Tutorial

Journal of Educational and Behavioral Statistics
Vol. XX, No. X, pp. 1–31
DOI: 10.3102/1076998619826040
Article reuse guidelines: sagepub.com/journals-permissions

Using JAGS for Bayesian Cognitive Diagnosis Modeling: A Tutorial

Peida Zhan
Zhejiang Normal University

Hong Jiao
Kaiwen Man
University of Maryland

Lijun Wang
Zhejiang Normal University

In this article, we systematically introduce the just another Gibbs sampler (JAGS) software program to fit common Bayesian cognitive diagnosis models (CDMs) including the deterministic inputs, noisy "and" gate model; the deterministic inputs, noisy "or" gate model; the linear logistic model; the reduced reparameterized unified model; and the log-linear CDM (LCDM). Further, we introduce the unstructured latent structural model and the higher order latent structural model. We also show how to extend these models to consider polytomous attributes, the testlet effect, and longitudinal diagnosis. Finally, we present an empirical example as a tutorial to illustrate how to use JAGS codes in R.

Keywords: *cognitive diagnosis modeling; Bayesian estimation; Markov chain Monte Carlo; DINA model; DINO model; rRUM; testlet; longitudinal diagnosis; polytomous attributes*

Introduction

In recent years, many cognitive diagnosis models (CDMs) have been proposed, such as the deterministic inputs, noisy "and" gate (DINA) model (Haertel, 1989; Junker & Sijtsma, 2001; Macready & Dayton, 1977); the deterministic inputs, noisy "or" gate (DINO) model (Templin & Henson, 2006); the linear logistic model (Maris, 1999); and the reduced reparameterized unified model (rRUM; Hartz, 2002). A few general CDMs have also become available, such as the log-linear CDM (LCDM; Henson, Templin, & Willse, 2009), the generalized DINA model (de la Torre, 2011), and the general diagnostic model (von Davier, 2008).

With the advancement in computing power and the Markov chain Monte Carlo (MCMC) algorithms, Bayesian CDM has become increasingly popular (Culpepper, 2015a; Culpepper & Hudson, 2018; DeCarlo, 2012; de la Torre & Douglas, 2004; Huang & Wang, 2014; F. Li, Cohen, Bottge, & Templin, 2016; X. M. Li & Wang, 2015; Sinharay & Almond, 2007; Zhan, Jiao, & Liao, 2018; Zhan, Jiao, Liao, & Bian, 2018). Multiple software programs are available to implement certain Bayesian MCMC algorithms, such as WinBUGS (Lunn, Thomas, Best, & Spiegelhalter, 2000), OpenBUGS (Spiegelhalter, Thomas, Best, & Lunn, 2014), just another Gibbs sampler (JAGS; Plummer, 2015), and MCMCpack (Martin, Quinn, & Park, 2011) package in R (R Core Team, 2016). However, there remains a lack of systematic introduction to using such software programs to fit Bayesian CDMs.

Unlike the frequentist approach that treats model parameters as fixed, the Bayesian approach considers them as random and uses (prior) distributions to model our beliefs regarding them. Within the frequentist framework, parameter estimation refers to a point estimate of each model parameter. In contrast, in a fully Bayesian analysis, we seek a whole (posterior) distribution of the model parameter, which includes the entire terrain of peaks, valleys, and plateaus (Levy & Mislevy, 2016). The posterior distribution of parameters, given the data, is proportional to the product of the likelihood of the data, given the parameters, and the prior distribution of the parameters. Typically, the posterior distribution is represented in terms of the posterior mean (or median, or mode) as a summary of central tendency, as well as the posterior standard deviation as a summary of variability.

Adopting the Bayesian MCMC estimation over a frequentist estimation—for example, a maximum likelihood estimation (see, e.g., Wagenmakers, Lee, Lodewyckx, & Iverson, 2008)—includes the following advantages: (a) It does not depend on asymptotic theory, (b) it treats both item and person parameters as random effects, (c) it incorporates the principle of parsimony by marginalization of the likelihood function, and (d) it is more robust in terms of handling complex models. Moreover, in the Bayesian estimation, the percentiles of the posterior can be used to construct the credible interval (or Bayesian confidence interval), which can be used for testing significance (Box & Tiao, 1973). Further, it is also easy to conduct model–data fit with posterior predictive model checking (PPMC).

