Journal of Statistical Software
August 2010, Volume 36, Code Snippet 1. http://www.jstatsoft.org/

BUGS Code for Item Response Theory

S. McKay Curtis
University of Washington

Abstract

I present **BUGS** code to fit common models from item response theory (IRT), such as the two parameter logistic model, three parameter logisitic model, graded response model, generalized partial credit model, testlet model, and generalized testlet models. I demonstrate how the code in this article can easily be extended to fit more complicated IRT models, when the data at hand require a more sophisticated approach. Specifically, I describe modifications to the **BUGS** code that accommodate longitudinal item response data.

Keywords: education, psychometrics, latent variable model, measurement model, Bayesian inference, Markov chain Monte Carlo, longitudinal data.

1. Introduction

In this paper, I present **BUGS** (Gilks, Thomas, and Spiegelhalter 1994) code to fit several models from item response theory (IRT). Several different software packages are available for fitting IRT models. These programs include packages from Scientific Software International (du Toit 2003), such as **PARSCALE** (Muraki and Bock 2005), **BILOG-MG** (Zimowski, Muraki, Mislevy, and Bock 2005), **MULTILOG** (Thissen, Chen, and Bock 2003), and **TESTFACT** (Wood, Wilson, Gibbons, Schilling, Muraki, and Bock 2003). The Comprehensive R Archive Network (CRAN) task view "Psychometric Models and Methods" (Mair and Hatzinger 2010) contains a description of many different R packages that can be used to fit IRT models in the R computing environment (R Development Core Team 2010). Among these R packages are **ltm** (Rizopoulos 2006) and **gpcm** (Johnson 2007), which contain several functions to fit IRT models using marginal maximum likelihood methods, and **eRm** (Mair and Hatzinger 2007), which contains functions to fit several variations of the Rasch model (Fischer and Molenaar 1995). Volume 20 of the *Journal of Statistical Software* is devoted to "Psychometrics in R" (de Leeuw and Mair 2007) and contains articles on how to fit a multilevel Rasch model with the **lme4** package (Doran, Bates, Bliese, and Dowling 2007; Bates and Maechler 2010), how

to fit multilevel polytomous item response models using Markov chain Monte Carlo (MCMC) methods (Fox 2007), and how to fit item response models that account for response times (Fox, Entink, and van der Linden 2007). The **SCORIGHT** software (Wang, Bradlow, and Wainer 2005) uses MCMC methods to estimate parameters in testlet models.

The use of **BUGS** software to estimate IRT models, however, allows the user to alter existing code to fit new variations of current models that cannot be fit in existing software packages. For example, longitudinal or multilevel data can easily be accommodated by small changes to existing **BUGS** code. The **BUGS** software takes care of the "grunt work" involved in estimating model parameters by constructing an MCMC algorithm to sample from the posterior distribution. As one anonymous reviewer stated it, in **BUGS** "the user does not have to bother thinking about how extensions of a model can be estimated". Thus, using **BUGS** frees the user to experiment with different models that may be more appropriate for specialized data than the models that can currently be fit in other software packages.

Of course, more complicated models involve more parameters than simpler models, and the analyst must specify prior distributions for these new parameters. This is a small price to pay, however, for the flexibility that the Bayesian framework and **BUGS** software provide.

Throughout this article, I assume that the reader has some basic understanding of IRT models and working knowledge of a software implementation of the **BUGS** language. However, if this is not the case, I give some references in Section 2 to sources that discuss IRT models and references to sources that contain tutorials and descriptions of **BUGS**.

Sections 3–8 each start with a brief description of one of the following models:

1. Two parameter logistic model (2PLM, Lord and Novick 1968).
2. Three parameter logistic model (3PLM, Birnbaum 1968).
3. Graded response model (GRM, Samejima 1969).
4. Generalized partial credit model (GPCM, Muraki 1992).
5. Testlet model (Bradlow, Wainer, and Wang 1999).
6. Generalized testlet model (Li, Bolt, and Fu 2006).

I follow each model description with **BUGS** code for fitting each of the models and provide comments on various aspects of the code which may be nonintuitive. In Section 9, I describe how to extend the **BUGS** code of the earlier sections to model item response data in longitudinal studies. I conclude the article with an example of how to use R to call the **BUGS** code for one particular IRT model and how to use the output to check the fit of the model.

2. Preliminaries

2.1. The BUGS language and software

The **BUGS** language is a syntax for defining statistical models. The **BUGS** language is implemented in three software packages: **WinBUGS** (Lunn, Thomas, Best, and Spiegelhalter 2000; Spiegelhalter, Thomas, Best, and Lunn 2003), **OpenBUGS** (Thomas, O'Hara, Ligges,

and Sturtz 2006; Spiegelhalter, Thomas, Best, and Lunn 2010), and **JAGS** (Plummer 2003, 2010a). Each of these software packages can be downloaded for free from their respective websites.

WinBUGS is the oldest of these software packages and is (more or less) frozen in its development. **OpenBUGS** is the open-source version of **WinBUGS** and is actively being developed with new features. Lunn, Spiegelhalter, Thomas, and Best (2009) is a recent article by the creators of the **BUGS** language and **WinBUGS** and **OpenBUGS** software packages that describes the history of the language and other issues in the **BUGS** language. A tutorial on how to use **WinBUGS** and **OpenBUGS** can be obtained in their respective help manuals. These manuals can be accessed via **WinBUGS**'s and **OpenBUGS**'s help menus. A "movie" tutorial for **WinBUGS** and **OpenBUGS** can be found at `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/winbugsthemovie.html`. **WinBUGS** and **OpenBUGS** can be called from within R by using the R packages **R2WinBUGS** (Sturtz, Ligges, and Gelman 2005) and **BRugs** (Thomas *et al.* 2006).

JAGS is an implementation of **BUGS** written and maintained by Martyn Plummer in the C++ programming language. Although very similar to **WinBUGS** and **OpenBUGS**, the **JAGS** implementation of **BUGS** has some nontrivial syntax differences that are described in the **JAGS** manual (see Plummer 2010a, Chapter 8). When appropriate, I note some of these differences as they relate to IRT models.

Unlike **WinBUGS** and **OpenBUGS**, **JAGS** is a command line program, which can make it a little cumbersome to use. However, **JAGS** can be called from R using the R packages **rjags** (Plummer 2010b) and **R2jags** (Su and Yajima 2010).

2.2. Item response theory

I assume that the reader has working knowledge of basic IRT models; however, to establish notation, I briefly discuss each IRT model for which I provide code. The reader is encouraged to consult other sources for more detailed descriptions of the models discussed here. Excellent sources for learning IRT are Baker and Kim (2004), who provide a mathematically detailed introduction to IRT; Hambleton, Swaminathan, and Rogers (1991), who give an intuitive introduction to the topic; and Wainer, Bradlow, and Wang (2007), who provide an introduction to testlet models.

3. Two parameter logistic model

The 2PLM is used for data collected on n individuals who have each given responses on p different items. The items have binary outcomes, i.e., the items are scored as 1 if correct and 0 if not. The i-th individual in the sample is assumed to have a latent ability θ_i, and the i-th individual's response on the $j-th$ item is a random variable Y_{ij} with a Bernoulli distribution. The probability that the $i-th$ individual correctly answers the $j-th$ item (i.e., the probability that $Y_{ij}=1$) is assumed to have the following form

$$p_{ij} = P(Y_{ij} = 1|\theta_i, \alpha_j, \delta_j) = \frac{1}{1+\exp\{-\alpha_j(\theta_i - \delta_j)\}} \tag{1}$$

where α_j is called the discrimination parameter and δ_j is called the difficulty parameter for

```
1  model{
2    for (i in 1:n){
3      for (j in 1:p){
4        Y[i, j] ~ dbern(prob[i, j])
5        logit(prob[i, j]) <- alpha[j] * (theta[i] - delta[j])
6      }
7      theta[i] ~ dnorm(0.0, 1.0)
8    }
9
10   for (j in 1:p){
11     delta[j] ~ dnorm(m.delta, pr.delta)
12     alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
13   }
14   pr.delta <- pow(s.delta, -2)
15   pr.alpha <- pow(s.alpha, -2)
16 }
```

Table 1: Two parameter logistic IRT model.

item j. Note that (1) may be written equivalently

$$\text{logit}(p_{ij}) = \alpha_j(\theta_i - \delta_j), \tag{2}$$

where $\text{logit}(x) = \log\left(\frac{x}{1-x}\right)$.

Each latent ability θ_i is assumed to come from a standard normal distribution. Additionally, in a Bayesian analysis, item parameters are given prior distributions

$$\alpha_j \sim \mathsf{N}_+\left(m_\alpha, s_\alpha^2\right)$$
$$\delta_j \sim \mathsf{N}\left(m_\delta, s_\delta^2\right)$$

where m_α, s_α, m_δ, and s_δ are constants specified before the data analysis, $\mathsf{N}\left(m, s^2\right)$ denotes a normal distribution with mean m and variance s^2, and $\mathsf{N}_+\left(m, s^2\right)$ denotes the normal distribution truncated to the positive real line.

Table 1 contains **BUGS** code to fit the 2PLM. In general, the code speaks for itself; however, I list a few comments below that may clarify some aspects of the code.

- Line 5 uses the `logit` function to specify the logistic ogive as the link function for the model. A normal ogive could be used by changing line 5 to

  ```
  probit(prob[i, j]) <- alpha[j] * (theta[i] - delta[j])
  ```

- The **BUGS** language parametrizes the normal distribution in terms of the precision—the inverse of the variance. Thus, standard deviations for prior distributions on the item parameters need to be converted to precisions in lines 14 and 15.

- A truncated normal distribution is specified for the discrimination parameters in line 12 by using the `I(0, )` operator. Strictly speaking, in **WinBUGS**, there are no truncated

distributions; the `I(.,.)` operator is used only to denote censored observations (Spiegelhalter *et al.* 2003). However, when all parameters in a prior distribution are observed, the `I(.,.)` operator will mimic the behavior of a truncated distribution (Lunn *et al.* 2009, p. 3061).

JAGS and **OpenBUGS** remove this ambiguity between truncation and censoring by introducing the truncation operator `T(.,.)`. The truncation operator, however, is not available for all distributions in **OpenBUGS**, and it is unclear from the **JAGS** manual whether the truncation operator can be used on all distributions implemented in **JAGS**.

