition, because every panelist with high latent preference for A recorded such preference for every replication and vice versa, every replication is given a high expected posterior *discrimination* value. However, despite these glaring

differences from the *Balanced* example, the observed proportion of consumers preferring A remains the same for each replication, producing expected *difficulty* parameters of approximately zero.

Increasing Discrimination Data

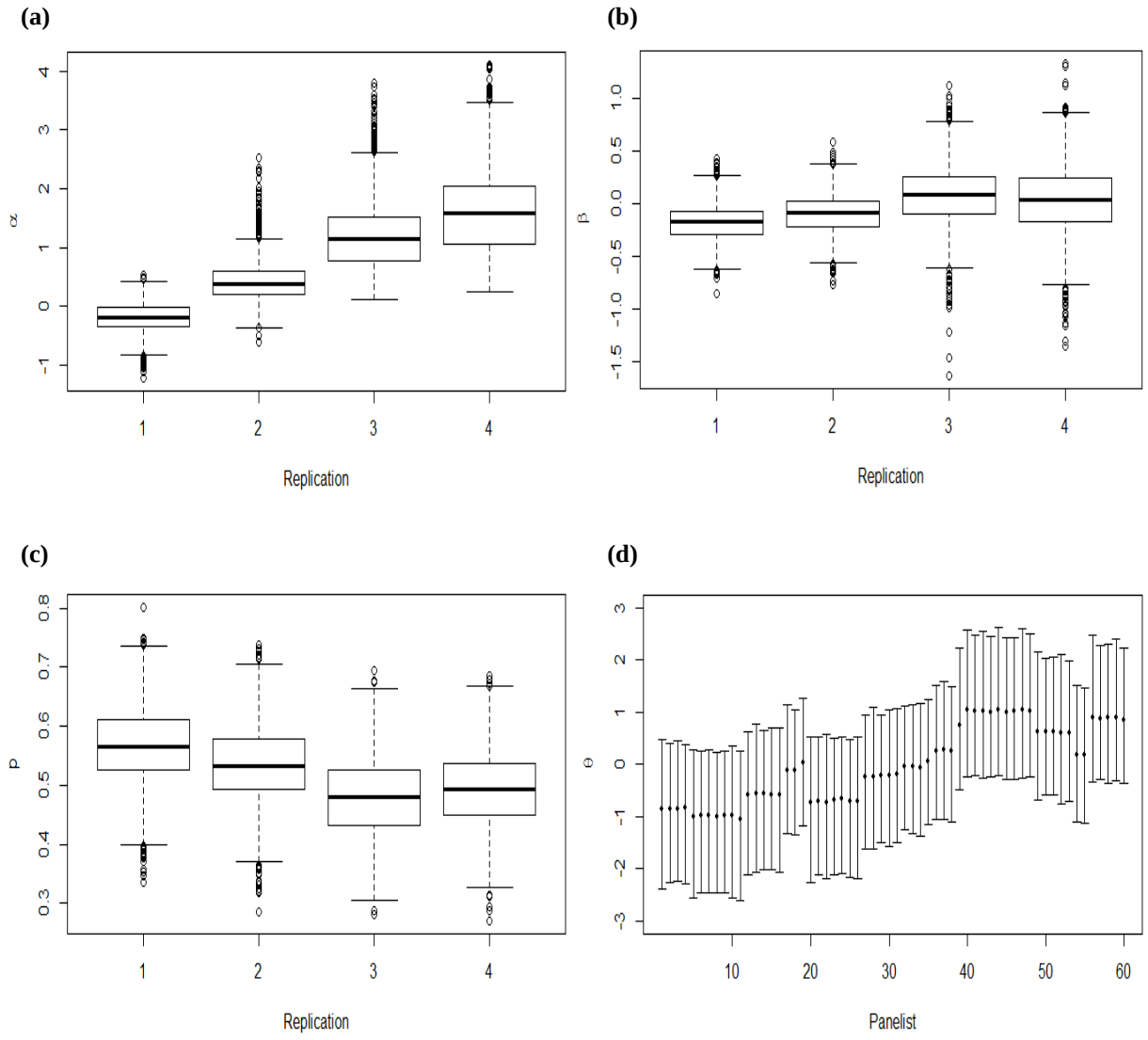
Recall the purpose of the *discrimination* parameter: α is the value that determines a replication's ability to distinguish between consumers of different latent preferences for product A. In order for a model to result in increasing *discrimination* values across replications, the dataset would require few response differences between panelists of varying latent abilities in early replications, i.e., low *discrimination*, and large differences in later replications, i.e., high *discrimination*. This will cause a mix of observed preferences for A and B throughout the range of latent preferences at early replications, whereas, in the later replications, the recorded A selections are generally associated with consumers of high latent preference for A and not with those with high latent preference for B. To generate such a dataset, each consumer's total number of preferences for product A was thought of as a crude representation of his *latent preference*. After 60 consumer response patterns were somewhat arbitrarily generated, the individual responses were adjusted to reflect the characteristics described above. For example, the pattern of 1's and 0's in the first replication indicated little association between the consumer's crude *"latent preference"* and his observed product preference, i.e., low *discrimination*. In contrast, the observed preferences for A in the last replication were almost exclusively constrained to consumers with three or four total A selections, i.e., high *discrimination*. We attempted to create a gradual transition from low to high *discrimination* across the four replications. The full dataset, with a total of 60 response patterns, can be found in Table 3.3.

As displayed in Figures 3.5(a) and 3.5(b), this example achieved the desired effect: a gradual increase in *item discrimination* over the four replications with very little change in *item difficulty*. While very little difference exists between consumers of high and low *latent preference* in the first replication, the distinction grows across time. By the fourth replication, the *item discrimination* the total number of recorded preferences for A. (Remember that these totals are crude representations of the consumers' *latent preferences*.) For this reason, the posterior means of θ_i vary across the interval (-1, 1), as seen in Figure 3.5(d).

Table 3.3 The collection of possible response patterns and their respective counts for the *Increasing Discrimination* dataset.

	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	1	4	5
	1	1	0	1	3	2
	1	1	0	0	2	7
	1	0	1	1	3	5
	1	0	1	0	2	5
	1	0	0	1	2	3
	1	0	0	0	1	7
	0	1	1	1	3	9
	0	1	1	0	2	1
	0	1	0	1	2	3
	0	1	0	0	1	5
	0	0	1	1	2	1
	0	0	1	0	1	2
	0	0	0	1	1	1
	0	0	0	0	0	4
Total	34	32	28	30		

Figure 3.5 Summary plots from the *Increasing Discrimination* data for the posterior distributions of item discrimination (a), item difficulty (b), probability of preferring A (c), and latent preference (d).



Increasing Difficulty Data

Recall that the *difficulty* of an item in preference testing context determines the relative probability of preferring product A for the same consumer and a constant *item discrimination*. A "difficult" replication would often require a higher preference for product A in order to select A in that test, whereas a less "difficult" test would often require less preference for A in order to

choose A as the preferred product. Two datasets were created to illustrate increasing *difficulty* across replications, each with 135 total subjects. Since the *difficulty* parameter is closely related to the observed proportion of product A preference, both datasets were generated with a decreasing observed proportion of consumers choosing product A in successive replications. Specifically, the observed proportions of consumers choosing A in the four replications were $(p_1, p_2, p_3, p_4) = (4/5, 3/5, 2/5, 1/5)$. This was accomplished by first creating 81 response patterns where all panelists chose A in the first replication, two-thirds chose A in the second, one-third chose A in the third, and none chose A in the fourth. Additional panelists with consistent preferences were then added, 27 preferring A at every replication and 27 preferring B at every replication. This was to avoid computational issues related to observing proportions of one and zero in replications 1 and 4, respectively. While these observed proportions were identical for both *Increasing Difficulty* datasets, the specific response patterns were not.

Monotone Method

In missing data analysis, the term *monotone* refers to a missing data pattern in which all missing values are contiguous, ie., if value v_j is missing, then all values $\{v_k;\ k > j\}$ are also missing. Here we apply this term to the creation of the *Increasing Difficulty* dataset. For the *Monotone* approach, all recorded preferences for product B at a given replication are contiguous. That is, if observation $y_{ij} = 0$, then all observations $\{y_{lj}; l > i\} = 0$. This resulted in five distinct response patterns, displayed in Table 3.4.