Currently, numerous CDM studies are rather technical and limited to statistical and psychometric researchers (Templin & Hoffman, 2013). There is a lack of available software for more applied practitioners who would like to use CDMs in developing their diagnostic testing programs or conducting empirical research. Moreover, most existing programs for cognitive diagnosis are either limited in terms of model options or commercialized. For example, the Arpeggio Suite (Bolt et al., 2008), mdltm (von Davier, 2005), CDM package (George, Robitzsch, Kiefer, Gross, & Uenlue, 2016), and the GDINA package (Ma & de la Torre, 2016a) limit users to a few options. In addition, although the Mplus (Muthén & Muthén, 2010) and the flexMIRT (Cai, 2017) can be used to fit many CDMs (Hansen, Cai, Monroe, & Li, 2016; Templin & Hoffman, 2013), their commercialization may limit their access to certain researchers or students. More importantly, CDMs can only be estimated on the basis of the frequentist estimation methods (e.g., maximum likelihood estimation) that are embedded in their software.

In this article, we demonstrate how to use the freeware JAGS to fit several popular CDMs and present the codes of these CDMs. Then, researchers will be able to adapt these codes to fit extended CDMs, which cannot be fitted into existing software or packages.

The Gibbs sampler is used by default in JAGS. In general, JAGS makes it easy to construct a Markov chain for parameters and does not require users to derive the posterior distribution of the model parameters by hand. Moreover, the R2jags package (Version 0.5-7; Su & Yajima, 2015) in R could easily be used to call JAGS. Further, it must be noted that the JAGS codes presented in this study can be easily generalized to other BUGS software programs with minor revisions, such as WinBUGS and OpenBUGS.[1]

The following sections illustrate JAGS codes for five CDMs: (1) the DINA model, (2) the DINO model, (3) the log-linear model (LLM), (4) the rRUM, and (5) the LCDM. Apart from these five models, which are based on the unstructured (or saturated) latent structural model, we also demonstrate the higher order latent structural model (de la Torre & Douglas, 2004). Further, we also present the extensions to the polytomous attributes, the testlet effect, and the longitudinal diagnosis using JAGS. Lastly, we conduct an empirical example analysis to illustrate how to use the R2jags package to run the example JAGS codes.

The DINA Model

Let Y_{ni} be the response of person n ($n = 1, \ldots, N$) to item i ($i = 1, \ldots, I$). Let α_{nk} be the binary variable for person n on attribute k ($k = 1, \ldots, K$,), where $\alpha_{nk} = 1$ indicates that person n shows mastery over attribute k and $\alpha_{nk} = 0$ indicates nonmastery, and let $\alpha_n = (\alpha_{n1}, \ \ldots \ , \alpha_{nK})'$ be the nth person's attribute pattern. Let q_{ik} denote the element in an $I \times K$ Q-matrix (Tatsuoka, 1983), with q_{ik} indicating whether attribute k is required in order to answer item i correctly. If the attribute is required, $q_{ik} = 1$, otherwise $q_{ik} = 0$.

Among CDMs, the DINA model is one of the most popular models because of its simple structure and straightforward interpretation. The DINA model can be expressed in the following manner:

$$p_{ni} = P(Y_{ni} = 1|\alpha_n) = g_i + (1 - s_i - g_i)\eta_{ni}, \tag{1}$$

where p_{ni} is the correct response probability of person n to item i; s_i and g_i are the slipping and guessing probabilities, respectively, of item i, which describe the item-level aberrant response probability; η_{ni} is the ideal response for person n to

TABLE 1.
The DINA Model

```
1.  model{
2.    for (n in 1:N) {
3.      for (i in 1:I) {
4.        for (k in 1:K) {w[n, i, k] <- pow(alpha[n, k], Q[i, k])}
5.          eta[n, i] <- prod(w[n, i, 1:K])
6.          p[n, i] <- g[i] + (1 - s[i] - g[i]) * eta[n, i]
7.          Y[n, i] ~ dbern(p[n, i])}
8.    for (k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
9.    c[n] ~ dcat(pai[1:C])}
10.   pai[1:C] ~ ddirch(delta[1:C])
11.   for (i in 1:I) {
12.     s[i] ~ dbeta(a.s, b.s)
13.     g[i] ~ dbeta(a.g, b.g)  T(0, 1 - s[i])}}
```

Note. DINA = deterministic inputs, noisy "and" gate.

item i based on the conjunctive condensation rule (Maris, 1999), assuming a value of 1 if person n possesses all the attributes required for item i and a value of 0 if the person lacks at least one of the required attributes; mathematically, it is expressed in the following manner:

$$\eta_{ni} = \prod_{k=1}^{K} w_{nik} = \prod_{k=1}^{K} \alpha_{nk}^{q_{ik}}, \tag{2}$$

where w_{nik} can be treated as the latent response on item i for person n to attribute k. Table 1 presents the JAGS codes to fit the DINA model. The codes are elaborated below.