OpenBUGS still accepts the `I(.,.)` operator, but **JAGS** does not. Therefore, if the code in Table 1 is to run in **JAGS**, line 12 should be changed to

```
alpha[j] ~ dnorm(m.alpha, pr.alpha) T(0, )
```

- Some authors have used a log-normal distribution as a prior for the discrimination parameters α_j (e.g., Patz and Junker 1999). The log normal distribution can be specified for the discrimination parameters in **BUGS** with the code

  ```
  alpha[j] ~ dlnorm(m.alpha, pr.alpha)
  ```

  Be aware, however, that the log-normal distribution has two parameters and that the mean and variance of the log-normal distribution are functions of *both* parameters. Therefore, changing only one parameter (e.g., `m.alpha` in the previous code) in the log-normal distribution will change both the mean and the variance of the log-normal distribution, which makes prior specification tricky.

- The Rasch model is a special case of the 2PLM where each discrimination parameter `alpha[j]` is set equal to one (see, for example, Baker and Kim 2004, Chapter 5). The **BUGS** code in Table 1 can easily be adapted to fit a Rasch model by changing line 12 to

  ```
  alpha[j] <- 1.0
  ```

  or simply deleting all references to `alpha[]` from the code.

4. Three parameter logistic model

The 3PLM is often used to analyze data from multiple choice tests where subjects try to choose the correct answer from a list of possible answers and may end up choosing the correct answer just by chance. The 3PLM is similar to the 2PLM except that the probability that the i-th individual will respond positively to the j-th item is dependent on a parameter η_j that is constrained to lie in the unit interval:

$$p_{ij} = P(Y_{ij} = 1|\theta_i, \alpha_j, \delta_j) = \eta_j + (1 - \eta_j)\frac{1}{1 + \exp\{-\alpha_j(\theta_i - \delta_j)\}}. \tag{3}$$

The parameter η_j is sometimes called a "guessing" parameter because it represents the probability that an individual of extremely low ability could guess the correct answer on the j-th item just by chance.

```
1  model{
2    for (i in 1:n){
3      for (j in 1:p){
4        Y[i, j] ~ dbern(prob[i, j])
5        logit(prob.star[i, j]) <- alpha[j] * (theta[i] - delta[
             j])
6        prob[i, j] <- eta[j] + (1 - eta[j]) * prob.star[i, j]
7      }
8      theta[i] ~ dnorm(0.0, 1.0)
9    }
10
11   for (j in 1:p){
12     delta[j] ~ dnorm(m.delta, pr.delta)
13     alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
14     eta.star[j] ~ dbeta(a.eta, b.eta)
15     eta[j] <- guess.ind[j] * eta.star[j]
16   }
17   pr.delta <- pow(s.delta, -2)
18   pr.alpha <- pow(s.alpha, -2)
19 }
```

Table 2: Three parameter logistic IRT model.

Equation (3) can also be written as

$$\begin{aligned} \text{logit}(p^*_{ij}) &= \alpha_j(\theta_i - \delta_j) \\ p_{ij} &= \eta_j + (1 - \eta_j)p^*_{ij}. \end{aligned}$$

Not all items are required to have a guessing parameter. For items with no guessing parameter, $\eta_j \equiv 0.0$. For items with a guessing parameter, the η_j parameters are assigned beta prior distributions

$$\eta_j \sim \mathsf{Beta}(a_\eta, b_\eta)\,.$$

All remaining model parameters are assigned priors as in the 2PLM of Section 3.

Table 2 contains **BUGS** code for the 3PLM. Below are some comments on this code.

- The code uses a "guess indicator" vector, `guess.ind[]`, to denote which items have a guessing parameter and which items do not. Element `j` of `guess.ind[]` is `1` if item `j` has a guessing parameter and `0` otherwise.
- The use of the `eta.star[]` and the `guess.ind[]` vectors simplifies the process of specifying which items have guessing parameters and which do not. For items with guessing parameters, `eta.star[j]` is set equal to `eta[j]` in line 15. For items with no guessing parameter, `guess.ind[j]` is equal to zero, which, forces `eta[j]` to be zero in line 15.
- The "long" way to specify which items have guessing parameters and which do not would be to specify each probability in line 6 "by hand" rather than in a loop. For example, if we have a test where items 2 and 4 have a guessing parameter and items 1, 3, and 5 do not, we could use the following code to implement this model

```
for (i in 1:n){
  prob[i, 1] <- prob.star[i, 1]
  prob[i, 2] <- eta[2] + (1 - eta[2]) * prob.star[i, 2]
  prob[i, 3] <- prob.star[i, 3]
  prob[i, 4] <- eta[4] + (1 - eta[4]) * prob.star[i, 4]
  prob[i, 5] <- prob.star[i, 5]
}
```

Specifying the model in the above manner may lead to a slightly more efficient MCMC algorithm, because there is no additional overhead in sampling `eta.star[j]` values for items with no guessing parameter and there is no additional overhead in computing `eta[j]` in line 15 of Table 2. However, specifying each item probability by hand leaves the code more cluttered than in Table 2 and does not allow the code to be easily reused on other data sets.

5. Graded response model

As with the 2PLM and 3PLM, the GRM is used for data collected on n subjects who have responded to each of p items. However, each item can have more than 2 ordered response categories. Thus, the response Y_{ij} of the i-th individual to the j-th item can take values in the set $\{1, \ldots, K_j\}$, where K_j is the largest category of the j-th item. The probability that the i-th subject will select the k-th category on the j-th item is constructed by first considering the cumulative probabilities

$$P_{ijk} = P(Y_{ij} \leq k|\theta_i) = F_L(\kappa_{jk} - \alpha_j\theta_i) \tag{4}$$

where κ_{jk} is a threshold, and $F_L(\cdot)$ is the CDF of the logistic distribution. Each item has $K_j - 1$ thresholds $\kappa_{j1}, \ldots, \kappa_{j,K_j-1}$ that must satisfy the order constraint $\kappa_{j1} < \cdots < \kappa_{j,K_j-1}$. The probability p_{ijk} that the i-th subject will select the k-th category on item j can now be written as

$$\begin{aligned} p_{ij1} &= P_{ij1} \\ p_{ijk} &= P_{ijk} - P_{i,j,k-1} \quad \text{for } k = 2, \ldots, K_j - 1 \\ p_{ijK_j} &= 1 - P_{i,j,K_j-1}. \end{aligned}$$

Priors on item parameters α_j and δ_j are the same as the priors for the 2PLM. The priors for the threshold parameters must account for the order constraint $\kappa_{j1} < \cdots < \kappa_{j,K_j-1}$. A prior on the threshold parameters can be induced by defining unconstrained auxiliary parameters $\kappa^*_{j1}, \ldots, \kappa^*_{j,K_j-1}$ such that

$$\kappa^*_{jk} \sim \mathsf{N}\big(m_\kappa, s^2_\kappa\big)$$

for $k = 1, \ldots, K_j - 1$. Prior distributions on the thresholds for the j-th item are obtained by setting κ_{jk} equal to the k-th order statistic of the auxiliary variables $\kappa^*_{j,1}, \ldots, \kappa^*_{j,K_j-1}$ for the

```
model{
  for (i in 1:n){
    for (j in 1:p){
      Y[i, j] ~ dcat(prob[i, j, 1:K[j]])
    }
    theta[i] ~ dnorm(0.0, 1.0)

    for (j in 1:p){
      for (k in 1:(K[j]-1)){
        logit(P[i, j, k]) <- kappa[j, k] - alpha[j] *
              theta[i]
      }
      P[i, j, K[j]] <- 1.0
    }

    for (j in 1:p){
      prob[i, j, 1] <- P[i, j, 1]
      for (k in 2:K[j]){
        prob[i, j, k] <- P[i, j, k] - P[i, j, k-1]
      }
    }
  }

  for (j in 1:p){
    alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
  }
  pr.alpha <- pow(s.alpha, -2)

  for (j in 1:p){
    for (k in 1:(K[j]-1)){
      kappa.star[j, k] ~ dnorm(m.kappa, pr.kappa)
      kappa[j, k] <- ranked(kappa.star[j, 1:(K[j]-1)], k)
    }
  }
  pr.kappa <- pow(s.kappa, -2)
}
```

Table 3: Graded response model.

j-th item, that is

$$\begin{aligned}\kappa_{j1} &= \kappa^*_{j,[1]}\\ \kappa_{j2} &= \kappa^*_{j,[2]}\\ &\vdots\\ \kappa_{j,K_j-1} &= \kappa^*_{j,[K_j-1]}\end{aligned}$$

where $\kappa_{j,[k]}$ denotes the k-th order statistic of $\kappa^*_{j1}, \ldots, \kappa^*_{j,K_j-1}$. This approach to modeling thresholds is recommended by Plummer (2010a, p. 36).

Table 3 contains **BUGS** code to fit the GRM. Below are some comments on the code.

- Because the responses are no longer have binary, the `dcat` function must be used to specify the distribution of the data. The `dcat` function defines a categorical distribution with more than two categories. The `dcat` function requires that the data for item `j` are coded as `1, 2, ..., K[j]`, i.e., the data cannot contain any zeros.

- The `ranked` function on line 31 returns the `k`-th smallest value in the vector `kappa.star[j, 1:(K[j]-1)]`.

- Some tests may not have the same numbers of categories for all items. Therefore, not all values in the matrix `kappa[,]` will be defined. **WinBUGS** and **OpenBUGS** run without any problems with these undefined parameters in the GRM. However, **JAGS** crashes when it tries to set monitors for the defined `kappa` parameters in the presence of undefined `kappa` parameters. The GRM can be fit in **JAGS** by setting the undefined item-step parameters to some arbitrary fixed value (e.g., zero) by passing those values to **JAGS** as data. For example, if the numbers of categories for a 5 item test are 2, 3, 4, 4, and 4, then a 5×3 matrix of the following format

```
NA    0    0
NA   NA    0
NA   NA   NA
NA   NA   NA
NA   NA   NA
```

  should be passed to **JAGS** as data for the matrix `kappa`.

- The **JAGS** implementation of **BUGS** does not have the `ranked` function used on line 31; instead, **JAGS** has a `sort` function. The `sort` function in **JAGS** takes a vector as its input and returns the vector sorted in ascending order. Thus, lines 28–33 in Table 3 should be replaced with

```
for (j in 1:p){
  for (k in 1:(K[j]-1)){
    kappa.star[j, k] ~ dnorm(m.kappa, pr.kappa)
  }
  kappa[j, 1:(K[j]-1)] <- sort(kappa.star[j, 1:(K[j]-1)])
}
```

  for the code to work in **JAGS**.