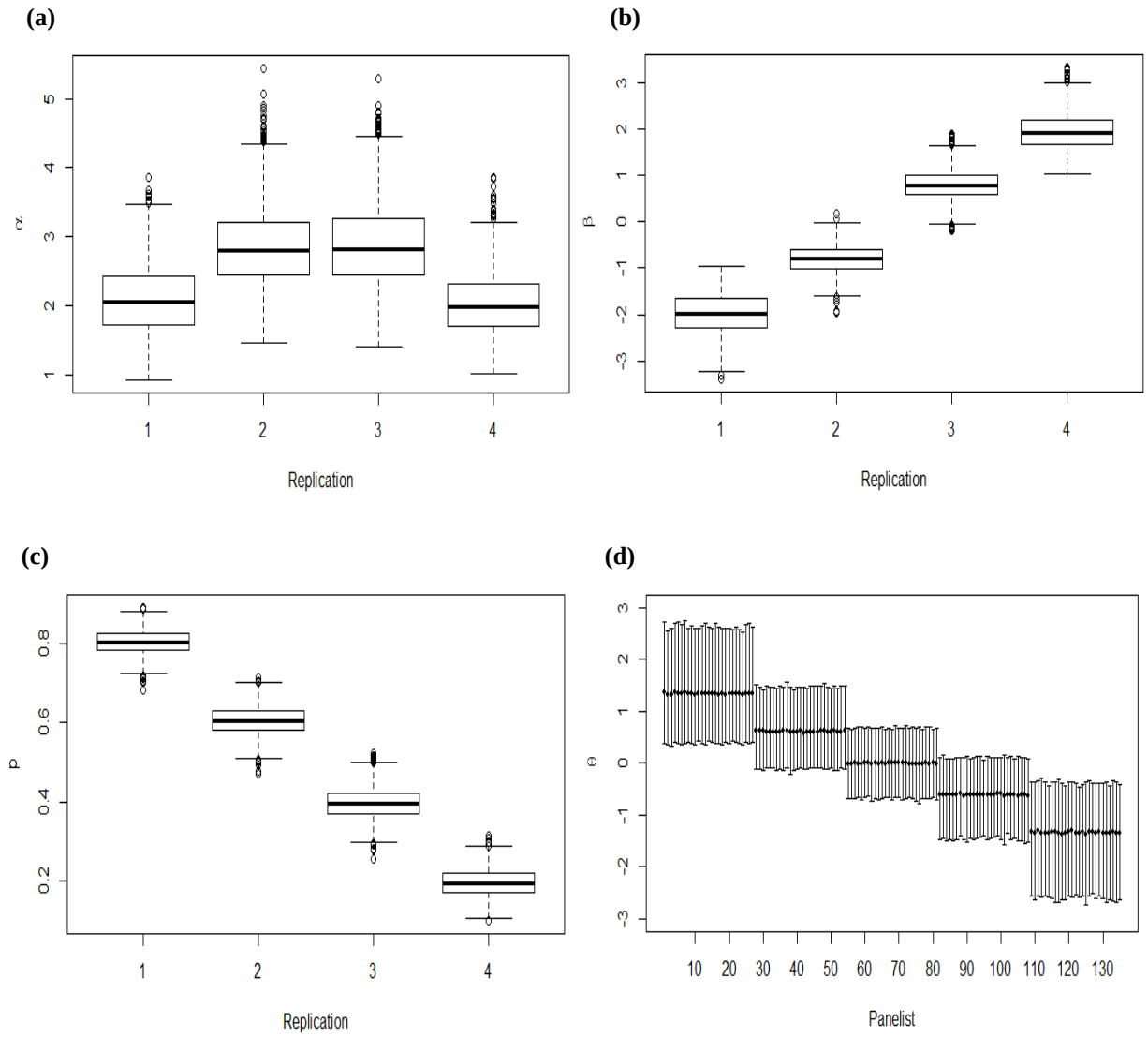
Refer to Figure 3.6 for summaries of the *Monotone Increasing Difficulty* example. This dataset produces β_j posterior distributions that predictably increase across the four replications, because the observed preferences for product A decrease in a similar pattern. It is important to remember that the number of times a consumer prefers A is directly related to his subsequent

Table 3.4 The five possible response patterns using the *Monotone* approach to create an *Increasing Difficulty* dataset.

	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	1	4	27
	1	1	1	0	3	27
	1	1	0	0	2	27
	1	0	0	0	1	27
	0	0	0	0	0	27
Total	108	81	54	27		

latent preference θ_i. This helps explain several things. First, there are five possibilities for a panelist's total number of recorded preferences for A. Therefore, we see five groupings of *latent preference* posterior distributions, with high latent preferences for A corresponding to four observed A selections and low latent preferences for A corresponding to zero observed A selections. Also, replications 1 and 4 have lower *discrimination* values than replications 2 and 3, yet they are still quite large values. High *discrimination* in the first replication is most likely because, given that a panelist chose product B, then he continued to choose product B for every test. Likewise, if a panelist chose product A in the final replication, then he chose product A for every replication. This establishes a relationship between observed preference and *latent preference*, which roughly translates to high *discrimination*. However, given that a consumer preferred A in replication 1, that consumer's total number of A selections could be 1, 2, 3, or 4. An observed preference for A in replication 1 could be associated with a relatively low *latent preference* or an extremely high *latent preference.* This fact limits replications 1 and 4 from having higher *discrimination* values, which appear in replications 2 and 3.

Figure 3.6 Summary plots from the *Monotone Increasing Difficulty* data for the posterior distributions of item discrimination (a), item difficulty (b), probability of preferring A (c), and latent preference (d).



Non-Monotone Method

In the *Non-Monotone* approach, the intent was still to observe a decreasing number of preferences for product A through the four replications. However, the locations of recorded preferences for product A in replications 2 and 3 were randomized before adding the 54 panelists

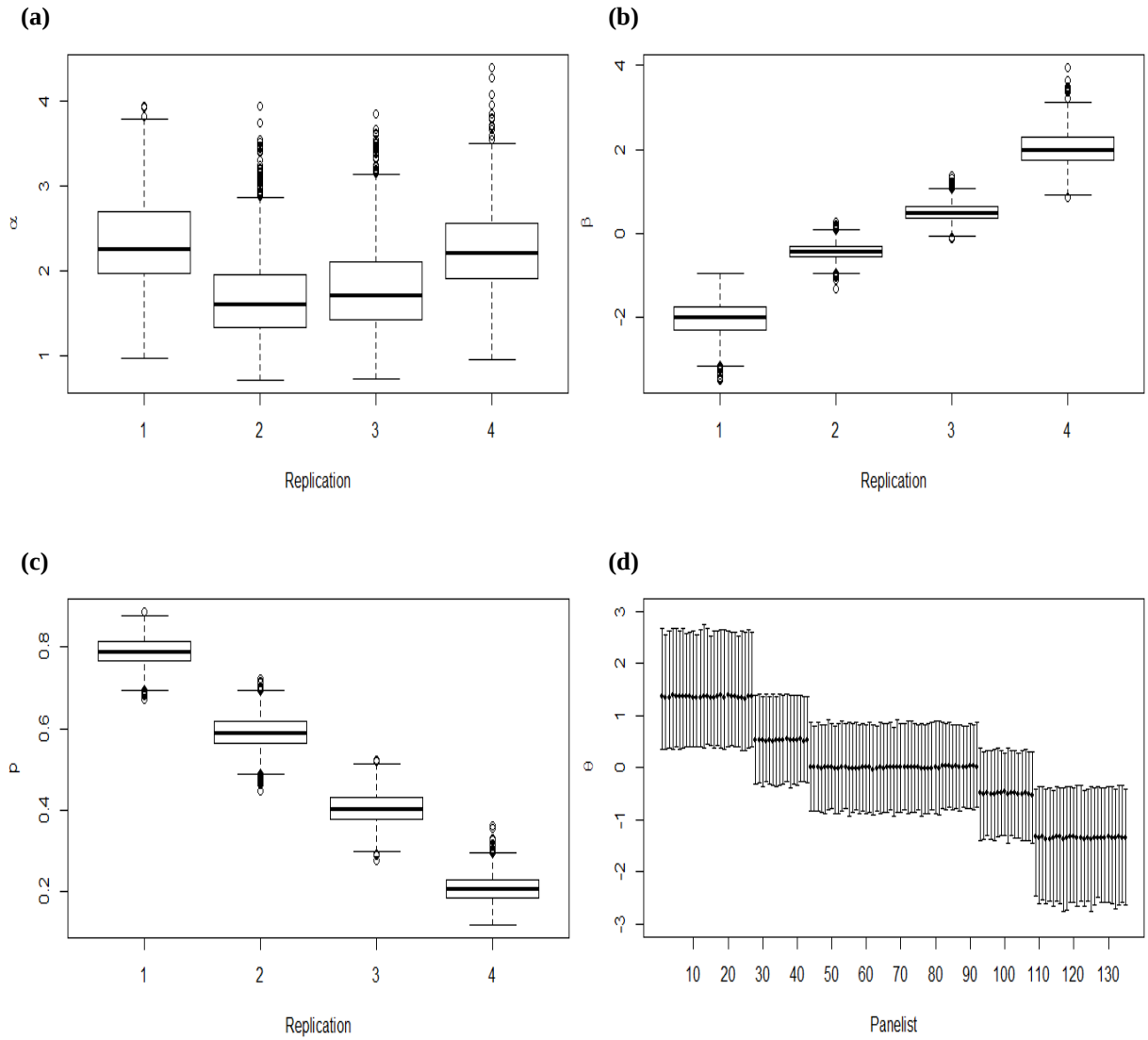
with consistent preferences. This resulted in a dataset with six distinct response patterns, rather than five. The added complexity represents the new ability for a panelist to prefer product A at non-consecutive replications, which could eliminate the monotone property of the previous dataset. After random placement of these preferences, we arrived at the dataset outlined in Table 3.5.

Table 3.5 The six possible response patterns using the *Non-Monotone* approach to create an *Increasing Difficulty* dataset.

	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	1	4	27
	1	1	1	0	3	16
	1	1	0	0	2	38
	1	0	1	0	2	11
	1	0	0	0	1	16
	0	0	0	0	0	27
Total	108	81	54	27		

The small difference in the construction of an *Increasing Difficulty* dataset leads to very different *discrimination* parameter values. While the increasing posterior distributions of the *difficulty* parameter β_j remain in Figure 3.7(b), we now observe in Figure 3.7(a) that the posterior mean *discrimination* values for replications 2 and 3 are significantly reduced. This is due to the added response pattern complexity, since a preference for product A in replication 2 or 3 no longer directly translates to a high *latent preference.* Furthermore, the posterior means of the *latent preference* θ_i may still be segregated into five groups, displayed in Figure 3.7(d), but their arrangement has been changed by the new dataset structure. While there exists an equal count of each possible number of A selections in the *Monotone* setup (27 each), we see different counts in the *Non-Monotone* setup, (27, 16, 49, 16, and 27). It is especially interesting

Figure 3.7 Summary plots from the *Non-Monotone Increasing Difficulty* data for the posterior distributions of item discrimination (a), item difficulty (b), probability of preferring A (c), and latent preference (d).



to note that the expected *latent preference* for consumers with response pattern (1,1,0,0) is essentially equal to that of consumers who responded (1,0,1,0). Because *difficulty* increases across replications, we might expect to see higher *latent preferences* assigned to consumers who preferred product A in replication 3 than to those who did so in replication 2. However, these values naturally occur because an "easy" replication translates to higher probabilities of choosing

product A. Consumers who responded (1,1,0,0) chose A in the "easier" replication 2 and chose B in the more "difficulty" replication 3, which should be expected. Although consumers who responded (1,0,1,0) indeed chose A in the more "difficult" replication 3, they also chose B in the "easier" replication 2. These two responses counteract each other, resulting in *latent preferences* similar to consumers who gave a (1,1,0,0) response pattern.