Line 1 signals the beginning of the model. Lines 2 through 7 specify the measurement model, lines 8 through 10 are the unstructured latent structural model and priors, and lines 11 through 13 are the priors assumed for the item parameters.

A part of the parameters in Table 1 is assigned with certain previously defined values, including `all.patterns`, `C`, `delta`, `Y`, and `Q`. Specifically, `C` is the number of all possible attribute profiles, typically 2^K; `all.patterns` is a given $C \times K$ matrix that contains all possible attribute patterns, one for each row; and `delta` is the scale parameter vector of the Dirichlet distribution. For generalization, we set `delta` = (1, 1, . . . , 1), which means that the mixing proportion, `pai`, for all possible patterns follows an uninformative uniform prior distribution; `Y` is an $N \times I$ item response matrix; and `Q` is the $I \times K$ Q-matrix. More details regarding the use of these previously defined parameters in the JAGS codes are provided in the An Empirical Example: A Tutorial section.

In the unstructured latent structural model, line 8 describes the method used to obtain the following attributes: $\alpha_{nk} = \alpha_{ck}$, where $c \in \{1, \ldots, C\}$ indicates person n's attribute profile and is assumed to follow a categorical distribution, with the mixing proportion of the cth pattern.

TABLE 2.
The RDINA Model

```
1.  model{
2.    for (n in 1:N){
3.      for (i in 1:I){
4.        for (k in 1:K){w[n, i, k] <- pow(alpha[n, k], q[i, k])}
5.        eta[n, i] <- prod(w[n, i, 1:K])
6.        logit(p[n, i]) <- lamda0[i] + lamdaK[i] * eta[n,i]
7.        Y[n,i] ~ dbern(p[n, i])}
8.    for (k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
9.    c[n] ~ dcat(pai[1:C])}
10.   pai[1:C] ~ ddirch(delta[1:C])
11.   for(i in 1:I){
12.     lamda0[i]~dnorm(mean.lamda0, pr.lamda0)
13.     lamdaK[i]~dnorm(mean.lamdaK, pr.lamdaK) T(0, )}}
```

Note. RDINA = reparameterized deterministic inputs, noisy “and” gate.

Lines 12 and 13 specify the prior distribution of s_i and g_i, respectively. In addition, a monotonicity restriction ($g_i < 1 - s_i$) is added by truncation `T(0, 1-s[i])`. In order to increase the universality of our codes and/or to represent vague prior beliefs, uninformative priors may be employed.[2] Hence, the scale parameters of the β distributions are assigned as `a.s` = `b.s` = `a.g` = `b.g` = 1, which is identical to a linearly truncated bivariate uninformative uniform prior for s_i and g_i. On the other hand, according to certain previous experiences, informative priors can also be used. For example, according to the results of a few previous studies (e.g., Y. Chen, Culpepper, Chen, & Douglas, 2018; DeCarlo, 2012; de la Torre & Douglas, 2004; Zhan, Jiao, Liao, & Bian, 2018), the quality of items in the fraction subtraction test (Tatsuoka, 1990) is relatively good. Hence, more informative priors can be used by setting `a.s` = 1, `b.s` = 3, `a.g` = 1, and `b.g` = 3, which have a greater probability of sampling small numbers (e.g., 0.1) than the uniform prior.[3]

In addition, when no previous experience is available for the scale parameters, a prior on the scale parameters—which is called a hyperprior—can be used. A few extra lines can be added in the following manner:

```
a.s ~ dunif (0.1, 5),
b.s ~ dunif (0.1, 5),
a.g ~ dunif (0.1, 5),
b.g ~ dunif (0.1, 5).
```

Then, `a.s`, `b.s`, `a.g`, and `b.g` can be estimated.