6. Generalized partial credit model

The GPCM (Muraki 1992) is an alternative to the GRM for ordinal item responses. In the GPCM, the probability that the i-th individual selects category k on item j is

$$p_{ijk} = P(Y_{ij} = k|\theta_i) = \frac{\exp\left\{\sum_{\ell=1}^{k} \alpha_j(\theta_i - \beta_{j\ell})\right\}}{\sum_{m=1}^{K_j} \exp\left\{\sum_{\ell=1}^{m} \alpha_j(\theta_i - \beta_{j\ell})\right\}},$$

where $\beta_{j1} \equiv 0$ for all j. The parameters β_{jk} are often called "item-step" parameters. Unlike the threshold parameters in the GRM, the item-step parameters do not need to satisfy any order restrictions and can be given normal prior distributions with no additional constraints

$$\beta_{j\ell} \sim \mathsf{N}\left(m_\beta, s_\beta^2\right)$$

for $j = 1, \ldots, p$ and $\ell = 2, \ldots, K_j$.

Even though, mathematically, the item-step parameters do not need to satisfy order constraints, if parameter estimates of item-step parameters for a certain item do not satisfy the ordering $\beta_{j1} < \cdots < \beta_{jK_j}$, then it might be more appropriate to model the offending item with less categories (see, for example, Reckase 2009, page 34).

The discrimination parameters α_j are assigned a truncated normal prior distribution as in the previous models. As usual, the latent abilities θ_i are assumed to follow a standard normal distribution.

Table 4 contains **BUGS** code for the GPCM. Below are some comments on this code.

- As with the threshold parameters in the GRM, not all of the item-step parameters are defined when the items have differing numbers of response categories. **WinBUGS** and **OpenBUGS** run without problems in spite of the undefined parameters. **JAGS**, however, crashes with an error when running this model. The GPCM can be fit in **JAGS** by setting the undefined item-step parameters to some arbitrary fixed value (e.g., zero) by passing those values to **JAGS** as data. For example, if the data contain 5 items with numbers of categories 2, 2, 3, 3, and 4, then a 5×4 matrix with the following structure

  ```
  NA   NA    0    0
  NA   NA    0    0
  NA   NA   NA    0
  NA   NA   NA    0
  NA   NA   NA   NA
  ```

  should be passed to **JAGS** as data for the matrix `beta[,]`.

- The partial credit model (Masters 1982, PCM) is a special case of the GPCM where each discrimination parameter `alpha[j]` is set equal to one. The code in Table 4 can be adapted to fit the PCM by changing line 21 to read

  ```
  alpha[j] <- 1.0
  ```

  or by removing all references to `alpha[]` from the code.

```
model{
  for (i in 1:n){
    for (j in 1:p){
      Y[i, j] ~ dcat(prob[i, j, 1:K[j]])
    }
    theta[i] ~ dnorm(0.0, 1.0)
  }

  for (i in 1:n){
    for (j in 1:p){
      for (k in 1:K[j]){
        eta[i, j, k] <- alpha[j] * (theta[i] - beta[j, k])
        psum[i, j, k] <- sum(eta[i, j, 1:k])
        exp.psum[i, j, k] <- exp(psum[i, j, k])
        prob[i, j, k] <- exp.psum[i, j, k] / sum(exp.psum[i,
           j, 1:K[j]])
      }
    }
  }

  for (j in 1:p){
    alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
    beta[j, 1] <- 0.0
    for (k in 2:K[j]){
      beta[j, k] ~ dnorm(m.beta, pr.beta)
    }
  }
  pr.alpha <- pow(s.alpha, -2)
  pr.beta <- pow(s.beta, -2)
}
```

Table 4: Generalized partial credit model.

7. Testlet model

The testlet model is used for tests that are structured into groups of items that share some common feature. These groups of items are called "testlets". For example, many tests require a subject to read a certain passage and then answer two or more questions about the passage. Responses to items from the same testlet will tend to be more highly correlated than items from different testlets after accounting for the latent ability.

In a testlet model for binary responses, the probability that the i-th subject answers the j-th item correctly is assumed to have the following form

$$p_{ij} = P(Y_{ij} = 1|\theta_i) = \frac{1}{1 + \exp\left\{-\alpha_j(\theta_i - \delta_j + \gamma_{id(j)})\right\}}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, p$.

Each subject has n_T testlet effects $\gamma_{id(j)}$, which can be thought of as testlet-specific abilities. The testlet effects explicitly model the correlation among items in a testlet after accounting for the latent ability. The function $d(\cdot)$ maps values in $\{1, \ldots, p\}$ to $\{0, 1, \ldots, n_T\}$. In other words, the function $d(\cdot)$ denotes which items belong to which testlets, so if $d(5) = 3$, then the 5-th item belongs to the 3rd testlet. When $d(j) = 0$, the j-th item does not belong to any testlet and, therefore, has a testlet effect equal to zero for all subjects, i.e., $\gamma_{i0} \equiv 0$ for $i = 1, \ldots, n$.

Testlet-effect parameters are assumed to come from normal distributions, and each testlet is given its own testlet-specific variance $\sigma^2_{d(j)}$

$$\gamma_{id(j)} \sim \mathsf{N}\left(0, \sigma^2_{d(j)}\right)$$

The testlet variances are assigned inverse-gamma prior distributions

$$\sigma^2_{d(j)} \sim \mathsf{InvGam}\left(a_{\sigma^2_\gamma}, b_{\sigma^2_\gamma}\right).$$

Priors for all other model parameters are assigned as in the 2PLM.

Table 5 contains **BUGS** code for the testlet model. Below are some comments on this code.

- As with the **BUGS** code for other models, the terms `n` and `p` denote the number of subjects and items, respectively, and must be passed to **BUGS** as data. Additionally, the user must also specify the number of testlets `n.t` and the testlet-identifier vector `d[]`.
- The specification of `d[j]` in the **BUGS** code differs somewhat from the mathematical formulation outlined in the beginning of this section. **BUGS** does not allow subscripts equal to 0, thus, we cannot use `gamma[i, 0] <- 0.0` to set $\gamma_{i0} \equiv 0$. Instead, we let `gamma[i, n.t + 1] <- 0.0`. Users must account for this specification in their data by assigning a value of `n.t + 1` to `d[j]` if the `j`-th item does not belong to a testlet. For example, in a ten item test, if items 1–4 are part of the first testlet, item 5 is not a part of any testlet, items 6–9 are part of the second testlet, and item 10 is not a part of any testlet, then `n.t=2` and the vector `d[]` should be

  ```
  d = c(1, 1, 1, 1, 3, 2, 2, 2, 2, 3)
  ```

- The **BUGS** language has no direct method of specifying an inverse-gamma prior distribution. The typical method of specifying an inverse-gamma prior for a parameter is to specify a gamma prior on the inverse of the parameter. This "trick" is used in lines 24 and 25 to specify an inverse-gamma prior on the `sigsq.gamma[k]` parameters.
- Inverse-gamma priors have typically been used on variance parameters for reasons of convenience. The inverse-gamma prior is the conjugate prior for a variance parameter from a normal likelihood, so the update in an MCMC algorithm is a simple random draw from an inverse-gamma distribution. However, since **BUGS** frees the user from having to directly code the MCMC algorithm, other priors for variance parameters may be preferable. Gelman and Hill (2007) use diffuse uniform priors on *standard deviations* in many of the examples in their text book. To implement this approach in the testlet model, users may simply replace lines 24 and 25 with

```
model{
  for (i in 1:n){
    for (j in 1:p){
      Y[i, j] ~ dbern(prob[i, j])
      logit(prob[i, j]) <- alpha[j] * (theta[i] - delta[j] +
          gamma[i,d[j]])
    }

    theta[i] ~ dnorm(0.0, 1.0)

    for (k in 1:n.t){
      gamma[i, k] ~ dnorm(0.0, pr.gamma[k])
    }
    gamma[i, n.t + 1] <- 0.0
  }

  for (j in 1:p){
    alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0.0, )
    delta[j] ~ dnorm(m.delta, pr.delta)
  }
  pr.alpha <- pow(s.alpha, -2)
  pr.delta <- pow(s.delta, -2)

  for (k in 1:n.t){
    pr.gamma[k] ~ dgamma(a.sigsq.gamma, b.sigsq.gamma)
    sigsq.gamma[k] <- 1.0/pr.gamma[k]
  }
}
```

Table 5: Testlet model.

```
sigma.gamma[k] ~ dunif(0, 100)
pr.gamma[k] <- pow(sigma.gamma[k], -2)
```

See Gelman (2006) for further discussion on priors for variance parameters in hierarchical models.

- In this section, I have defined the testlet model only for binary responses. However, the code in Table 5 can be adapted for other models (e.g., the GRM) by combining the code in Table 5 with code for models in the earlier sections. As mentioned previously, this ability to quickly adapt **BUGS** for new models is one of the major advantages to using **BUGS**.

8. Generalized testlet model

In the testlet model, the testlet effects $\gamma_{id(j)}$ are forced to have the same discrimination parameter α_j as the latent ability. The generalized testlet model relaxes this restriction by introducing p new discrimination parameters $\alpha_{21}, \ldots, \alpha_{2p}$ for the testlet effects. In the generalized testlet model, the probability that the i-th subject correctly answers the j-th item is modeled as

$$p_{ij} = P(Y_{ij} = 1|\theta_i) = \frac{1}{1 + \exp\left\{-(\alpha_{1j}\theta_i - \zeta_j + \alpha_{2j}\gamma_{id(j)})\right\}}.$$

Both sets of discrimination parameters are given truncated normal prior distributions

$$\alpha_{1j} \sim \mathsf{N}_+\left(m_{\alpha_1}, s^2_{\alpha_1}\right)$$
$$\alpha_{2j} \sim \mathsf{N}_+\left(m_{\alpha_2}, s^2_{\alpha_2}\right),$$

and the difficulty parameters ζ_j are given normal prior distributions

$$\zeta_j \sim \mathsf{N}\left(m_\zeta, s^2_\zeta\right).$$

```
model{
  for (i in 1:n){
    for (j in 1:p){
      Y[i, j] ~ dbern(prob[i, j])
      logit(prob[i, j]) <- alpha1[j] * theta[i] - zeta[j] +
         alpha2[j] * gamma[i,d[j]]
    }

    theta[i] ~ dnorm(0.0, 1.0)

    for (k in 1:n.t){
      gamma[i, k] ~ dnorm(0.0, 1.0)
    }
    gamma[i, n.t + 1] <- 0.0
  }

  for (j in 1:p){
    alpha1[j] ~ dnorm(m.alpha1, pr.alpha1) I(0.0, )
    alpha2[j] ~ dnorm(m.alpha2, pr.alpha2) I(0.0, )
    zeta[j] ~ dnorm(m.zeta, pr.zeta)
  }
  pr.alpha1 <- pow(s.alpha1, -2)
  pr.alpha2 <- pow(s.alpha2, -2)
  pr.zeta <- pow(s.zeta, -2)
}
```

Table 6: Generalized testlet model.