Model Comparison

While the parameter estimation for our seven preference test datasets assists with interpretation and evaluates the model's sensitivity to changing data, it says nothing of the effects of altering prior distributions, link functions, or parameterization. To investigate such effects, we selected several model variations to compare over the *Raisin Bran* and *Cola* datasets. In addition to the original two-parameter *probit* model with prior distributions $\theta_i \sim N(0,1)$, $\alpha_j \sim N(1,1)$, and $\beta_j \sim N(0,1)$, we considered sixteen models that changed the parameters of the normal prior distributions of either α or β. We then explored the possibility of restricting $\alpha > 0$ through our prior beliefs; this included three Gamma distributions of mean 1 and differing variances, as well as the original Normal prior, truncated from the left at zero.

Next, models with different parameterizations were added. The model with one *item discrimination* parameter, shared between all replications, is expressed as follows:

$$P(y_{ij} = 1|\theta_i) = F(\alpha\theta_i - \beta_j) \quad \textbf{(3.1)}$$

This model allows the "difficulty" of choosing product A to vary over time, but the replications' ability to distinguish between consumers of different *latent preferences* is constant. In contrast, we may allow *discrimination* to vary over time while keeping *difficulty* constant. The model with one *item difficulty* parameter, shared between all replications, is shown below:

$$P(y_{ij} = 1|\theta_i) = F(\alpha_j\theta_i - \beta) \quad \textbf{(3.2)}$$

The final model change results in the One-Parameter IRM, where either α or β is not used:

$$P(y_{ij} = 1|\theta_i) = F(\theta_i - \beta_j) \quad \textbf{(3.3)}$$

$$P(y_{ij} = 1|\theta_i) = F(\alpha_j\theta_i) \quad \textbf{(3.4)}$$

In multiple choice testing context, the model shown in Equation 3.3 describes a situation where only the relative difficulty of the exam questions are relevant, and each question discriminates between students at an "average" rate. On the other hand, Equation 3.4 allows different *discrimination* values between questions but only an average difficulty. The latter situation seems better fit for replicated preference testing, where the *difficulty* of each replication may be constant, but the model comparison results will shed light on these hypotheses. Two prior specifications were used for the General One-Parameter models in Equations 3.1 and 3.2, and one prior specification each was used for the One-Parameter models in Equations 3.3 and 3.4. Finally, all 27 models were applied using both the *probit* and *logit* links, resulting in a total of 54 models. An exhaustive display of these models can be found in Tables 3.6 and 3.7. These changes can be summarized by five alterations to the 2PP IRM:

1) the hyper-parameters were changed for the Normal α prior, keeping $\beta \sim N(0,1)$
2) the hyper-parameters were changed for the Normal β prior, keeping $\alpha \sim N(1,1)$
3) the prior distribution for α was restricted to a positive support
4) the parameterization was altered
5) *probit* and *logit* links were used for all models created in 1-4 above

In order to compare 54 models' goodness of fit and complexity, we employed the deviance information criterion (DIC). DIC is analogous to the Akaike information criterion (AIC) and Bayesian information criterion (BIC), and it can be easily calculated using our MCMC samples. DIC is the standard method of comparison in Bayesian model selection with MCMC simulation. Deviance is a measure of model fit, and is given by

$$D(\boldsymbol{\lambda}) = -2\log[p(\boldsymbol{y}|\boldsymbol{\lambda})], \quad \textbf{(3.5)}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p)$ are the parameters of interest and $p(\boldsymbol{y}|\boldsymbol{\lambda})$ is the data likelihood. The fit of a model under the Bayesian framework can be expressed by the posterior expectation of the deviance, $\overline{D} = E_{\lambda|y}[D]$. The model is then penalized according to the effective number of parameters it uses, $p_D = \overline{D} - D(\bar{\boldsymbol{\lambda}})$, where $\bar{\boldsymbol{\lambda}}$ is the posterior mean vector of $\boldsymbol{\lambda}$. The DIC is defined as the sum of these two quantities,

$$DIC = \overline{D} + p_D \quad \textbf{(3.6)}$$

Since a low $\overline{D}$ indicates good model fit and a low p_D indicates simplicity, a better model will generally possess the lower DIC. Although the DIC says nothing about a model's objective "correctness," it is a powerful tool for determining which of our 54 models fit the data with the most accuracy and efficiency.

Recall from the Methods chapter that the algorithm was implemented using OpenBUGS, in order to simplify model specification. The fitting of these 54 models required very little adjustment to the existing OpenBUGS code, usually a one-line adjustment to the model file. Once the OpenBUGS model files were adapted, the DIC values were easily extracted and compared from the default output using 'BRugs' in R. Table 3.6 displays the full DIC results from all 54 models for the *Raisin Bran* dataset, and Table 3.7 displays those for the *Cola* dataset. The Two-Parameter Models (often called 2PP and 2PL) implement a *discrimination* parameter

and a *difficulty* parameter for each preference test replication. The General One-Parameter (β) models (often called 1PP and 1PL) fix *discrimination* across replications but allow that value to be estimated, while *difficulty* is estimated at each replication. In contrast, the General One-Parameter (α) models fix *difficulty* across replications but allow that value to be estimated, while *discrimination* is estimated at each replication. The One-Parameter models set *discrimination* and *difficulty* to one and zero, respectively. Note that, with a *logit* link, Model 24 becomes the well-known Rasch model. The *probit* link is overwhelmingly favored in these results, as no DIC values for *logit* models are within 15 units of the best DIC for either dataset.

These DIC results also allow a glimpse into the mechanisms behind the Item Response Model and the estimation algorithm. We can now compare adjustments to the prior distributions to adjustments in the parameterization in terms of the resulting DIC. For example, we often see a large difference in DIC between the full two-parameter models and the models with reduced parameterization. As mentioned above, there is also a noticeable difference between the *probit* and *logit* link functions, with *probit* displaying a clear advantage. However, changing the prior distributions did not seem to affect the models' DICs quite so dramatically. For the *Cola* dataset in particular, no fewer than nine models gave a DIC within 5 units of the lowest value. This indicates that the prior specification is not as crucial for the *Cola* data, since many priors produce results that fit the data equivalently. On the other hand, no alternative models were within 5 units of the lowest *Raisin Bran* DIC, indicating that this particular data is marginally more sensitive to the choice of prior distributions.

Although DIC does not identify an objectively "best" model, the 2PP with prior distributions $\alpha \sim N(2, 0.33)$ and $\beta \sim N(0,1)$ resulted in the lowest DIC for both datasets. Further examination revealed that models with relatively low DIC values produced nearly

identical posterior parameter distributions, so this choice of priors is particularly interesting. All posterior mean α values for both datasets are near 1, yet the DIC statistic prefers an informative prior with a mean of 2; the underlying causes must remain a topic of future investigation.

Table 3.6 DIC values for all 54 models applied to the *Raisin Bran* dataset. The notation $N(\mu,\sigma^2)$ indicates a Normal prior distribution with mean μ and variance σ^2, $G(\alpha,\beta)$ indicates a Gamma prior with shape α and rate β, and $TN(\mu,\sigma^2)$ is the Truncated Normal distribution bounded below at zero. The lowest DIC value is given in bold.

ID	Parameterization	Alpha Prior Dist.	Beta Prior Dist.	DIC (Probit)	DIC (Logit)
1	Two-Parameter	N(1, 1)	N(0, 1)	1186	1216
2	Two-Parameter	N(0.5, 0.33)	N(0, 1)	1197	1233
3	Two-Parameter	N(0.5, 1)	N(0, 1)	1192	1214
4	Two-Parameter	N(0.5, 3)	N(0, 1)	1189	1205
5	Two-Parameter	N(1, 0.33)	N(0, 1)	1190	1241
6	Two-Parameter	N(1, 3)	N(0, 1)	1189	1206
7	Two-Parameter	N(2, 0.33)	N(0, 1)	**1178**	1200
8	Two-Parameter	N(2, 1)	N(0, 1)	1186	1221
9	Two-Parameter	N(2, 3)	N(0, 1)	1186	1208
10	Two-Parameter	N(1, 1)	N(-1, 0.33)	1186	1216
11	Two-Parameter	N(1, 1)	N(-1, 1)	1186	1212
12	Two-Parameter	N(1, 1)	N(-1, 3)	1187	1211
13	Two-Parameter	N(1, 1)	N(0, 0.33)	1195	1226
14	Two-Parameter	N(1, 1)	N(0, 3)	1190	1211
15	Two-Parameter	N(1, 1)	N(1, 0.33)	1202	1234
16	Two-Parameter	N(1, 1)	N(1, 1)	1193	1220
17	Two-Parameter	N(1, 1)	N(1, 3)	1186	1214
18	Two-Parameter	Gamma(3, 3)	N(0, 1)	1195	1212
19	Two-Parameter	Gamma(1, 1)	N(0, 1)	1192	1206
20	Two-Parameter	Gamma(0.33, 0.33)	N(0, 1)	1191	1206
21	Two-Parameter	TN(1, 1)	N(0, 1)	1192	1206
22	Gen. One-Parameter (β)	N(1, 1)	N(0, 1)	1202	1216
23	Gen. One-Parameter (β)	N(1, 3)	N(0, 1)	1203	1215
24	One-Parameter (β)	---	N(0, 1)	1201	1264
25	Gen. One-Parameter (α)	N(1, 1)	N(0, 1)	1212	1227
26	Gen. One-Parameter (α)	N(1, 1)	N(0, 3)	1213	1226
27	One-Parameter (α)	N(1, 1)	---	1229	1250

Table 3.7 DIC values for all 54 models applied to the *Cola* dataset. The notation $N(\mu,\sigma^2)$ indicates a Normal prior distribution with mean μ and variance σ^2, $G(\alpha,\beta)$ indicates a Gamma prior with shape α and rate β, and $TN(\mu,\sigma^2)$ is the Truncated Normal distribution bounded below at zero. The lowest DIC values are given in bold.