The JAGS codes of the reparameterized DINA (RDINA) model (DeCarlo, 2012) are presented in Table 2. The RDINA model uses the logit link function and it is equivalent to the regular DINA model, which can be expressed in the following manner:

$$p_{ni} = P(Y_{ni} = 1|\alpha_n) = \frac{\exp(\lambda_{0,i} + \lambda_{(K),i}\eta_{ni})}{1 + \exp(\lambda_{0,i} + \lambda_{(K),i}\eta_{ni})}, \tag{3}$$

where the intercept parameter ($\lambda_{0,i}$) defines the log odds of a correct response to item i for a person who is not a master of either one of the attributes; $\lambda_{(K),i}$ is the K-way interaction effect parameter for item i. In this formulation, the regular g_i and s_i parameters in Equation 1 can be described in the following manner:

$$p_{ni} = \begin{cases} \dfrac{\exp(\lambda_{0,i})}{1+\exp(\lambda_{0,i})} = g_i & \text{if } \eta_{ni} = 0 \\ \dfrac{\exp(\lambda_{0,i} + \lambda_{(K),i})}{1+\exp(\lambda_{0,i} + \lambda_{(K),i})} = 1 - s_i & \text{if } \eta_{ni} = 1 \end{cases}. \tag{4}$$

Lines 4 through 10 specify the model. Line 12 specifies the distribution for the $\lambda_{0,i}$ parameter. A normal prior distribution is assumed targeting a mean of -1.096—that is, `mean.lamda0` $= -1.096$. This is equivalent to a mean guessing value, g_i of 0.25, which equals the random guessing probability of a four-option item. Line 13 specifies the distribution for the $\lambda_{(K),i}$ parameter. A normal prior distribution is assumed to target a mean of 2.192, that is, `mean.lamdaK` $= 2.192$. This makes the mean value of s_i also equal to 0.25. For the sake of generalization, a less informative prior is assumed. Then, the variances of prior distributions for $\lambda_{0,i}$ and $\lambda_{(K),i}$ parameters can be set at 4. JAGS parameterizes the normal distribution in terms of precision (i.e., the inverse of the variance). Thus, a variance of 4 must be converted to a precision of `pr.lamda0` $= 0.25$ and `pr.lamdaK` $= 0.25$ in lines 12 and 13, respectively. In addition, hyperpriors can be used here, such as

```
mean.lamda0 ~ dnorm(-1.096, 0.5),
pr.lamda0 ~ dgamma(1, 1).
```
[4]

Further, the monotonicity restriction ($g_i < 1 - s_i$) is realized by constraining $\lambda_{(K),i}$ parameters to be positive. Thus, a truncated normal distribution is specified for $\lambda_{(K),i}$ in line 13 by truncation `T(0, )`.

The DINO Model

The DINO model, similar to the DINA model, models the probability of a correct response as a function of a slipping parameter, s_i, and a guessing parameter, g_i. However, the ideal response, η_{ni}, in the DINO model is modeled on the basis of the disjunctive condensation rule (Maris, 1999) rather than the conjunctive condensation rule, as in the DINA model. η_{ni} is expressed as

$$\eta_{ni} = 1 - \prod_{k=1}^{K} w_{nik} = 1 - \prod_{k=1}^{K} (1 - \alpha_{nk})^{q_{ik}}, \tag{5}$$

which is an indicator of whether person n has mastered at least one of the required attributes for item i. Thus, $\eta_{ni} = 1$ for any person who has mastered one or more of the item's required attributes, and $\eta_{ni} = 0$ for a person who has mastered none of the required attributes. Although the DINO model shares a dual relationship

TABLE 3.
The DINO Model

```
1.  model{
2.    for (n in 1:N) {
3.      for (i in 1:I) {
4.        for (k in 1:K) {w[n, i, k] <- pow(1 - alpha[n, k], Q[i, k])}
5.        eta[n, i] <- 1 - prod(w[n, i, 1:K])
6.        p[n, i] <- g[i] + (1 - s[i] - g[i]) * eta[n, i]
7.        Y[n, i] ~ dbern(p[n, i])}
8.    for (k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
9.    c[n] ~ dcat(pai[1:C])}
10.   pai[1:C] ~ ddirch(delta[1:C])
11.   for (i in 1:I) {
12.     s[i] ~ dbeta(1, 1)
13.     g[i] ~ dbeta(1, 1)  T(, 1 - s[i])}}
```

Note. DINO = deterministic inputs, noisy "or" gate.

with the DINA model (Köhn & Chiu, 2016), it is easier for practitioners to directly fit the DINO model to the data.

Table 3 presents the JAGS codes for the DINO model. The differences between the DINO and DINA models are easily handled by JAGS, as shown in lines 4 and 5 in Tables 1 and 3, respectively.