The ability parameters are assumed to come from a standard normal distribution. As in the testlet model, the testlet effects $\gamma_{d(j)}$ are assumed to come from normal distributions. However, for identifiability, the testlet effects are no longer permitted to have testlet specific variances. The variances of the testlet effects are restricted to be one

$$\gamma_{id(j)} \sim \mathsf{N}(0, 1)\,.$$

Table 6 contains **BUGS** code for the generalized testlet model. The code uses the same "trick" to define a zero testlet effect as in the testlet model, so the comments of Section 7 apply to the code here as well.

9. Extending the BUGS code

A major advantage of using **BUGS** is that any basic model can be easily extended by changing or adding only a few lines of **BUGS** code in existing model files. The **BUGS** software then takes care of the details of the estimation procedure for the parameters of the new model. I demonstrate this point in this section by extending the 2PLM to the longitudinal setting.

In many studies, subjects are administered the same set of test items on multiple occasions. Thus, instead of having a single vector of responses, the i-th individual has a matrix of responses composed of T response vectors

$$\begin{aligned}\mathbf{Y}_i &= \begin{pmatrix}\mathbf{Y}_{i1} & \mathbf{Y}_{i2} & \cdots & \mathbf{Y}_{iT}\end{pmatrix} \\ &= \begin{pmatrix} Y_{i11} & Y_{i12} & \cdots & Y_{i1T} \\ Y_{i21} & Y_{i22} & \cdots & Y_{i2T} \\ \vdots & \vdots & \ddots & \vdots \\ Y_{ip1} & Y_{ip2} & \cdots & Y_{ipT} \end{pmatrix}\end{aligned}$$

one response vector $\mathbf{Y}_{it}$ at each time point t, and an ability vector

$$\boldsymbol{\theta}_i = (\theta_{i1}, \ldots, \theta_{iT})'$$

where parameter θ_{it} is the ability of the i-th subject at time t.

Because the ability parameters $\theta_{i1}, \ldots, \theta_{iT}$ are measured on the same subject over a period of time, latent abilities on the same subject will be more correlated than latent abilities among different subjects. A statistical model for this scenario should account for this correlation structure.

9.1. Population-averaged covariance models

In longitudinal data analysis, there are two main modeling strategies for modeling the covariance of repeated observations (Davidian 2005). The first strategy is called "population averaged", which, in the context of item response theory, involves assuming a covariance structure directly on the population distribution of all ability vectors $\boldsymbol{\theta}_i$. Ability vectors $\boldsymbol{\theta}_i$ are assumed to come from a normal distribution

$$\boldsymbol{\theta}_i \sim \mathsf{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$$

with some mean structure $\boldsymbol{\mu}_\theta$ and covariance $\boldsymbol{\Sigma}_\theta$. Popular structures for the covariance are the "uniform" or "compound symmetric" covariance

$$\begin{aligned}\boldsymbol{\Sigma} &= (\sigma^2\rho) \\ &= \sigma^2\mathbf{R} \\ &= \sigma^2\begin{pmatrix} 1 & \rho & \rho & \cdots & \rho \\ \rho & 1 & \rho & \cdots & \rho \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \rho & \cdots & 1 \end{pmatrix}\end{aligned}$$

the AR(1) structure

$$\begin{aligned}\boldsymbol{\Sigma} &= \sigma^2\mathbf{R} \\ &= (\sigma^2\rho^{|i-j|}) \\ &= \sigma^2\begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{T-1} \\ \rho & 1 & \rho & \cdots & \rho^{T-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{T-1} & \rho^{T-2} & \rho^{T-3} & \cdots & 1 \end{pmatrix}\end{aligned}$$

and the Markov structure for unequally spaced time points

$$\begin{aligned}\boldsymbol{\Sigma} &= (\sigma^2\rho^{d_{ij}}) \\ &= \sigma^2\begin{pmatrix} 1 & \rho^{d_{12}} & \rho^{d_{13}} & \cdots & \rho^{d_{1T}} \\ \rho^{d_{21}} & 1 & \rho^{d_{23}} & \cdots & \rho^{d_{2T}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{d_{T1}} & \rho^{d_{T2}} & \rho^{d_{T3}} & \cdots & 1 \end{pmatrix}\end{aligned}$$

where $\sigma^2 > 0$, $-1 < \rho < 1$, $d_{ij} = |t_i - t_j|$, and t_k is the time at the k-th observation.

Each of the above models implies a constant variance of the latent ability across time. Less restrictive versions of each of the above covariance models can be obtained by allowing the variance to be different at each time point. These "heterogeneous" covariance structures can be modeled by pre and post multiplying the correlation matrices $\mathbf{R}$ in the above models by a diagonal matrix

$$\mathbf{D}_\theta = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_T \end{pmatrix}$$

with unique standard deviations along the diagonal.

For identifiability, the mean and variance at one time point must be specified to establish the location and scale of the latent ability distribution (Tavares and Andrade 2006, pp. 105, 106). Setting the first element of $\boldsymbol{\mu}_\theta$ to zero and the first variance σ_1^2 in $\boldsymbol{\Sigma}_\theta$ to one is one possibility.

BUGS code for AR(1) models

Table 7 contains **BUGS** code for the 2PLM, where the code has been extended to account for longitudinal data with an AR(1) covariance structure with constant variance. Below are some comments on this code.

- The code for the likelihood portion of the model now has an extra loop indexed by time `t`, where the quantities `Y[, , ]`, `prob[, , ]`, and `theta[,]` are now indexed by this third subscript.
- Analogous to the univariate normal distribution, the multivariate normal distribution in **BUGS** is parametrized in terms of the precision matrix, which is the inverse of the covariance matrix. Therefore, the covariance matrix `Sigma.theta[,]` must be inverted in line 30. The resulting precision matrix `Pr.theta[,]` is then used in the `dmnorm` function.
- The syntax for matrix operations is more flexible in **JAGS** than in **WinBUGS** or **OpenBUGS**. For example, line 30 in Table 7 can be replaced with the following cleaner code in **JAGS**

  ```
  Pr.theta <- inverse(Sigma.theta)
  ```

- Because the variance of the latent ability is restricted to be constant across time and the variance must be specified exactly for at least one time period to establish identifiability, `sigsq.theta` is set equal to `1.0` in line 21.
- **WinBUGS** and **OpenBUGS** do not experience problems when fitting the AR(1) model. However, the default MCMC algorithm constructed by **JAGS** does not reach convergence.

Table 8 contains **BUGS** code for the AR(1) covariance structure with heterogeneous variances. Below are some comments on this code. The only major substantive difference between the code in Table 7 and in Table 8 is that `sigsq.theta[]` is now a vector where the first element of the vector is set to one and the remaining elements are given gamma prior distributions. There is also a slight difference in how the heterogeneous AR(1) structure is constructed (see line 27 of Table 8) relative to the AR(1) structure with constant variance (see line 26 in Table 7). Line 27 of Table 8 implements the pre and post multiplication of the correlation matrix $\mathbf{R}$ by the standard deviation matrix $\mathbf{D}_\theta$.

BUGS code for unstructured covariance

Researchers may be reluctant to specify a particular structure for the covariance matrix of the latent abilities. Wishart distributions are typically used to specify priors for unstructured covariance matrices. However, a Wishart prior cannot be used for the covariance of latent abilities, because at least one variance must be set to a constant for identifiability.

One approach to solve this problem is to parametrize the covariance matrix $\mathbf{\Sigma}_\theta$ in terms of its Cholesky decomposition

$$\mathbf{\Sigma}_\theta = \mathbf{L}_\theta \mathbf{L}'_\theta \tag{5}$$

where $\mathbf{L}_\theta$ is a lower triangular matrix with positive entries on the diagonal and unrestricted entries below the diagonal. Setting the first element of $\mathbf{L}_\theta$ equal to one ensures that the first

```
model{
  for (t in 1:T){
    for (i in 1:n){
      for (j in 1:p){
        Y[i, j, t] ~ dbern(prob[i, j, t])
        logit(prob[i, j, t]) <- alpha[j] * (theta[i, t] -
           delta[j])
      }
    }
  }

  for (i in 1:n){
    theta[i, 1:T] ~ dmnorm(mu.theta[], Pr.theta[,])
  }

  mu.theta[1] <- 0.0
  for (t in 2:T){
    mu.theta[t] ~ dnorm(m.mu.theta, pr.mu.theta)
  }
  pr.mu.theta <- pow(s.mu.theta, -2)

  sigsq.theta <- 1.0
  Sigma.theta[1, 1] <- sigsq.theta
  for (i in 2:T){
    Sigma.theta[i,i] <- sigsq.theta
    for (j in 1:(i-1)){
      Sigma.theta[i, j] <- sigsq.theta * pow(rho, i - j)
      Sigma.theta[j, i] <- Sigma.theta[i, j]
    }
  }
  Pr.theta[1:T, 1:T] <- inverse(Sigma.theta[,])
  rho ~ dunif(-1.0, 1.0)

  for (j in 1:p){
    alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
    delta[j] ~ dnorm(m.delta, pr.delta)
  }
  pr.alpha <- pow(s.alpha, -2)
  pr.delta <- pow(s.delta, -2)
}
```

Table 7: Longitudinal two parameter logistic model with AR(1) covariance structure.

```
model{
  for (t in 1:T){
    for (i in 1:n){
      for (j in 1:p){
        Y[i, j, t] ~ dbern(prob[i, j, t])
        logit(prob[i, j, t]) <- alpha[j] * (theta[i, t] -
           delta[j])
      }
    }
  }

  for (i in 1:n){
    theta[i, 1:T] ~ dmnorm(mu.theta[], Pr.theta[,])
  }

  mu.theta[1] <- 0.0
  for (t in 2:T){
    mu.theta[t] ~ dnorm(m.mu.theta, pr.mu.theta)
  }
  pr.mu.theta <- pow(s.mu.theta, -2)

  sigsq.theta[1] <- 1.0
  Sigma.theta[1, 1] <- 1.0
  for (i in 2:T){
    sigsq.theta[i] ~ dgamma(a.sigsq.theta, b.sigsq.theta)
    Sigma.theta[i, i] <- sigsq.theta[i]
    for (j in 1:(i-1)){
      Sigma.theta[i, j] <- sqrt(sigsq.theta[i]) * sqrt(sigsq.
         theta[j]) * pow(rho, i - j)
      Sigma.theta[j, i] <- Sigma.theta[i, j]
    }
  }
  Pr.theta[1:T, 1:T] <- inverse(Sigma.theta[,])
  rho ~ dunif(-1.0, 1.0)

  for (j in 1:p){
    alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
    delta[j] ~ dnorm(m.delta, pr.delta)
  }
  pr.alpha <- pow(s.alpha, -2)
  pr.delta <- pow(s.delta, -2)
}
```

Table 8: Longitudinal two parameter logistic regression model with heterogeneous AR(1) covariance structure.

element of $\mathbf{\Sigma}_\theta$ is also equal to one. Putting priors on the elements of $\mathbf{L}_\theta$ is straightforward, because the only elements that must satisfy restrictions are the diagonal elements, which need only be positive.