ID	Parameterization	Alpha Prior Dist.	Beta Prior Dist.	DIC (Probit)	DIC (Logit)
1	Two-Parameter	N(1, 1)	N(0, 1)	1487	1509
2	Two-Parameter	N(0.5, 0.33)	N(0, 1)	1496	1516
3	Two-Parameter	N(0.5, 1)	N(0, 1)	1493	1507
4	Two-Parameter	N(0.5, 3)	N(0, 1)	1490	1502
5	Two-Parameter	N(1, 0.33)	N(0, 1)	1491	1521
6	Two-Parameter	N(1, 3)	N(0, 1)	1491	1504
7	Two-Parameter	N(2, 0.33)	N(0, 1)	**1484**	1503
8	Two-Parameter	N(2, 1)	N(0, 1)	**1484**	1512
9	Two-Parameter	N(2, 3)	N(0, 1)	**1484**	1503
10	Two-Parameter	N(1, 1)	N(-1, 0.33)	1489	1507
11	Two-Parameter	N(1, 1)	N(-1, 1)	1488	1508
12	Two-Parameter	N(1, 1)	N(-1, 3)	1488	1508
13	Two-Parameter	N(1, 1)	N(0, 0.33)	1490	1509
14	Two-Parameter	N(1, 1)	N(0, 3)	1485	1508
15	Two-Parameter	N(1, 1)	N(1, 0.33)	1495	1512
16	Two-Parameter	N(1, 1)	N(1, 1)	1492	1510
17	Two-Parameter	N(1, 1)	N(1, 3)	1487	1509
18	Two-Parameter	Gamma(3, 3)	N(0, 1)	1494	1505
19	Two-Parameter	Gamma(1, 1)	N(0, 1)	1490	1504
20	Two-Parameter	Gamma(0.33, 0.33)	N(0, 1)	1490	1499
21	Two-Parameter	TN(1, 1)	N(0, 1)	1490	1503
22	One-Parameter (General)	N(1, 1)	N(0, 1)	1519	1524
23	One-Parameter (General)	N(1, 3)	N(0, 1)	1517	1523
24	One-Parameter	---	N(0, 1)	1513	1520
25	Gen. One-Parameter (Alpha)	N(1, 1)	N(0, 1)	1491	1505
26	Gen. One-Parameter (Alpha)	N(1, 1)	N(0, 3)	1490	1504
27	One-Parameter (Alpha)	N(1, 1)	---	1508	1526

Chapter 4 - Computation

The Gibbs sampler for this model requires great care in terms of programming and model specification. Whereas the primary advantages to the use of the Item Response Model in the preference testing context are related to interpretation, most of the potential drawbacks are related to computation. Although the MCMC simulations are easily accessible, certain conditions may produce unfavorable results. Such issues will be addressed here, along with a summary of the computational journey which brought us to the published results.

The first step in applying the Two-Parameter IRM was the *Shyness* data example, which outlined the step-by-step process for sampling values from the appropriate posterior distributions. We first programmed this algorithm into the function 'gibbs', whose sampling functions are contained in the packages presented by Trautmann (2014) and Venables (2002). The full function can be found in Appendix A. The function inputs the data, prior information about all three of the model's parameters, the number of iterations, and the duration of the burn-in period. Using a loop structure, we then employ the aforementioned latent variable approach and sample from the full conditional densities of every α, β, and θ parameter, as outlined in Equations 2.4, 2.5, and 2.6 in the Methods chapter. In this approach, the sampled values of the latent variable Z_{ij} are stored, although we do not use them. With the full collection of sampled values stored in R objects, plot generation is simple and efficient.

Next, the five hypothetical datasets were created to evaluate the model's performance under specific conditions. Initially, the *Increasing Difficulty* datasets featured 81 consumers, where the observed proportion of preferences for A in each replication were $(p_1, p_2, p_3, p_4) = (1, 2/3, 1/3, 0)$. The original *Monotone Increasing Difficulty* dataset can be seen in Table 4.1. Whereas all of these preferences were contiguous in the *Monotone* dataset, their location was

randomized in the *Non-Monotone* dataset. These locations were only randomized once, and the resulting dataset was used through the entirety of the estimation process. This original *Non-Monotone Increasing Difficulty* dataset can be found in Table 4.2. However, when verifying algorithm convergence, the *Non-Monotone* dataset experienced less than ideal mixing, particularly for the four α parameters. The initial structure used observed proportions of A preferences of 100% and 0% in replications 1 and 4, respectively. Such extreme proportions can often cause computational issues, so, to obtain a dataset for illustration purposes, more response patterns were added to make the observed preference proportions more realistic. This was accomplished by adding 27 consumers who consistently preferred product A and 27 consumers who consistently preferred product B. The adjusted *Monotone* and *Non-Monotone* datasets were shown in Tables 3.4 and 3.5, respectively. The addition of B preferences to the first replication and A preferences to the last replication greatly improved the algorithm's mixing, though it is far from exemplary. Plots of the sampled α_3 values for the *Non-Monotone Increasing Difficulty* and *Extreme* datasets are shown in Figure 4.1 for comparison.

In order to expedite the exploration of different model specifications, we used OpenBUGS software, specifically designed for "Bayesian inference Using Gibbs Sampling" with adaptive rejection sampling. This method does not require the use of the data augmentation method presented in the Methods chapter; instead, the model in Formula 1.10 is used. The full OpenBUGS model file and data file code can be found in Appendix B. As stated earlier, the appropriate model files were easily obtained by adapting the code written for the two-parameter logistic item response model (2PL IRM) by Curtis (2010). Once these model and data files are deemed functional, they are inputted to R via the 'BRugs' package, which runs the OpenBUGS sampling algorithm and returns the sampled values in R. This allowed use of R's powerful

Table 4.1 The three distinct response patterns present in the original *Monotone Increasing Difficulty* dataset.

	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	0	3	27
	1	1	0	0	2	27
	1	0	0	0	1	27
Total	81	54	27	0		

Table 4.2 The four distinct response patterns present in the original *Non-Monotone Increasing Difficulty* dataset.

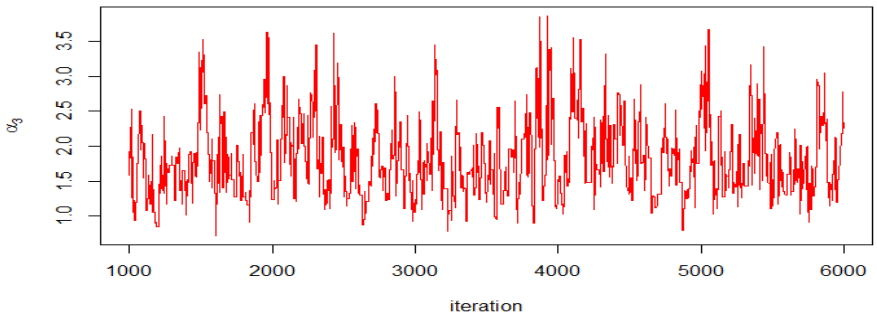
	Rep 1	Rep 2	Rep 3	Rep 4	Total A Selections	Count
	1	1	1	0	3	16
	1	1	0	0	2	38
	1	0	1	0	2	11
	1	0	0	0	1	16
Total	81	54	27	0		

graphics capabilities, while still utilizing OpenBUGS's efficient computational process. Table 4.3 compares the elapsed procedure times of the sampling process for both the *Raisin Bran* and *Cola* datasets between the two methods of computation. While marginally higher in computational costs, 'BRugs' saved time via the simplicity of the OpenBUGS syntax. The necessary adjustments to the model specifications in our DIC calculations were typically one-line changes in prior distribution or link function, where an identical adjustment to our original R function would have required detailed specification of the appropriate conditional distributions.

In cases where OpenBUGS is not available or the user is unfamiliar with BUGS syntax, many standalone R packages are available for the purpose of Item Response Modeling. The 'ltm' package by Rizopoulos (2006) fits several variations of the latent trait model, where 'eRm',

Figure 4.1 Plots of the sampled values over time for the third *item discrimination* parameter. Shown are values resulting from the *Non-Monotone Increasing Difficulty* dataset (a) and the *Extreme* dataset (b). Although much improved over the initial dataset structure, the *Non-Monotone Increasing Difficulty* data still experiences minor mixing issues, where the *Extreme* data plot is ideal.

(a)



(b)

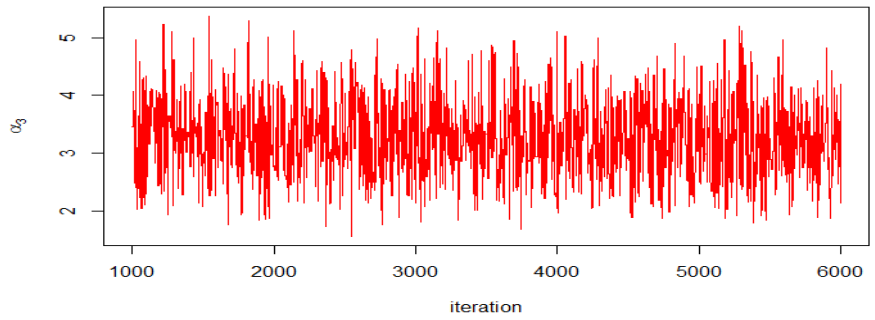


Table 4.3 A comparison of elapsed computing times between two programming methods for the item response model with Gibbs Sampling, (6000 iterations). The Manual method uses a custom R function with a single nested loop, requiring the installation of the 'truncnorm' and 'MASS' packages. The alternative was use of the 'BRugs' package in R, whose primary function 'BRugsFit' allows OpenBUGS to sample values through the R interface. The Manual programming proved quicker in computation, but required much more effort to specify.

	Programming Method	
	Manual	'BRugs'
Raisin Bran Data	48.61 sec	56.81 sec
Cola Data	49.56 sec	58.20 sec

presented by Mair (2007), focuses on extensions of the Rasch Model, both in unidimensional environments. The 'mirt' package from Chalmers (2012) was created for estimation of multidimensional IRT parameters. From a Bayesian perspective 'ltbayes' by Johnson (2014) and 'MCMCpack' by Martin, Quinn, and Park (2011) are well-known alternatives. Mair (2015) names and explains these and many more packages which can be used for item response modeling in R.