The LLM

The LLM (also called the compensatory reparameterized unified model) is constructed on the basis of the compensatory condensation rule (Maris, 1999). The LLM can be expressed in the following manner:

$$p_{ni} = P(Y_{ni} = 1|\alpha_n) = \frac{\exp\left(\lambda_{0,i} + \sum_{k=1}^{K} \lambda_{k,i} w_{nik}\right)}{1 + \exp\left(\lambda_{0,i} + \sum_{k=1}^{K} \lambda_{k,i} w_{nik}\right)} = \frac{\exp\left(\lambda_{0,i} + \sum_{k=1}^{K} \lambda_{k,i} \alpha_{nk} q_{ik}\right)}{1 + \exp\left(\lambda_{0,i} + \sum_{k=1}^{K} \lambda_{k,i} \alpha_{nk} q_{ik}\right)}, \tag{6}$$

where $\lambda_{k,i}$ is the kth main effect parameter and all $\lambda_{k,i} \geq 0$. In the LLM, the lowest correct response probability is $\frac{\exp(\lambda_{0,i})}{1+\exp(\lambda_{0,i})}$ and denotes the probability of a correct response to item i without mastering any of the required attributes. The probability increases as a function of each required attribute that is mastered, as defined by $\lambda_{k,i}$. Finally, the highest probability is $\frac{\exp(\lambda_{0,i}+\sum_{k=1}^{K} \lambda_{k,i} q_{ik})}{1+\exp(\lambda_{0,i}+\sum_{k=1}^{K} \lambda_{k,i} q_{ik})}$, which denotes the probability of an incorrect response to item i by mastering all the required attributes.

The JAGS codes of the LLM are presented in Table 4. For 1 item, the number of main effect parameters is $\sum_{k=1}^{K} q_{ik}$, which is the number of attributes assessed

TABLE 4.
The LLM

```
model{
  for (n in 1:N){
    for (i in 1:I){
      for (k in 1:K){w[n, i, k] <- alpha[n, k] * Q[i, k]}
      eta[n, i] <- inprod(lamda[i, 1:K], w[n, i, 1:K])
      logit(p[n, i]) <- lamda0[i] + eta[n, i]
      Y[n, i] ~ dbern(p[n, i])}
  for (k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
  c[n] ~ dcat(pai[1:C])}
  pai[1:C] ~ ddirch(delta[1:C])
  for(i in 1:I){
    lamda0[i] ~ dnorm(-1.096, 0.25)
    for(k in 1:K){
      lamda[i, k] <- xlamda[i, k] * Q[i, k]
      xlamda[i, k] ~ dnorm(0, 0.25) T(0, )}}}}
```

Note. LLM = log-linear model.

by this item. For example, if an item requires the first two attributes and the test requires a total of three attributes, the number of main effect parameters in this item is two rather than three. Thus, only two main effect parameters must be monitored and reported. A prior can be induced to the main effect parameters by defining auxiliary parameters `xlamda` that originate from the truncated normal distribution. `lamda` was used for monitoring and final reporting.

The rRUM

In the DINA model, the aberrant responses are modeled at the item level. However, in practice, it may appear reasonable that a respondent lacking only one of the measured attributes has a higher chance of a correct response than a respondent who has not mastered any of the measured attributes. To further differentiate between respondents who have not mastered at least one attribute, the noisy inputs, deterministic "and" gate (NIDA) model (Junker & Sijtsma, 2001) models the aberrant responses at the attribute level, but with equality constraints across items. A straightforward extension of the NIDA model is the generalized NIDA (G-NIDA) model (de la Torre, 2011), in which the slipping and guessing parameters are allowed to vary across items. Thus, there are $2\sum_{i=1}^{I}\sum_{k=1}^{K} q_{ik}$ parameters to be estimated, which makes the model unidentifiable (Culpepper & Hudson, 2018; Jiang, 1996). To make the G-NIDA model identifiable, Hartz (2002) proposed the rRUM, which is a reparameterized version of the G-NIDA model (Culpepper & Hudson, 2018; de la Torre, 2011). The rRUM can be expressed in the following manner:

$$p_{ni} = P(Y_{ni} = 1|\alpha_n) = \pi_i^* \prod_{k=1}^{K} r_{ik}^{*\,w_{nik}} = \pi_i^* \prod_{k=1}^{K} r_{ik}^{*\,(1-\alpha_{nk})q_{ik}}, \tag{7}$$