Table 9 contains **BUGS** code for fitting a model with an unstructured covariance for the population distribution of the ability vectors $\boldsymbol{\theta}_i$. Below are some comments on this code.

- Gamma priors are used to restrict the diagonal elements of `L.theta` to be positive, but other similar priors may be used (e.g., `dlnorm`).
- **WinBUGS** and **OpenBUGS** do not have a built-in matrix multiplication function. Therefore, the matrix multiplication in Equation (5) is performed explicitly using loops and the `inprod` function. The `inprod` function computes the inner product of two vectors.
- **JAGS** offers more support for vector and matrix calculations than **WinBUGS** and **OpenBUGS**. In **JAGS**, lines 29–33 can be replaced by the single line

  ```
  Sigma.theta <- L.theta %*% t(L.theta)
  ```

- In general, the mixing of the Markov chains for this model is poor in all of the **BUGS** packages, and the chains must be run for a long time (several hundred thousand iterations) to achieve a good sample from the posterior.

9.2. Subject-specific covariance models

The other modeling strategy for longitudinal data is sometimes called "subject specific", which, in the context of item response theory, involves assuming that the ability θ_{it} at time t is some function (usually a linear combination) of random coefficients and time. For example, one subject-specific model is a linear function over time

$$\theta_{it} = \gamma_i^{(0)} + \gamma_i^{(1)} t$$

where

$$\begin{aligned} \gamma_i^{(0)} &\sim \mathsf{N}\left(\mu_{\gamma_0}, \sigma^2_{\gamma_0}\right) \\ \gamma_i^{(1)} &\sim \mathsf{N}\left(\mu_{\gamma_1}, \sigma^2_{\gamma_1}\right). \end{aligned}$$

In this model, each individual in the data is assumed to have two "random coefficients" that determine the individual's linear trajectory of ability over time.

Like the population averaged models, the subject-specific models must incorporate restrictions on the location and scale of the latent abilities to establish identifiability. In population-averaged models, the mean and variance of the ability population at some time point t can be set to constants—usually zero and one, respectively. In subject-specific models, the mean and variance of the population of one of the random coefficients can be set to constants. For example, in the linear, subject-specific model specified above, setting $\mu_{\gamma_0} = 0$ and $\sigma^2_{\gamma_0} = 1$ establishes the location and scale of the latent abilities by setting the distribution of the abilities at $t = 0$ to be equal to the standard normal distribution.

```
model{
  for (t in 1:T){
    for (i in 1:n){
      for (j in 1:p){
        Y[i, j, t] ~ dbern(prob[i, j, t])
        logit(prob[i, j, t]) <- alpha[j] * (theta[i, t] -
           delta[j])
      }
    }
  }

  for (i in 1:n){
    theta[i, 1:T] ~ dmnorm(mu.theta[], Pr.theta[,])
  }

  mu.theta[1] <- 0.0
  for (t in 2:T){
    mu.theta[t] ~ dnorm(m.mu.theta, pr.mu.theta)
  }
  pr.mu.theta <- pow(s.mu.theta, -2)

  L.theta[1, 1] <- 1.0
  for (i in 2:T){
    L.theta[i, i] ~ dgamma(a.L.theta, b.L.theta)
    for (j in 1:(i-1)){
      L.theta[i, j] ~ dnorm(m.L.theta, s.L.theta)
      L.theta[j, i] <- 0.0
    }
  }
  for (i in 1:T){
    for (j in 1:T){
      Sigma.theta[i, j] <- inprod(L.theta[i, 1:T], L.theta[j
         , 1:T])
    }
  }
  Pr.theta[1:T, 1:T] <- inverse(Sigma.theta[,])

  for (j in 1:p){
    alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
    delta[j] ~ dnorm(m.delta, pr.delta)
  }
  pr.alpha <- pow(s.alpha, -2)
  pr.delta <- pow(s.delta, -2)
}
```

Table 9: Longitudinal two parameter logistic model with unstructured covariance.

```
1  model{
2    for (t in 1:T){
3      for (i in 1:n){
4        for (j in 1:p){
5          Y[i, j, t] ~ dbern(prob[i, j, t])
6          logit(prob[i, j, t]) <- alpha[j] * (theta[i, t] -
               delta[j])
7        }
8      }
9    }
10
11   for (i in 1:n){
12     for (t in 1:T){
13       theta[i, t] <- gamma0[i] + gamma1[i] * (t-1)
14     }
15   }
16
17   for (i in 1:n){
18     gamma0[i] ~ dnorm(0.0, 1.0)
19     gamma1[i] ~ dnorm(mu.gamma1, pr.gamma1)
20   }
21   mu.gamma1 ~ dnorm(m.mu.gamma1, pr.mu.gamma1)
22   pr.gamma1 ~ dgamma(a.pr.gamma1, b.pr.gamma1)
23   sigsq.gamma1 <- 1.0/pr.gamma1
24   pr.mu.gamma1 <- pow(s.mu.gamma1, -2)
25
26   for (j in 1:p){
27     alpha[j] ~ dnorm(m.alpha, pr.alpha) I(0, )
28     delta[j] ~ dnorm(m.delta, pr.delta)
29   }
30   pr.alpha <- pow(s.alpha, -2)
31   pr.delta <- pow(s.delta, -2)
32 }
```

Table 10: Longitudinal two parameter logistic IRT model with linear random effects.

Table 10 contains **BUGS** code to fit the subject-specific, longitudinal 2PLM with random coefficients. In line 13 of the code I have used `(t-1)` to force the first time point to be zero. This helps in the interpretation of model parameters because under this parametrization the intercept random effects `gamma0[i]` are the abilities at the first time point.

10. Example

In this section I demonstrate how to use the **BUGS** code in the previous sections to analyze a real data set. I use the `Environment` data set from the R package **ltm**. The data are a subset of responses from the 1990 British Social Attitudes Survey and contain responses of

291 individuals to six questions involving environmental issues. The possible answers to each question are "not very concerned", "slightly concerned", and "very concerned". The ordinal nature of the responses implies that the GRM or the GPCM should be used to analyze these data. In what follows, I use the GRM.

It is often convenient to call **WinBUGS** (or **OpenBUGS**, or **JAGS**) from some other data-processing software as this allows the user to compute posterior summaries and create plots of posterior draws that are not available directly in **BUGS** software. For this example, I use the `openbugs` function in the **R2WinBUGS** package to call **OpenBUGS** from R.

In the data analysis of this section, I follow the approach of Rizopoulos (2006) by fitting two versions of the GRM—a constrained model, where the discrimination parameters are forced to be equal ($\alpha_1 = \cdots = \alpha_6 = \alpha$) for each of the six survey questions, and an unconstrained model, where all six discrimination parameters are freely estimated. The constrained model can be fit by removing the subscript `[j]` from the parameter `alpha` in line 10 of Table 3 as in

```
logit(P[i, j, k]) <- kappa[j, k] - alpha * theta[j]
```

and removing the prior on line 24 of Table 3 from the loop and the subscript `[j]` from the parameter `alpha` as in

```
alpha ~ dnorm(m.alpha, pr.alpha) I(0, )
```

The relevant R packages and the data can be loaded in the R session with the following code:

```
R> library("R2WinBUGS")
R> library("ltm")
R> library("mcmcplots")
R> data("Environment")
```

The data set `Environment` is a `data.frame` with six variables each of class `factor`. Because the data are inherently ordinal, the variables in the `Environment` data set must first be converted to class `ordered` by calling the `ordered` function on each variable and setting the `levels` attribute. This can be done by using the `lapply` function to apply the `ordered` function with argument

```
levels = c("not very concerned", "slightly concerned", "very concerned")
```

to each of the variables in `Environment`.

Additionally, in order to be processed by **OpenBUGS**, the data must be coerced to a `matrix` of numeric values that correspond to each level of the ordinal response:

- 1 → "not very concerned"
- 2 → "slightly concerned"
- 3 → "very concerned"

The above operations can be executed in one line of R code:

```
R> Y <- data.matrix(data.frame(lapply(Environment, ordered, levels =
+    c("not very concerned", "slightly concerned", "very concerned"))))
```

Several constants—sample size (`n`), number of survey questions (`p`), number of response categories per question (`K`), and parameters for prior distributions (`m.alpha`, `s.alpha`, `m.kappa`, and `s.kappa`)—need to be passed to **OpenBUGS** as data. These constants are first stored as variables in the R session, then the names of these variables and the data matrix (`Y`) are stored in the character vector `data`, which will be passed to the `openbugs` function.

```
R> p <- ncol(Y)
R> n <- nrow(Y)
R> m.alpha <- 1.0
R> s.alpha <- 2.5
R> m.kappa <- 0.0
R> s.kappa <- 2.5
R> K <- apply(Y, 2, max)
R> data <- c("Y", "n", "p", "K", "m.alpha", "s.alpha", "m.kappa", "s.kappa")
```

Finally, **OpenBUGS** must be told which parameters to "monitor" and MCMC settings such as the number of iterations to "burn", the "thinning" interval, and the total number of iterations to run the Markov chains.

```
R> monitor <- c("alpha", "theta", "kappa")
R> n.burn <- 4000
R> n.thin <- 10
R> n.sim <- 500 * n.thin + n.burn
```

I use a burn-in period of 4,000 iterations to ensure that **OpenBUGS** can complete the adaptive phase of the MCMC algorithm. The number of parallel chains is kept at the default value of 3.