Lastly, the Two-Parameter Item Response Model naturally brings an issue of non-identifiability, due to the product of *discrimination* and *latent preference*. α is expected to be positive, because we expect an increase in a consumer's underlying preference for product A to be accompanied by a higher probability of selecting A as the preferred product. Therefore, positive *latent preference* values can be interpreted as above-average preferences for product A, and negative *latent preference* values indicate below-average preferences for product A. However, samples from OpenBUGS will occasionally contain negative *discrimination* values and, thus, *latent preference* values with opposite interpretations. This problem can be circumvented by restricting $\alpha > 0$. If one would rather not sacrifice the flexibility of allowing α to be freely estimated, it will often suffice to suggest such a restriction through a prior distribution with either a positive mean or an entirely positive support. Johnson and Albert (1999) argues that identifiability fails to be a problem when a proper prior is selected for the *latent preference* parameters, and Curtis (2010) states that the variance of the *latent preferences* must be constant over time to establish identifiability, but we still encountered difficulties. In these instances, it is still possible to interpret *latent preference* and *discrimination*: consumer i possesses an above-average preference for product A in replication j when the sign of θ_i matches the sign of α_j and a below-average preference for product A when the signs are different.

However, this is not a solution to the issue of non-identifiability, and the interpretation is much more easily understood when positive and negative values of θ_i indicate above-average and below-average *latent preferences*, respectively. The search for such a solution will remain a topic of future research. Because the primary goal of this research is to propose the model and aid in its interpretation, we only present results which contained mostly positive *discrimination* values.

Although the application of a hierarchical version of the IRM would be a natural progression, our focus on interpretation led to use of a simpler version of the model. The hierarchical model could be useful for further research and in practice.

Chapter 5 - Discussion

The primary problem with conventional analysis of preference test data using p_A, the probability of preferring product A over product B, is the inability to allow these probabilities to differ among consumers. Cochrane et al. (2005) found that results become more stable with replicated preference tests, due to potential inconsistent consumer preferences over time. The Two-Parameter IRM allows this flexibility in probabilities and accounts for multiple preference tests per customer, making it a powerful tool for preference test data analysis. Although other Bayesian models could be applied in this setting, our goal was to explore the interpretation of the Two-Parameter IRM, not perform an exhaustive search for the best model.

In addition, the Two-Parameter IRM estimates characteristics of the preference tests themselves, rather than only characteristics of the consumers. Given a consumer's *latent ability* and a replication's *discrimination* and *difficulty*, we may be able to predict a preference response more accurately than by using the consumer's history of responses alone. This can be determined by investigating several models' posterior predictive distributions. Secondly, the *discrimination* and *difficulty* measures provide insight into consumer preferences over time, information which transcends any specific product or dataset. Even with two different products with entirely different consumer demographics, there may exist similarities in patterns of recorded preferences from one replication to the next. These item parameters represent the similarities and differences in response patterns between different products and consumer panels. As such, the IRM may be better-suited for exploring the nature of consumer preferences than for making conclusions about a specific product. Although it may be difficult to make product decisions based directly on results from the IRM, this information could greatly benefit the

advertising and market research sectors by approaching the psychology of preference testing through a proven mathematical model.

References

Albert, James H., and Siddhartha Chib. "Bayesian Analysis of Binary and Polychotomous Response Data." *Journal of the American Statistical Association* 88.422 (1993): 669-679.

Bi, Jian. "Difficulties and a Way Out: A Bayesian Approach for Sensory Difference and Preference Tests." *Journal of Sensory Studies* 18.1 (2003): 1-18.

Carlin, Bradley P., and Thomas A. Louis. *Bayesian Methods for Data Analysis.* 3rd ed (2009). Boca Raton: Taylor & Francis Group, LLC.

Chalmers, R. Philip. "{mirt}: A Multidimensional Item Response Theory Package for the {R} Environment." *Journal of Statistical Software* 48.6 (2012): 1-29.

Cochrane, Chun-Yen Chang, Suzanne Dubnicka, and Thomas Loughin. "Comparison of Methods for Analyzing Replicated Preference Tests." *Journal of Sensory Studies* 20.6 (2005): 484-502.

Curtis, S. McKay. "BUGS Code for Item Response Theory." *Journal of Statistical Software* 36 (2010).

Ennis, Daniel M., and Jian Bi. "The Beta-Binomial Model: Accounting for Inter-Trial Variation in Replicated Difference and Preference Tests." *Journal of Sensory Studies* 13.4 (1998): 389-412.

Geman, Stuart, and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6.6 (1984): 721-741.

Greenberg, Allan, and Sy Collins. "Paired Comparison Taste Tests: Some Food for Thought." *Journal of Marketing Research* 3.1 (1966): 76-80.

Johnson, Timothy R. "ltbayes: Simulation-Based Bayesian Inference for Latent Traits of Item Response Models." R package version 0.3. (2014). 13 July 2015. <http://CRAN.R-project.org/package=ltbayes>

Johnson, Valen E., and James H. Albert. *Ordinal Data Modeling* (1999). New York: Springer-Verlag.

Lunn, David J., Andrew Thomas, Nicky Best, and David Spiegelhalter. "WinBUGS – A Bayesian modelling framework: Concepts, structure, and extensibility." *Statistics and Computing* 10.4 (2000): 325-337.

Mair, Patrick. "CRAN Task View: Psychometric Models and Methods." *The Comprehensive R Archive Network.* n.p., 4 May 2015.

Mair, Patrick, and R. Hatzinger. “Extended Rasch modeling: The eRm package for the application of IRT models in R.” *Journal of Statistical Software* 20.9 (2007): 1-20.

Martin, Andrew D., Kevin M. Quinn, and John Hee Park. “{MCMCpack}: Markov Chain Monte Carlo in {R}.” *Journal of Statistical Software* 42.9 (2011): 22.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. “Equation of State Calculations by Fast Computing Machines.” *The Journal of Chemical Physics* 21.6 (1953): 1087-1092.

Meyners, Michael. “Proper and Improper Use and Interpretation of Beta-Binomial Models in the Analysis of Replicated Difference and Preference Tests.” *Food Quality and Preference* 18.5 (2007): 741-750.

R Core Team. “R: A Language and Environment for Statistical Computing.” R Foundation for Statistical Computing, Vienna, Austria (2014). 13 July 2015. <http://www.R-project.org/>

Rizopoulos, Dimitris. “ltm: An R package for Latent Variable Modelling and Item Response Theory Analyses.” *Journal of Statistical Software* 17.5 (2006): 1-25.

Thomas, Andrew, Bob O'Hara, Uwe Ligges, and Sibylle Sturtz. “Making BUGS Open.” *R News* 6.1 (2006): 12-17.

Trautmann, Heike, Detlef Steuer, Olaf Mersmann, and Björn Bornkamp. “truncnorm: Truncated normal distribution.” R Package Version 1.0-7 (2014). 13 July 2015. <http://CRAN.R-project.org/package=truncnorm>

Venables, William N., and Brian D. Ripley. *Modern Applied Statistics with S.* 4th ed (2002). New York: Springer.

Wilke, Kristine D., Chun-Yen Chang Cochrane, and Edgar Chambers IV. “Multiple Preference Tests Can Provide More Information on Consumer Preferences.” *Journal of Sensory Studies* 21.6 (2006): 612-625.

Appendix A - R Code for Gibbs Sampling

```
# Install Necessary Packages #
install.packages("truncnorm")
library(truncnorm)
install.packages("MASS")
library(MASS)

# gibbs function runs the Gibbs Sampling algorithm and returns arrays of
#     parameter values

gibbs <- function(y,mu.a=1,mu.b=0,sigma0=diag(2),mu.theta=0,niter=6000,
      nburn=1000){

  # data dimensions
  n <- dim(y)[1]
  k <- dim(y)[2]

  # creation of output storage arrays
  z <- array(0,dim=c(n,k,niter))
  a <- matrix(mu.a,nrow=niter,ncol=k)
  b <- matrix(mu.b,nrow=niter,ncol=k)
  theta <- matrix(mu.theta,nrow=niter,ncol=n)

  # these are used to store parameter values at each iteration
  zt <- z[,,1]
  at <- a[1,]
  bt <- b[1,]
  thetat <- theta[1,]

  # the Gibbs Sampling loop
  for (i in 2:niter){
    # sample latent values zij
    at.mat <- matrix(at,nrow=n,ncol=k,byrow=T)
    bt.mat <- matrix(bt,nrow=n,ncol=k,byrow=T)
    thetat.mat <- matrix(thetat,nrow=n,ncol=k,byrow=F)
    mean.z <- at.mat*thetat.mat-bt.mat
    zt <- ifelse(y==1,rtruncnorm(n*k,a=0,mean=mean.z),
      rtruncnorm(n*k,b=0,mean=mean.z))
```

```
    # sample latent traits thetai
    d <- sum(at^2)+1
    mean.theta <- apply(at.mat*(zt+bt.mat),1,sum)/d
    se.theta <- sqrt(1/d)
    thetat <- rnorm(n,mean.theta,se.theta)

    # sample item parameters (aj,bj)
    X <- cbind(thetat,-1)
    sig.inv <- solve(sigma0)
    mu0 <- c(mu.a,mu.b)
    for (j in 1:k){
      vj <- solve(t(X)%*%X+sig.inv)
      mj <- vj%*%(t(X)%*%zt[,j]+sig.inv%*%mu0)
      abt <- mvrnorm(1,mj,vj)
      at[j] <- abt[1]
      bt[j] <- abt[2]
    }

    # store sampled values
    z[,,i] <- zt
    a[i,] <- at
    b[i,] <- bt
    theta[i,] <- thetat
  }
  return(list(z=z[,,(nburn+1):niter],a=a[(nburn+1):niter,],
      b=b[(nburn+1):niter], theta=theta[(nburn+1):niter,]))
}
# z is 3-D array with dimensions [n,k,iter]
# a is 2-D array with dimensions [iter,k]
# b is 2-D array with dimensions [iter,k]
# theta is 2-D array with dimensions [iter,n]

##########################
## Shyness Data Example ##
##########################

# Shyness dataset can be found at
#     http://www-math.bgsu.edu/~albert/ord_book/Chapter6/ratings.dat

set.seed(20)
sim <- gibbs(shyness) # all other arguments are default

# pull out results for easy plot generation
a <- sim$a
b <- sim$b
```

```
z <- sim$z
t <- sim$theta
k <- length(a[1,])
n <- length(t[1,])

# # # # # # # # # # # # # # # # # # # # #
# Plots for Shyness Data in Figure 1.3 #
# # # # # # # # # # # # # # # # # # # # #

## Whisker Plots for A
list.a <- lapply(seq_len(ncol(a)), function(i) a[,i])
means.a <- apply(a,2,mean)
stripchart(list.a,col="white",axes=F,frame.plot=T,group.names=rep("",k),xlab=
      "Item",ylab=expression(alpha),ylim=c(-1.8,4.5),vertical=T)
for (i in 1:k) points(i,means.a[i],pch=19,cex=0.5)
for (i in 1:k) lines(c(i,i),quantile(a[,i],c(.05,.95)))
for (i in 1:k) lines(c(i-.5,i+.5),rep(quantile(a[,i],c(.05)),2))
for (i in 1:k) lines(c(i-.5,i+.5),rep(quantile(a[,i],c(.95)),2))
axis(1,at=seq(1,k,10))
axis(2,at=seq(-3,5))

## Whisker Plots for B
list.b <- lapply(seq_len(ncol(b)), function(i) b[,i])
means.b <- apply(b,2,mean)
stripchart(list.b,col="white",axes=F,frame.plot=T,group.names=rep("",k),xlab=
      "Item",ylab=expression(beta),ylim=c(-3,3),vertical=T)
for (i in 1:k) points(i,means.b[i],pch=19,cex=0.5)
for (i in 1:k) lines(c(i,i),quantile(b[,i],c(.05,.95)))
for (i in 1:k) lines(c(i-.5,i+.5),rep(quantile(b[,i],c(.05)),2))
for (i in 1:k) lines(c(i-.5,i+.5),rep(quantile(b[,i],c(.95)),2))
axis(1,at=seq(10,k,10))
axis(2,at=seq(-3,3))

## Whisker Plots for Theta
list.t <- lapply(seq_len(ncol(t)), function(i) t[,i])
means.t <- apply(t,2,mean)
stripchart(list.t,col="white",axes=F,frame.plot=T,group.names=rep("",n),xlab=
      "Student",ylab=expression(theta),ylim=c(-2.2,2),vertical=T)
for (i in 1:n) points(i,means.t[i],pch=19,cex=0.5)
for (i in 1:n) lines(c(i,i),quantile(t[,i],c(.05,.95)))
for (i in 1:n) lines(c(i-.5,i+.5),rep(quantile(t[,i],c(.05)),2))
for (i in 1:n) lines(c(i-.5,i+.5),rep(quantile(t[,i],c(.95)),2))
axis(1,at=seq(10,n,10))
axis(2,at=seq(-3,3))
```

```
#########################
## Real-World Datasets ##
#########################

# note that 0='preference for B', 1='preference for A'

# Raisin Bran
rb<-matrix(c(rep(c(1,1,1,1),139),
             rep(c(1,1,1,0),6),
             rep(c(1,1,0,1),13),
             rep(c(1,0,1,1),16),
             rep(c(0,1,1,1),28),
             rep(c(1,1,0,0),5),
             rep(c(1,0,1,0),10),
             rep(c(0,1,1,0),8),
             rep(c(1,0,0,1),9),
             rep(c(0,1,0,1),6),
             rep(c(0,0,1,1),13),
             rep(c(1,0,0,0),11),
             rep(c(0,1,0,0),8),
             rep(c(0,0,1,0),7),
             rep(c(0,0,0,1),12),
             rep(c(0,0,0,0),14)),ncol=4,byrow=T)

# Cola
cola<-matrix(c(rep(c(1,1,1,1),65),
               rep(c(1,1,1,0),17),
               rep(c(1,1,0,1),24),
               rep(c(1,0,1,1),19),
               rep(c(0,1,1,1),19),
               rep(c(1,1,0,0),16),
               rep(c(1,0,1,0),11),
               rep(c(0,1,1,0),14),
               rep(c(1,0,0,1),15),
               rep(c(0,1,0,1),9),
               rep(c(0,0,1,1),17),
               rep(c(1,0,0,0),9),
               rep(c(0,1,0,0),12),
               rep(c(0,0,1,0),20),
               rep(c(0,0,0,1),8),
               rep(c(0,0,0,0),21)),ncol=4,byrow=T)
```

```
# # # # # # # # # # # # # # # # # # # # # # # #
# Results and Plots for Real-World Datasets #
# # # # # # # # # # # # # # # # # # # # # # # #

set.seed(10)
gibbs.rb <- gibbs(rb)

# Summarize results
a <- gibbs.rb$a
b <- gibbs.rb$b
t <- gibbs.rb$theta
z <- gibbs.rb$z
n <- length(t[1,])
k <- length(a[1,])
iter <- length(a[,1])

set.seed(10)
gibbs.cola <- gibbs(cola)

# Summarize Results
a <- gibbs.cola$a
b <- gibbs.cola$b
t <- gibbs.cola$theta
z <- gibbs.cola$z
n <- length(t[1,])
k <- length(a[1,])
iter <- length(a[,1])

############################
## Hypothetical Datasets ##
############################

# Balanced
y <- matrix(c(rep(c(1,1,1,1),5),
              rep(c(1,1,1,0),5),
              rep(c(1,1,0,1),5),
              rep(c(1,0,1,1),5),
              rep(c(0,1,1,1),5),
              rep(c(1,1,0,0),5),
              rep(c(1,0,1,0),5),
              rep(c(0,1,1,0),5),
              rep(c(1,0,0,1),5),
              rep(c(0,1,0,1),5),
              rep(c(0,0,1,1),5),
              rep(c(1,0,0,0),5),
```

```
            rep(c(0,1,0,0),5),
            rep(c(0,0,1,0),5),
            rep(c(0,0,0,1),5),
            rep(c(0,0,0,0),5)),ncol=4,byrow=T)

# Extreme
y <- matrix(c(rep(c(1,1,1,1),40),rep(c(0,0,0,0),40)),ncol=4,byrow=T)

# Increasing Discrimination
# Dataset from Table 3.3 was written to a text file and imported as follows:
y <- as.matrix(read.table("examplefilepath\\Increasing Discrimination
     Data.txt"))

# Increasing Difficulty (Monotone)
# 81 consumers originally, 54 consumers added later (see Results chapter)
y <- matrix(c(rep(1,81),rep(1,54),rep(0,27),rep(1,27),rep(0,54),rep(0,81))
     ,ncol=4)
y <- rbind(matrix(rep(1,108),ncol=4),y,matrix(rep(0,108),ncol=4))

# Increasing Difficulty (Non-Monotone)
# Uses Incr. Diff. (Monotone) data and randomizes 2nd and 3rd columns
# 81 consumers originally, 54 consumers added later (see Results chapter)
set.seed(10)
y <- matrix(c(rep(1,81),sample(y[,2],81),sample(y[,3],81),rep(0,81)),ncol=4)
y <- rbind(matrix(rep(1,108),ncol=4),y,matrix(rep(0,108),ncol=4))

# # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# Results and Plots for Hypothetical Datasets #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # #

set.seed(7)
sim <- gibbs(y)

a <- sim$a
b <- sim$b
t <- sim$theta
z <- sim$z
n <- length(t[1,])
k <- length(a[1,])
iter <- length(a[,1])
```