The `openbugs` function can be used to call **OpenBUGS** and generate draws from the posterior distribution of the parameters for the constrained model. The **BUGS** model file is contained in the `grmeq.bug` file of the `bugs` subdirectory of the supplemental material. The `model.file` option in the code below will need to be modified if the working directory has not been set to the base directory of the supplemental material.

```
R> eq.alpha.out <- openbugs(data = data, inits = NULL,
+    parameters.to.save = monitor,
+    model.file = file.path(getwd(), "bugs/grmeq.bug"),
+    n.iter = n.sim, n.thin = n.thin, n.burnin = n.burn))
```

The `openbugs` function returns an object of class `bugs`. One way to access posterior draws in a `bugs` object is via the `sims.matrix` component of the `bugs` object. The `sims.matrix` component is a `matrix` object with number of rows equal to the number of saved iterations of the MCMC simulation and number of columns equal to the number of monitored parameters. Posterior draws of a single parameter can be accessed by passing the name of the parameter to the subset function for matrices, as in

```
eq.alpha.out$sims.matrix[, "theta[162]"]
```

or

```
eq.alpha.out$sims.matrix[, "alpha"]
```

The `grep` function provides a convenient way to access groups of parameters. For example, the code

```
R> parnames <- colnames(eq.alpha.out$sims.matrix)
R> eq.alpha.out$sims.matrix[, grep("theta", parnames)]
```

can be used to return a `matrix` of the posterior draws for all the ability parameters $\theta_1, \ldots, \theta_{291}$.

A call to the generic function `plot` produces some summary plots for the MCMC simulation.

```
R> plot(eq.alpha.out)
```

Among the several summary plots is a plot of the Gelman-Rubin (GR) convergence diagnostics (Gelman and Rubin 1992) for a subset of the parameters. The GR diagnostics are all close to 1, which gives some assurance that the chains have converged.

Additionally, standard, MCMC diagnostic plots (such as trace and autocorrelation plots) can be created in an HTML file and viewed in a browser with a call to the `mcmcplot` function in the **mcmcplots** package (Curtis 2010).

```
R> mcmcplot(eq.alpha.out, random = 20)
```

The fit of this model can be appraised using posterior predictive model checks (see, for example, Gelman, Carlin, Stern, and Rubin 2003, Chapter 6). This method of model checking involves simulating several new data sets using the posterior distribution of the model parameters. Summary statistics from the simulated data sets are compared with the same summary statistics from the observed data. If major discrepancies exist between the distribution of the summary statistics from simulated data and the summary statistics from the observed data, then the model is deemed to have poor fit.

In many latent variable procedures, the goal of model fitting is to find values of model parameters that give a model implied correlation or covariance matrix that is close to the raw correlation matrix computed from the sample data (see, for example, Bollen 1989). This idea can be used in the posterior predictive model checking procedure in the current example by comparing correlation matrices computed on simulated data sets with the correlation matrix from the observed data `Y`. Because the data in this example are ordinal, a nonparametric measure of correlation (e.g., Kendall's tau) must be used. (See Sinharay 2005, for more examples of summary statistics that can be used in the posterior predictive procedure to check the fit of various aspects of IRT models.) A summary of the procedure is below:

1. For $m = 1, \ldots, M$, repeat the following
   (a) For $i = 1, \ldots, n$ generate a new response vector for the i-th subject in the sample based on the m-th posterior draw of model parameters $(\theta_i^{(m)}, \alpha^{(m)}, \kappa_{11}^{(m)}, \ldots, \kappa_{p,K_j-1}^{(m)})$ from the posterior distribution.

   (b) Compute Kendall's correlation matrix for the new data set generated in the previous step, and save the values for later analysis.

2. Compute Kendall's correlation matrix for the raw data.

3. Compare the distribution of simulated values of Kendall's correlation to the observed values of Kendall's correlation.

The details of the procedure are implemented in the function `ppktau` contained in the `functions.R` file of the supplemental material to this article. The function `ppktau` requires the **plyr** package (Wickham 2009). In this example, the `ppktau` function returns a matrix with M rows (where $M = 1,500$ is the number of posterior simulations—500 iterations from 3 parallel chains) and 15 columns for the $p(p-1)/2$ correlations among the 6 variables. Appropriate column names are given to the posterior predictive output via the `makeNames` utility function (also included in the `functions.R` file). Kendall's tau values for the real data are computed with the `cor` function using the `method="kendall"` option.

```
R> pp.eq <- ppktau(eq.alpha.out$sims.matrix)
R> colnames(pp.eq) <- makeNames("ktau", 1:p, 1:p,
+    symmetric.matrix = TRUE, diag = FALSE)
R> ktau <- cor(Y, method = "kendall")[lower.tri(diag(p))]
```

The simulated values of Kendall's tau can be compared to the observed values of Kendall's tau by producing density plots of the simulations and marking on the density plots the corresponding value of the observed Kendall's tau. The function `plotpppval` (in `functions.R`) automates the process of producing density plots for each column of the simulated output. Additionally, the `plotpppval` function shades the area under the density from the observed Kendall's tau to the nearest tail. This tail area is known as the posterior predictive p value or Bayesian p value (Rubin 1984; Meng 1994; Gelman, Meng, and Stern 1996). The Bayesian p values are included on the left-hand side of the density plots produced by `plotpppval`.

```
R> plotpppval(pp.eq, ktau)
```

Figure 1 contains the plot created by the `plotpppval` function. Several of the posterior predictive p values are close to zero, which indicates the model does not fit the data.

The `caterplot` function in the **mcmcplots** package can be used to create "caterpillar" plots of 95% predictive intervals for the simulated Kendall's correlations in `pp.eq`. The `caterpoints` function can be used to overlay the observed values of the Kendall's tau on the same plot. Observed values of Kendall's tau that fall outside the prediction intervals indicate lack of fit.

```
R> caterplot(pp.eq, xlim = c(0.0, 1.0))
R> caterpoints(ktau, pch = "x", col = "red")
```

Figure 2 contains the plot created with the above code. It's clear from the plot that the constrained model does not fit the data well. Several of the observed values of Kendall's tau are far outside the 95% intervals.

The unconstrained model where each $\alpha_1, \ldots, \alpha_6$ is estimated separately can be fit to the `Environment` data and posterior predictive plots can be created using code similar to that above.

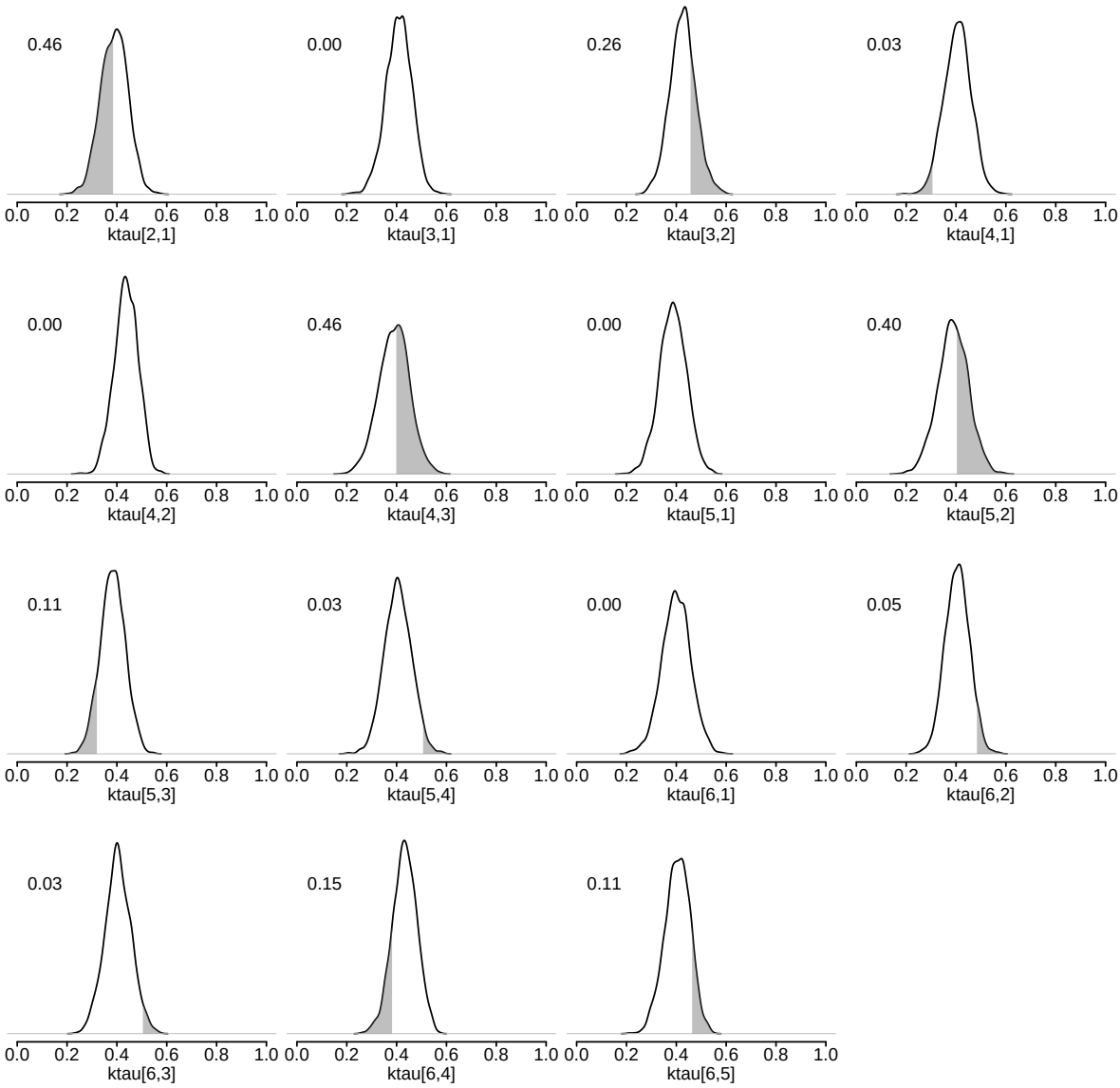

Figure 1: Density plots of simulated Kendall's tau values for the constrained model with $\alpha_1 = \cdots = \alpha_p = \alpha$. Shaded areas on the plot represent the posterior predictive p values. The numeric values of the posterior predictive p values are placed at the left of each density plot.

```
R> free.alpha.out <- openbugs(data = data, inits = NULL,
+    parameters.to.save = monitor,
+    model.file = file.path(getwd(), "bugs/grm.bug"),
+    n.iter = n.sim, n.thin = n.thin, n.burnin = n.burn)
R> plot(free.alpha.out)
R> pp.free <- ppktau(free.alpha.out$sims.matrix)
R> colnames(pp.free) <- makeNames("ktau", 1:p, 1:p,
+    symmetric.matrix = TRUE, diag = FALSE)
R> plotpppval(pp.free, ktau)
```