Appendix B - R and OpenBUGS Code for using 'BRugs'

```
#####################################################################
## OpenBUGS Model File: Two-Parameter Probit Item Response Model ##
#####################################################################

model {
      # Rearrange data into full dataset
      for (i in 1:culm[1]) {
            for (j in 1:p) {
                  Y[i, j] <- response[1,j]
            }
      }
      for (i in 2:R) {
            for (j in culm[i-1] + 1 : culm[i]) {
                  for (k in 1:p) {
                        Y[j, k] <- response[i, k]
                  }
            }
      }
      # Calculate probability of preferring A for each replication
      for (j in 1:p) {
            P[j] <- phi(-beta[j] / sqrt(1 + pow(alpha[j],2)))
      }
      # 2-Parameter Probit Model specification
      for (i in 1:n) {
            for (j in 1:p) {
                  Y[i, j] ~ dbern(prob[i, j])
                  prob[i, j] <- phi(alpha[j] * theta[i] - beta[j])
            }
            theta[i] ~ dnorm(0, 1)
      }
      # Specification of Prior Distributions
      for (j in 1:p) {
            beta[j] ~ dnorm(0,1)
            alpha[j] ~ dnorm(1,1)
      }
}
```

```
#########################
## OpenBUGS Data Files ##
#########################

## Raisin Bran Data ##
list(n=305, R=16, p=4, culm=c(139, 145, 158, 174, 202, 207, 217, 225, 234,
240, 253, 264, 272, 279, 291, 305), response=structure(.Data=c(
      1,1,1,1,
      1,1,1,0,
      1,1,0,1,
      1,0,1,1,
      0,1,1,1,
      1,1,0,0,
      1,0,1,0,
      0,1,1,0,
      1,0,0,1,
      0,1,0,1,
      0,0,1,1,
      1,0,0,0,
      0,1,0,0,
      0,0,1,0,
      0,0,0,1,
      0,0,0,0), .Dim=c(16,4)))

## Cola Data ##
list(n=295, R=16, p=4, culm=c(65, 82, 106, 125, 144, 160, 171, 185, 200, 209,
226, 235, 247, 267, 275, 296), response=structure(.Data=c(
      1,1,1,1,
      1,1,1,0,
      1,1,0,1,
      1,0,1,1,
      0,1,1,1,
      1,1,0,0,
      1,0,1,0,
      0,1,1,0,
      1,0,0,1,
      0,1,0,1,
      0,0,1,1,
      1,0,0,0,
      0,1,0,0,
      0,0,1,0,
      0,0,0,1,
      0,0,0,0), .Dim=c(16,4)))
```

```
## Balanced Data ##
list(n=80, R=16, p=4, culm=c(5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80),
response=structure(.Data=c(
      1,1,1,1,
      1,1,1,0,
      1,1,0,1,
      1,0,1,1,
      0,1,1,1,
      1,1,0,0,
      1,0,1,0,
      0,1,1,0,
      1,0,0,1,
      0,1,0,1,
      0,0,1,1,
      1,0,0,0,
      0,1,0,0,
      0,0,1,0,
      0,0,0,1,
      0,0,0,0), .Dim=c(16,4)))

## Extreme Data ##
list(n=80, R=2, p=4, culm=c(40, 80), response=structure(.Data=c(
      1,1,1,1,
      0,0,0,0), .Dim=c(2,4)))

## Increasing Discrimination Data ##
list(n=60, R=15, p=4, culm=c(5,7,12,21,28,33,34,37,40,41,48,53,55,56,60),
response=structure(.Data=c(
      1,1,1,1,
      1,1,0,1,
      1,0,1,1,
      0,1,1,1,
      1,1,0,0,
      1,0,1,0,
      0,1,1,0,
      1,0,0,1,
      0,1,0,1,
      0,0,1,1,
      1,0,0,0,
      0,1,0,0,
      0,0,1,0,
      0,0,0,1,
      0,0,0,0), .Dim=c(15,4)))
```

```
## Monotone Increasing Difficulty Data ##
list(n=135, R=5, p=4, culm=c(27,54,81,108,135), response=structure(.Data=c(
      1,1,1,1,
      1,1,1,0,
      1,1,0,0,
      1,0,0,0,
      0,0,0,0), .Dim=c(5,4)))

## Non-Monotone Increasing Difficulty Data ##
list(n=135, R=6, p=4, culm=c(27,43,81,92,108,135),
response=structure(.Data=c(
      1,1,1,1,
      1,1,1,0,
      1,1,0,0,
      1,0,1,0,
      1,0,0,0,
      0,0,0,0), .Dim=c(6,4)))

############################
## R Scripts for 'BRugs' ##
############################

# 'BRugsFit' will run simulations in OpenBUGS and store results in R

install.packages("BRugs")
library(BRugs)

# Raisin Bran Data

sim.rb <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Raisin Bran
      Data.txt", parametersToSave=c("alpha","beta","theta","P"),nBurnin=1000,
      nIter=5000, seed=10)

# Cola Data

sim.c <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Cola
      Data.txt", parametersToSave=c("alpha","beta","theta","P"),nBurnin=1000,
      nIter=5000, seed=10)

# Balanced Data

sim.b <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Balanced
      Data.txt", parametersToSave=c("alpha","beta","theta","P"),nBurnin=1000,
      nIter=5000, seed=10)
```

```
# Extreme Data

sim.e <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Extreme
      Data.txt", parametersToSave=c("alpha","beta","theta","P"),nBurnin=1000,
      nIter=5000, seed=10)

# Increasing Discrimination Data

sim.d <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Incr. Discr.
      Data.txt", parametersToSave=c("alpha","beta","theta","P"),
      nBurnin=1000, nIter=5000, seed=10)

# Increasing Difficulty (Monotone) Data

sim.dm <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Incr. Diff.
      (M) Data.txt", parametersToSave=c("alpha","beta","theta","P"),
      nBurnin=1000, nIter=5000, seed=10)


# Increasing Difficulty (Non-Monotone) Data

sim.dnm <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Incr. Diff.
      (NM) Data.txt", parametersToSave=c("alpha","beta","theta","P"),
      nBurnin=1000, nIter=5000, seed=10)

########################################
## Plot Generation for OpenBUGS Results ##
########################################

# # # # # # # # # # # # #
#   RB:        sim.rb   #
#   Cola:      sim.c    #
#   Balanced:  sim.b    #
#   Extreme:   sim.e    #
#   Discr.:    sim.d    #
#   Diff. (M): sim.dm   #
#   Diff. (NM): sim.dnm #
# # # # # # # # # # # # #

# Boxplots for alpha, beta, P
boxplot(samplesSample("alpha[1]"),samplesSample("alpha[2]"),samplesSample
      ("alpha[3]"), samplesSample("alpha[4]"),xlab="Replication",
      ylab=expression(alpha))
```

```
axis(1,at=seq(1,4))
boxplot(samplesSample("beta[1]"),samplesSample("beta[2]"),samplesSample("beta
      [3]"), samplesSample("beta[4]"),xlab="Replication", ylab=
      expression(beta))
axis(1,at=seq(1,4))
boxplot(samplesSample("P[1]"),samplesSample("P[2]"),samplesSample("P[3]"),
        samplesSample("P[4]"),xlab="Replication",ylab="p")
axis(1,at=seq(1,4))

# Example Histograms and Sim. Sequences for alpha, beta, theta
hist(samplesSample("alpha[3]"),xlab=expression(alpha[3]),main=" ")
hist(samplesSample("beta[2]"),xlab=expression(beta[2]),main=" ")
hist(samplesSample("theta[30]"),xlab=expression(theta[30]),main=" ")

samplesHistory("alpha[3]",mfrow=c(1,1),ylab=expression(alpha[3]),main=" ")
samplesHistory("beta[2]",mfrow=c(1,1),ylab=expression(beta[2]),main=" ")
samplesHistory("theta[30]",mfrow=c(1,1),ylab=expression(theta[30]),main=" ")

  # Optional Kernel Density Estimates for above nodes
  samplesDensity("alpha[3]",mfrow=c(1,1))
  samplesDensity("beta[2]",mfrow=c(1,1))
  samplesDensity("theta[30]",mfrow=c(1,1))

# Whisker Plots for theta

  # sim.b used for example, use appropriate sim results
  n <- dim(sim.b$Stats)[1]-12
  means.t <- tail(sim.b$Stats[,1],n)
  quant.t <- cbind(tail(sim.b$Stats[,4],n),tail(sim.b$Stats[,6],n))

t <- matrix(seq(1,n*10),nrow=10)
list.t <- lapply(seq_len(ncol(t)), function(i) t[,i])
stripchart(list.t,col="white",axes=F,frame.plot=T,group.names=rep("",n),xlab=
      "Panelist", ylab=expression(theta),ylim=c(-3,3),vertical=T)
for (i in 1:n) points(i,means.t[i],pch=19,cex=0.5)
for (i in 1:n) lines(c(i,i),c(quant.t[i,1],quant.t[i,2]))
for (i in 1:n) lines(c(i-.5,i+.5),rep(quant.t[i,1],2))
for (i in 1:n) lines(c(i-.5,i+.5),rep(quant.t[i,2],2))
axis(1,at=seq(10,n,10))
axis(2,at=seq(-3,3))
```

```
################################
## Model Comparison Using DIC ##
################################

#############################################################
##        LIST OF PRIOR DISTRIBUTIONS AND NOTATION         ##
##                                                         ##
##  Standard: a ~ N(1,1), b ~ N(0,1)                       ##
##  a - (.5,.33): a ~ N(.5,.33), b ~ N(0,1)                ##
##  a - (.5,1): a ~ N(.5,1), b ~ N(0,1)                    ##
##  a - (.5,3): a ~ N(.5,3), b ~ N(0,1)                    ##
##  a - (1,.33): a ~ N(1,.33), b ~ N(0,1)                  ##
##  a - (1,3): a ~ N(1,3), b ~ N(0,1)                      ##
##  a - (2,.33): a ~ N(2,.33), b ~ N(0,1)                  ##
##  a - (2,1): a ~ N(2,1), b ~ N(0,1)                      ##
##  a - (2,3): a ~ N(2,3), b ~ N(0,1)                      ##
##  b - (-1,.33): a ~ N(1,1), b ~ N(-1,.33)                ##
##  b - (-1,1): a ~ N(1,1), b ~ N(-1,1)                    ##
##  b - (-1,3): a ~ N(1,1), b ~ N(-1,3)                    ##
##  b - (0,.33): a ~ N(1,1), b ~ N(0,.33)                  ##
##  b - (0,3): a ~ N(1,1), b ~ N(0,3)                      ##
##  b - (1,.33): a ~ N(1,1), b ~ N(1,.33)                  ##
##  b - (1,1): a ~ N(1,1), b ~ N(1,1)                      ##
##  b - (1,3): a ~ N(1,1), b ~ N(1,3)                      ##
##  a - gamma(3,3): a ~ gamma(3,3), b ~ N(0,1)             ##
##  a - gamma(1,1): a ~ gamma(1,1), b ~ N(0,1)             ##
##  a - gamma(.33,.33): a ~ gamma(.33,.33), b ~ N(0,1)     ##
##  a - Truncated Normal: a ~ TNorm(1,1), b ~ N(0,1)       ##
##  1P a - (1,1): a ~ N(1,1), b ~ N(0,1)                   ##
##  1P a - (1,3): a ~ N(1,3), b ~ N(0,1)                   ##
##  1P a - 1: a = 1, b ~ N(0,1)                            ##
#############################################################

# DIC function returns DIC from model specified for Cola and Raisin Bran
#     datasets, (probit and logit), then they are inserted into a storage
#     matrix

DIC <- function(model.file.P,model.file.L){

# Cola, Probit
  sim.c.P <- BRugsFit(modelFile=model.file.P, numChains=1,
      data="examplefilepath\\Cola Data.txt", parametersToSave=
      c("alpha","beta","theta"), nBurnin=1000, nIter=5000, seed=10)
```

```
# Cola, Logit
  sim.c.L <- BRugsFit(modelFile=model.file.L, numChains=1,
      data="examplefilepath\\Cola Data.txt", parametersToSave=
      c("alpha","beta","theta"), nBurnin=1000, nIter=5000, seed=10)
# Raisin Bran, Probit
  sim.rb.P <- BRugsFit(modelFile=model.file.P, numChains=1,
      data="examplefilepath\\Raisin Bran Data.txt", parametersToSave=
      c("alpha","beta","theta"), nBurnin=1000, nIter=5000, seed=10)
# Raisin Bran, Logit
  sim.rb.L <- BRugsFit(modelFile=model.file.L, numChains=1,
      data="examplefilepath\\Raisin Bran Data.txt", parametersToSave=
      c("alpha","beta","theta"), nBurnin=1000, nIter=5000, seed=10)
# Results Storage
  res <- vector(length=4)
  res[1] <- sim.c.P$DIC[2,3]
  res[2] <- sim.c.L$DIC[2,3]
  res[3] <- sim.rb.P$DIC[2,3]
  res[4] <- sim.rb.L$DIC[2,3]
  return(res=res)
}

# creation of matrix for storage of DIC values
DIC.store <- matrix(nrow=24,ncol=4)

# Standard Model
DIC.store[1,] <- DIC(model.file.P="examplefilepath\\Standard Model
      (Probit).txt", model.file.L="examplefilepath\\Standard Model
      (Logit).txt")

# a ~ N(.5,.33)
DIC.store[2,] <- DIC(model.file.P="examplefilepath\\a - (.5,.33) (P).txt",
                     model.file.L="examplefilepath\\a - (.5,.33) (L).txt")

# a ~ N(.5,1)
DIC.store[3,] <- DIC(model.file.P="examplefilepath\\a - (.5,1) (P).txt",
                     model.file.L="examplefilepath\\a - (.5,1) (L).txt")

# a ~ N(.5,3)
DIC.store[4,] <- DIC(model.file.P="examplefilepath\\a - (.5,3) (P).txt",
                     model.file.L="examplefilepath\\a - (.5,3) (L).txt")

# a ~ N(1,.33)
DIC.store[5,] <- DIC(model.file.P="examplefilepath\\a - (1,.33) (P).txt",
                     model.file.L="examplefilepath\\a - (1,.33) (L).txt")
```

```
# a ~ N(1,3)
DIC.store[6,] <- DIC(model.file.P="examplefilepath\\a - (1,3) (P).txt",
                     model.file.L="examplefilepath\\a - (1,3) (L).txt")

# a ~ N(2,.33)
DIC.store[7,] <- DIC(model.file.P="examplefilepath\\a - (2,.33) (P).txt",
                     model.file.L="examplefilepath\\a - (2,.33) (L).txt")

# a ~ N(2,1)
DIC.store[8,] <- DIC(model.file.P="examplefilepath\\a - (2,1) (P).txt",
                     model.file.L="examplefilepath\\a - (2,1) (L).txt")

# a ~ N(2,3)
DIC.store[9,] <- DIC(model.file.P="examplefilepath\\a - (2,3) (P).txt",
                     model.file.L="examplefilepath\\a - (2,3) (L).txt")

# b ~ N(-1,.33)
DIC.store[10,] <- DIC(model.file.P="examplefilepath\\b - (-1,.33) (P).txt",
                      model.file.L="examplefilepath\\b - (-1,.33) (L).txt")

# b ~ N(-1,1)
DIC.store[11,] <- DIC(model.file.P="examplefilepath\\b - (-1,1) (P).txt",
                      model.file.L="examplefilepath\\b - (-1,1) (L).txt")

# b ~ N(-1,3)
DIC.store[12,] <- DIC(model.file.P="examplefilepath\\b - (-1,3) (P).txt",
                      model.file.L="examplefilepath\\b - (-1,3) (L).txt")

# b ~ N(0,.33)
DIC.store[13,] <- DIC(model.file.P=”examplefilepath\\b - (0,.33) (P).txt",
                      model.file.L="examplefilepath\\b - (0,.33) (L).txt")

# b ~ N(0,3)
DIC.store[14,] <- DIC(model.file.P="examplefilepath\\b - (0,3) (P).txt",
                      model.file.L="examplefilepath\\b - (0,3) (L).txt")

# b ~ N(1,.33)
DIC.store[15,] <- DIC(model.file.P="examplefilepath\\b - (1,.33) (P).txt",
                      model.file.L="examplefilepath\\b - (1,.33) (L).txt")

# b ~ N(1,1)
DIC.store[16,] <- DIC(model.file.P="examplefilepath\\b - (1,1) (P).txt",
                      model.file.L="examplefilepath\\b - (1,1) (L).txt")
```

```
# b ~ N(1,3)
DIC.store[17,] <- DIC(model.file.P="examplefilepath\\b - (1,3) (P).txt",
                      model.file.L="examplefilepath\\b - (1,3) (L).txt")

# a ~ gamma(3,3)  (mean,var) = (1,1/3)
DIC.store[18,] <- DIC(model.file.P="examplefilepath\\a - gamma(3,3)
      (P).txt", model.file.L="examplefilepath\\a - gamma(3,3) (L).txt")

# a ~ gamma(1,1)  (mean,var) = (1,1)
DIC.store[19,] <- DIC(model.file.P="examplefilepath\\a - gamma(1,1)
      (P).txt", model.file.L="examplefilepath\\a - gamma(1,1) (L).txt")

# a ~ gamma(.33,.33)  (mean,var) = (1,3)
DIC.store[20,] <- DIC(model.file.P="examplefilepath\\a - gamma(.33,.33)
      (P).txt", model.file.L="examplefilepath\\a - gamma(.33,.33) (L).txt")

# a ~ Truncated Normal
DIC.store[21,] <- DIC(model.file.P="examplefilepath\\a - Truncated Normal
      (P).txt", model.file.L="examplefilepath\\a - Truncated Normal
      (L).txt")

# 1P a - (1,1)
DIC.store[22,] <- DIC(model.file.P="examplefilepath\\1P a - (1,1) (P).txt",
                      model.file.L="examplefilepath\\1P a - (1,1) (L).txt")

# 1P a - (1,3)
DIC.store[23,] <- DIC(model.file.P="examplefilepath\\1P a - (1,3) (P).txt",
                      model.file.L="examplefilepath\\1P a - (1,3) (L).txt")

# 1P a - 1
DIC.store[24,] <- DIC(model.file.P="examplefilepath\\1P a - 1 (P).txt",
                      model.file.L="examplefilepath\\1P a - 1 (L).txt")
```