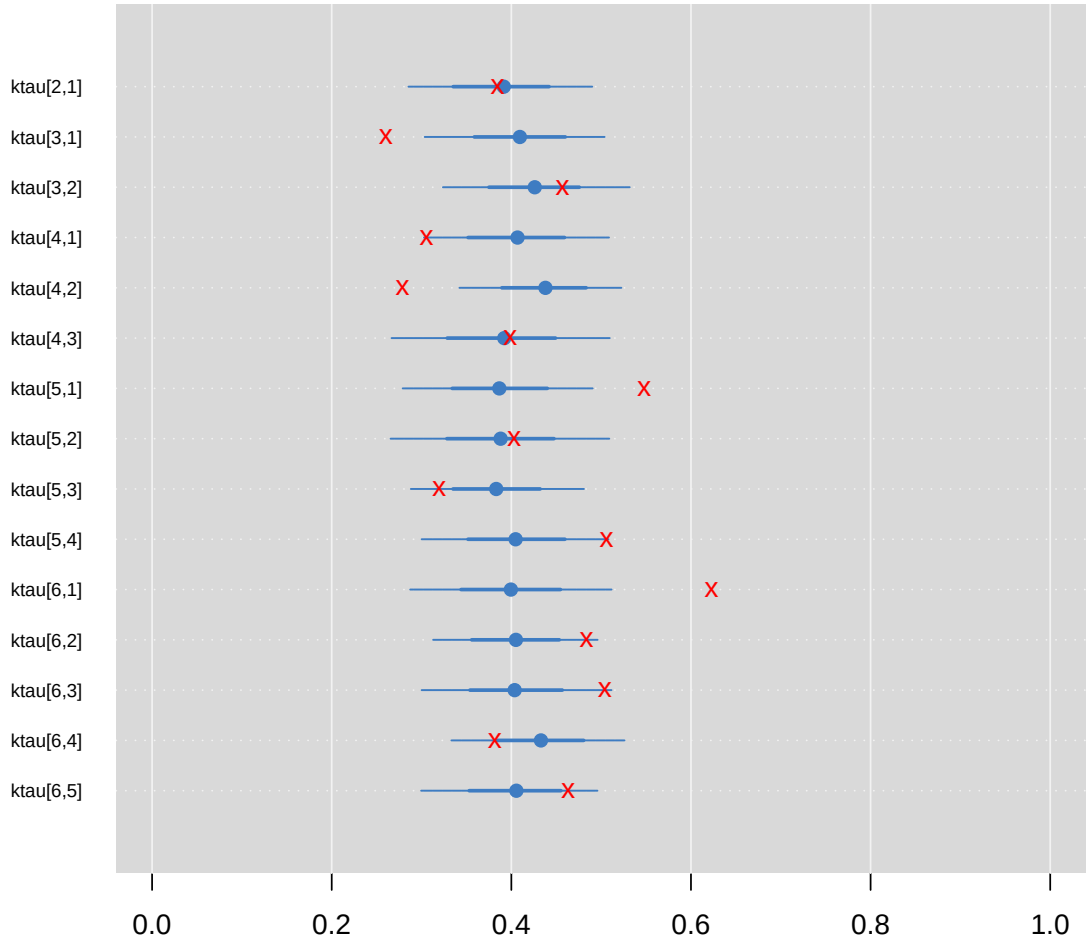

Figure 2: Ninety-five percent prediction intervals for simulated Kendall's tau values from the model with $\alpha_1 = \cdots = \alpha_p = \alpha$. Observed values of Kendall's tau are denoted with an "X" on the plot.

```
R> caterplot(pp.free, xlim = c(0.0,1.0))
R> caterpoints(ktau, pch = "x", col = "red")
```

Once again, the GR diagnostic values are all close to one, which gives some assurance that the Markov chains reached convergence. The posterior predictive p values in Figures 3 and 4 show that the unconstrained model fits better than the model with constrained discrimination parameters. None of the p values in Figure 3 is less than 0.03. Also, each the 95% predictive intervals in Figure 4 captures the corresponding observed value of Kendall's tau.

The results of the posterior predictive model check provide some assurance that the unre-

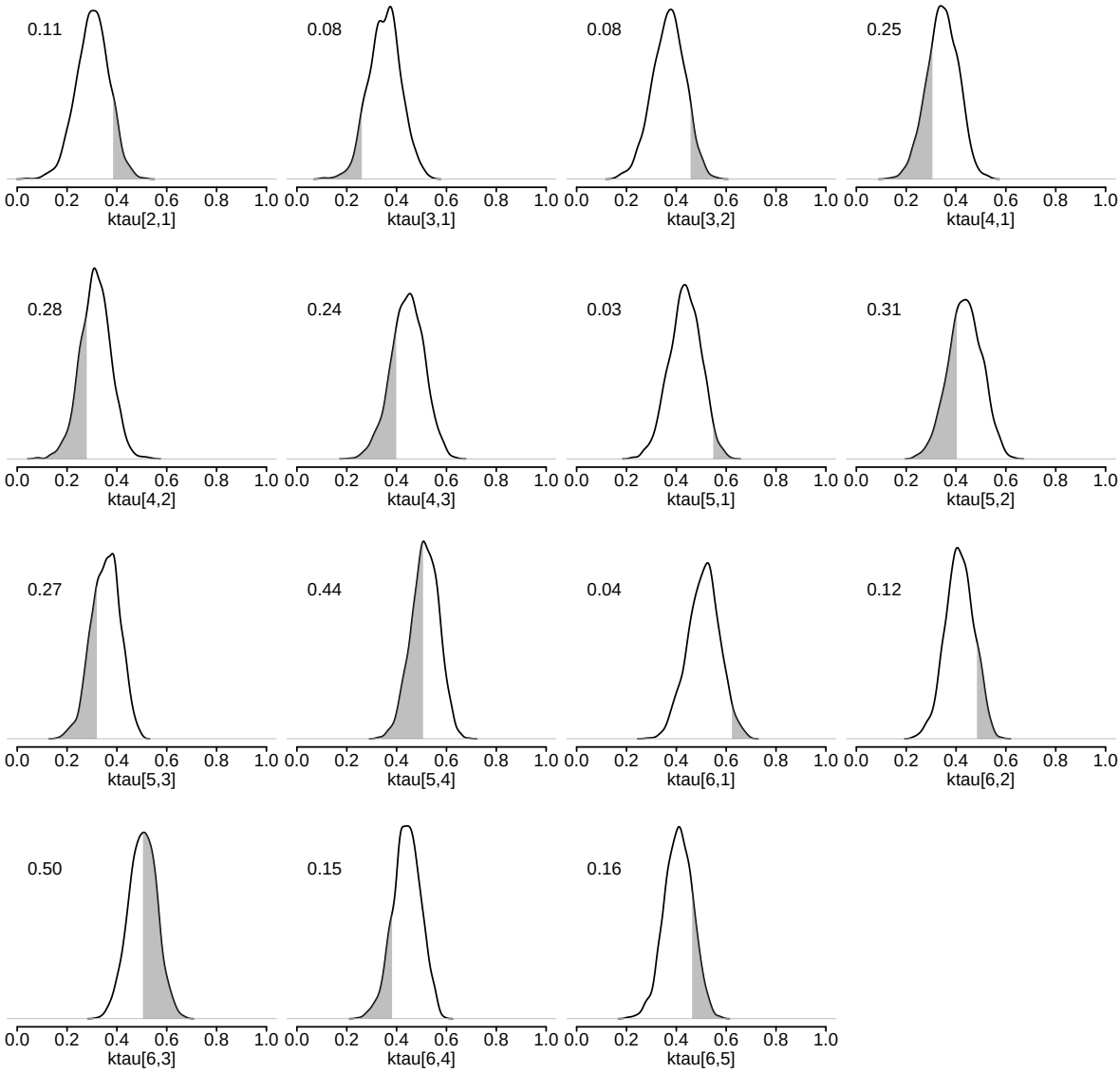

Figure 3: Density plots of simulated Kendall's tau values for the unconstrained model. Shaded areas on the plot represent the posterior predictive p values. The numeric values of the posterior predictive p values are placed at the left of each density plot.

stricted model fits the data. Further analysis of these data can now proceed in much the same way as in more classical analyses of item response data (see, for example, Rizopoulos 2006; Hambleton *et al.* 1991) by examining item characteristic and information curves or factor scores—except posterior means and standard deviations of model parameters should be used in place of maximum likelihood estimates and standard errors.

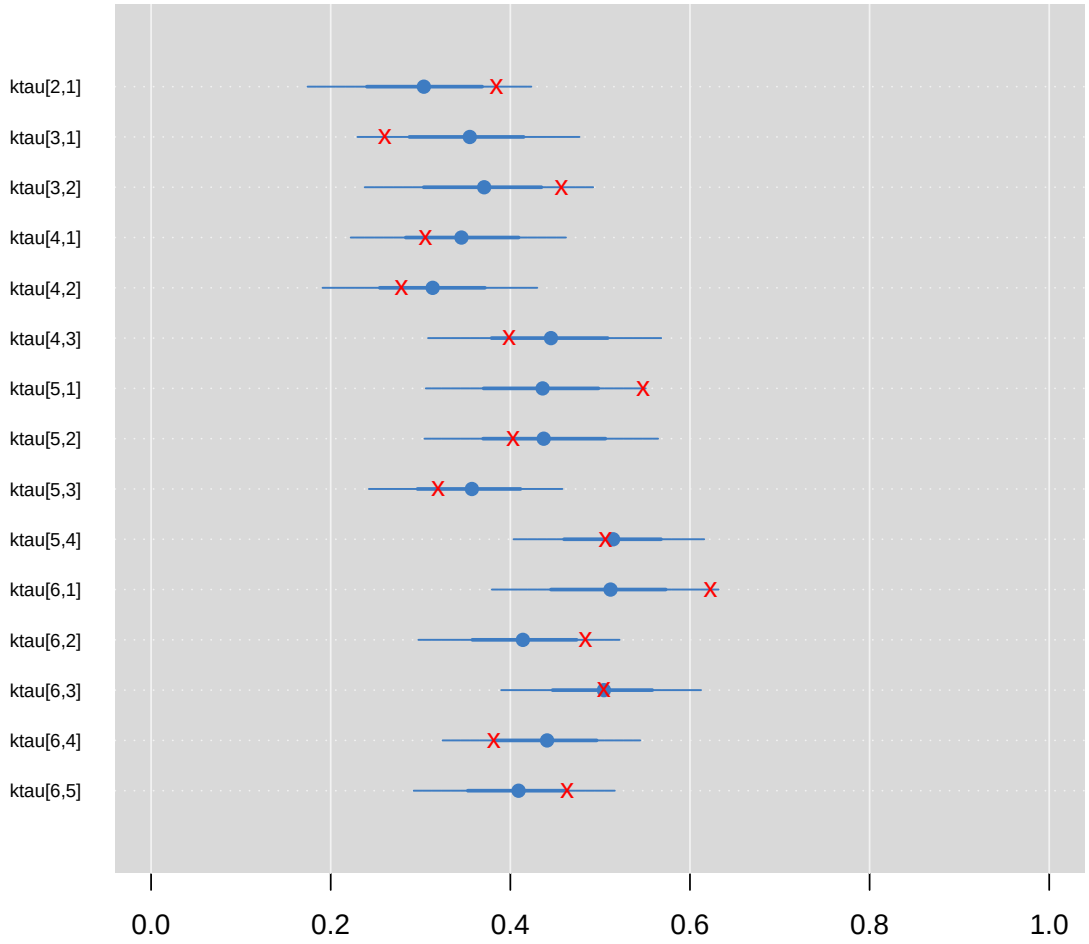

Figure 4: Ninety-five percent prediction intervals for simulated Kendall's tau values from the unconstrained model. Observed values of Kendall's tau are denoted with an "X" on the plot.

11. Summary

In this article, I have presented several snippets of **BUGS** code to fit many of the common IRT models found in the literature. I have also shown how to extend these models to other modeling situations. I hope that researchers who use IRT models will find it useful to have **BUGS** code to fit these basic models in one convenient source and will find that the code can quickly be adapted for use in novel settings.

Acknowledgments

This research was supported by Grant R01 AG 029672 from the National Institute on Aging, Paul K. Crane, PI. I would like thank Paul Crane and two anonymous referees for helpful comments that led to improvements in the paper. Also, I would like to thank Betsy Feldman, Laura Gibbons, Jonathan Gruhl, and Joey Mukherjee for helpful comments on an earlier presentation of the material contained in this paper.

References

Baker FB, Kim SH (2004). *Item Response Theory: Parameter Estimation Techniques*. 2nd edition. Marcel Dekker, Inc., New York.

Bates D, Maechler M (2010). ***lme4****: Linear Mixed-Effects Models Using S4 Classes*. R package version 0.999375-33, URL http://CRAN.R-project.org/package=lme4.

Birnbaum A (1968). "Some Latent Trait Models and Their Use in Inferring an Examinee's Ability." In FM Lord, MR Novick (eds.), *Statistical Theories of Mental Test Scores*, chapter 17–20, pp. 397–479. Addison-Wesley, Reading, MA.

Bollen K (1989). *Structural Equations with Latent Variables*. John Wiley & Sons, New York.

Bradlow ET, Wainer H, Wang X (1999). "A Bayesian Random Effects Model for Testlets." *Psychometrika*, **64**(2), 153–168.

Curtis SM (2010). ***mcmcplots****: Create Plots from MCMC Output*. R package version 0.1, URL http://CRAN.R-project.org/package=mcmcplots.

Davidian M (2005). "Applied Longitudinal Data Analysis: Lecture Notes." URL http://www.stat.ncsu.edu/people/davidian/courses/st732/.

de Leeuw J, Mair P (2007). "An Introduction to the Special Volume on Psychometrics in R." *Journal of Statistical Software*, **20**(1), 1–5. URL http://www.jstatsoft.org/v20/i01/.

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

Doran H, Bates D, Bliese P, Dowling M (2007). "Estimating the Multilevel Rasch Model: With the **lme4** Package." *Journal of Statistical Software*, **20**(2), 1–18. URL http://www.jstatsoft.org/v20/i02/.

du Toit M (2003). *IRT from SSI:* ***BILOG-MG****,* ***MULTILOG****,* ***PARSCALE****,* ***TESTFACT***. Scientific Software International, Lincolnwood, IL.

Fischer GH, Molenaar IW (eds.) (1995). *Rasch Models: Foundations, Recent Developments, and Applications*. Springer-Verlag, New York.

Fox JP (2007). "Multilevel IRT Modeling in Practice with the Package **mlirt**." *Journal of Statistical Software*, **20**(5), 1–16. URL http://www.jstatsoft.org/v20/i05/.

Fox JP, Entink RK, van der Linden W (2007). "Modeling of Responses and Response Times with the Package **cirt**." *Journal of Statistical Software*, **20**(7), 1–14. URL http://www.jstatsoft.org/v20/i07/.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **3**, 515–534.

Gelman A, Carlin JB, Stern HS, Rubin DB (2003). *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, FL.

Gelman A, Hill J (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, New York.

Gelman A, Meng XL, Stern H (1996). "Posterior Predictive Assessment of Model Fitness via Realized Discrepancies." *Statistica Sinica*, **6**, 733–808.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**, 457–511.

Gilks WR, Thomas A, Spiegelhalter DA (1994). "A Language and Program for Complex Bayesian Modelling." *The Statistician*, **43**, 169–178.

Hambleton RK, Swaminathan H, Rogers HJ (1991). *Fundamentals of Item Response Theory*. SAGE Publications, Inc., Newbury Park, CA.

Johnson MS (2007). "Marginal Maximum Likelihood Estimation of Item Response Models in R." *Journal of Statistical Software*, **20**(10), 1–24.

Li Y, Bolt DM, Fu J (2006). "A Comparison of Alternative Models for Testlets." *Applied Psychological Measurement*, **30**(1), 3–21.

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

Lunn D, Spiegelhalter D, Thomas A, Best N (2009). "The **BUGS** Project: Evolution, Critique and Future Directions." *Statistics in Medicine*, **28**, 3049–3067.

Lunn DJ, Thomas A, Best N, Spiegelhalter D (2000). "**WinBUGS**– A Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing*, **10**, 325–337.

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

Mair P, Hatzinger R (2010). "CRAN Task View: Psychometric Models and Methods." Version 2010-07-30, URL http://CRAN.R-project.org/view=Psychometrics.

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

Meng XL (1994). "Posterior Predictive p-values." *The Annals of Statistics*, **22**, 1142–1160.

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

Muraki E, Bock D (2005). ***PARSCALE** 4*. Scientific Software International, Inc., Lincolnwood, IL. URL http://www.ssicentral.com/.

Patz RJ, Junker BW (1999). "A Straightforward Approach to Markov Chain Monte Carlo Methods for Item Response Models." *Journal of Educational and Behavioral Statistics*, **24**(2), 146–178.

Plummer M (2003). "**JAGS**: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In K Hornik, F Leisch, A Zeileis (eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*. ISSN 1609-395X, URL http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/.

Plummer M (2010a). ***JAGS** Version 2.0.0 User Manual*. Lyon, France. URL http://www-fis.iarc.fr/~martyn/software/jags/.

Plummer M (2010b). ***rjags**: Bayesian Graphical Models Using MCMC*. R package version 2.1.0-4, URL http://CRAN.R-project.org/package=rjags.

Reckase MD (2009). *Multidimensional Item Response Theory*. Springer-Verlag, New York.

Rizopoulos D (2006). "**ltm**: An R Package for Latent Variable Modeling and Item Response Theory." *Journal of Statistical Software*, **17**(5), 1–25.

Rubin DB (1984). "Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician." *The Annals of Statistics*, **12**, 1151–1172.

Samejima F (1969). "Estimation of Latent Ability Using a Response Pattern of Graded Scores." Psychometrika Monograph, No. 17.

Sinharay S (2005). "Assessing Fit of Unidimensional Item Response Theory Models Using a Bayesian Approach." *Journal of Educational Measurement*, **42**(4), 375–394.

Spiegelhalter D, Thomas A, Best N, Lunn D (2003). ***WinBUGS** Version 1.4 User Manual*. MRC Biostatistics Unit. URL http://www.mrc-bsu.cam.ac.uk/bugs/.

Spiegelhalter D, Thomas A, Best N, Lunn D (2010). ***OpenBUGS** Version 3.1.1 User Manual*. Helsinki, Finland. URL http://www.openbugs.info/.

Sturtz S, Ligges U, Gelman A (2005). "**R2WinBUGS**: A Package for Running **WinBUGS** from R." *Journal of Statistical Software*, **12**(3), 1–16. URL http://www.jstatsoft.org/v12/i03/.

Su Y, Yajima M (2010). ***R2jags**: A Package for Running **JAGS** from R*. R package version 0.02-02, URL http://CRAN.R-project.org/package=R2jags.

Tavares HR, Andrade DF (2006). "Item Response Theory for Longitudinal Data: Item and Population Abilty Parameters Estimation." *Test*, **15**, 97–123.

Thissen D, Chen W, Bock D (2003). ***MULTILOG** 7*. Scientific Software International, Inc., Lincolnwood, IL. URL http://www.ssicentral.com/.

Thomas A, O'Hara B, Ligges U, Sturtz S (2006). "Making **BUGS** Open." *R News*, **6**(1), 12–17. URL http://CRAN.R-project.org/doc/Rnews/.

Wainer H, Bradlow ET, Wang X (2007). *Testlet Response Theory and its Applications*. Cambridge University Press, New York.

Wang X, Bradlow ET, Wainer H (2005). *User's Guide for* ***Scoright*** *(Version 3.0): A Computer Program for Scoring Tests Built of Testlets Including a Module for Covariate Analysis*. Educational Testing Service.

Wickham H (2009). ***plyr****: Tools for Splitting, Applying and Combining Data*. R package version 0.1.9, URL http://CRAN.R-project.org/package=plyr.

Wood R, Wilson D, Gibbons R, Schilling S, Muraki E, Bock D (2003). ***TESTFACT*** *4*. Scientific Software International, Inc., Lincolnwood, IL. URL http://www.ssicentral.com/.

Zimowski M, Muraki E, Mislevy R, Bock D (2005). ***BILOG-MG*** *3 – Multiple-Group IRT Analysis and Test Maintenance for Binary Items*. Scientific Software International, Inc., Lincolnwood, IL. URL http://www.ssicentral.com/.

Affiliation:

S. McKay Curtis
University of Washington
Department of Statistics
Box 354320
Seattle, WA 98195-4320, United States of America
E-mail: smcurtis@stat.washington.edu
URL: http://www.stat.washington.edu/~smcurtis/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 36, Code Snippet 1 *Submitted:* 2010-03-30
August 2010 *Accepted:* 2010-07-07