Appendix C - Additional Code

```
########################################
## Item Response Curves for Figure 1.2 ##
########################################

# Typical Item Response Curve: Figure 1.2 (a)
curve(pnorm(x),from=-3,to=3,xlab="Latent Ability",ylab="P(y=1)")

axis(2,at=seq(0,1,.1))
axis(3,at=seq(-3,3),labels=rep("",7),tck=.02)
axis(4,at=seq(0,1,.1),labels=rep("",11),tck=.02)

# Changing Difficulty, Keeping Discrimination Constant: Figure 1.2 (b)
curve(pnorm(x),from=-3,to=3,xlab="Latent Ability",ylab="P(y=1)")
curve(pnorm(x-1),from=-3,to=3,add=T,lty=5)
curve(pnorm(x+1),from=-3,to=3,add=T,lty=3)

axis(2,at=seq(0,1,.1))
axis(3,at=seq(-3,3),labels=rep("",7),tck=.02)
axis(4,at=seq(0,1,.1),labels=rep("",11),tck=.02)

legend(x=-3,y=.9,legend=c(expression(paste(alpha," = 1, ",beta," = -
      1")),expression(paste(alpha," = 1, ",beta," =
      0")),expression(paste(alpha," = 1, ",beta," =
      1"))),lty=c(3,1,5),bty="n")
text(x=c(-1.2,-.3,.7),y=c(.5,.5,.5),labels=c("Easy","Moderate","Difficult"))

# Changing Discrimination, Keeping Difficulty Constant: Figure 1.2 (c)
curve(pnorm(x),from=-3,to=3,xlab="Latent Ability",ylab="P(y=1)")
curve(pnorm(2*x),from=-3,to=3,add=T,lty=5)
curve(pnorm(.5*x),from=-3,to=3,add=T,lty=3)

axis(2,at=seq(0,1,.1))
axis(3,at=seq(-3,3),labels=rep("",7),tck=.02)
axis(4,at=seq(0,1,.1),labels=rep("",11),tck=.02)

legend(x=-3,y=.9,legend=c(expression(paste(alpha," = .5, ",beta," =
      0")),expression(paste(alpha," = 1, ",beta," =
      0")),expression(paste(alpha," = 2, ",beta," =
      0"))),lty=c(3,1,5),bty="n")
text(x=c(2,1.6,.77),y=c(.8,.89,.88),labels=c("Low","Moderate","High"))
```

```
# Extreme Values of Discrimination: Figure 1.2 (d)
curve(pnorm(0*x),from=-3,to=3,xlab="Latent Ability",ylab="P(y=1)",lty=5,
      ylim=c(0,1))
curve(pnorm(10000*x),from=-3,to=3,add=T)

axis(2,at=seq(0,1,.1))
axis(3,at=seq(-3,3),labels=rep("",7),tck=.02)
axis(4,at=seq(0,1,.1),labels=rep("",11),tck=.02)

legend(x=-3,y=.9,legend=c(expression(paste(alpha," = 0, ",beta," =
      0")),expression(paste(alpha," = 10,000 , ",beta," =
      0"))),lty=c(5,1),bty="n")
text(x=c(1.5,1.5),y=c(.45,.95),labels=c("'Useless'","'Ideal'"))

###################################
## Calculation of Computing Time ##
###################################

# this method returns system, user, and elapsed time for the called functions
# Raisin Bran used as example

# Gibbs Sampler programmed directly in R
ptm <- proc.time()
set.seed(10)
gibbs.rb <- gibbs(rb)
proc.time() - ptm

# Gibbs Sampler using OpenBUGS through 'BRugs' package
ptm <- proc.time()
sim.rb <- BRugsFit(modelFile="examplefilepath\\2-Parameter Probit Item
      Response Model.txt", numChains=1, data="examplefilepath\\Raisin Bran
      Data.txt", parametersToSave=c("alpha","beta","theta","P"),
      nBurnin=1000, nIter=5000, seed=10)
proc.time() - ptm
```