TABLE 5.
The rRUM

```
1. model{
2.   for(n in 1:N){
3.     for(i in 1:I){
4.       for(k in 1:K){w[n,i,k] <- (1 - alpha[n,k])*Q[i, k]}
5.       p[n, i] <- pai_star[i] * prod(pow(r_star[i, 1:K],w[n, i, 1:K]))
6.       Y[n, i] ~ dbern(p[n, i])}
7.     for(k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
8.     c[n] ~ dcat(pai[1:C])}
9.   pai[1:C] ~ ddirch(delta[1:C])
10.  for(i in 1:I){
11.    pai_star[i] ~ dbeta(a.pai_star, b.pai_star)
12.    for(k in 1:K){
13.      r_star[i,k] <- xr_star[i, k] * Q[i, k]
13.      xr_star[i,k] ~ dbeta(a.xr_star, b.xr_star)}}}
```

Note. rRUM = reduced reparameterized unified model.

where π_i^* is the baseline parameter that defines the probability of a correct response to item i, given all required attributes, and r_{ik}^* is the penalty parameter for not having mastered a required attribute k. In the rRUM, there are $\sum_{i=1}^{I}(1+\sum_{k=1}^{K} q_{ik})$ parameters to be estimated. The JAGS codes for this model are provided in Table 5.

Following the same sequence, the rRUM is first specified from lines 4 through 6, and priors are specified in the subsequent lines of the table. Please note the distinction between `pai` and `pai_star` in Table 5. The former is the mixing proportion for all possible patterns, while the latter is the baseline item parameter.

For 1 item, only $\sum_{k=1}^{K} q_{ik}$ penalty parameters must be monitored and reported. A prior can be induced to the penalty parameters by defining auxiliary parameter `xr_star`, which is assumed from a β distribution. `r_star` is utilized for monitoring and final reporting purposes. The baseline and penalty parameters both are restricted to values between 0 and 1. Thus, the β distributions are used as the priors. For noninformative priors, `a.pai_star`, `b.pai_star`, `a.xr_star`, and `b.xr_star` can be set as 1. In contrast, according to the meaning of these two parameters, more informative priors can be set as `a.pai_star` = 3, `b.pai_star` = 1, `a.xr_star` = 3, and `b.xr_star` = 1.

The LCDM

Among the CDMs, the LCDM is sufficiently general to encompass many popular CDMs (e.g., the DINA model, the DINO model, the rRUM, and the LLM), which are special cases obtained by imposing different constraints on item parameters (Henson et al., 2009; Rupp, Templin, & Henson, 2010). In the LCDM, the correct response probability for person n on item i is defined in the following manner:

$$p_{ni} = P(Y_{ni} = 1|\alpha_n)$$
$$= \frac{\exp(\lambda_{0,i} + \sum_{k=1}^{K} \lambda_{k,i}\alpha_{nk}q_{ik} + \sum_{k=1}^{K-1}\sum_{k'=k+1}^{K} \lambda_{kk',i}\alpha_{nk}\alpha_{nk'}q_{ik}q_{ik'} + \ldots + \lambda_{(K),i}\prod_{k=1}^{K}\alpha_{nk}q_{ik})}{1 + \exp(\lambda_{0,i} + \sum_{k=1}^{K} \lambda_{k,i}\alpha_{nk}q_{ik} + \sum_{k=1}^{K-1}\sum_{k'=k+1}^{K} \lambda_{kk',i}\alpha_{nk}\alpha_{nk'}q_{ik}q_{ik'} + \ldots + \lambda_{(K),i}\prod_{k=1}^{K}\alpha_{nk}q_{ik})}, \quad (8)$$

where the intercept parameter, $\lambda_{0,i}$, defines the log odds of a correct response for a person who does not master any attribute, $\lambda_{k,i}$ is the main effect of α_{nk}, $\lambda_{kk',i}$ is the two-way interaction effect of α_{nk} and $\alpha_{nk'}$, and $\lambda_{(K),i}$ is the K-way interaction effect. To keep p_{ni} increasing as the number of mastered attributes increase, $\lambda_{k,i}$s and all interaction effects are typically nonnegative. The interaction effects can assume any values.

For simplicity, we assume that only three attributes are required by a test, which means that there are three main effects, three two-way interaction effects, and one three-way interaction in the LCDM. The corresponding JAGS codes are presented in Table 6.

In the LCDM, the number of main effect parameters is $\sum_{k=1}^{K} q_{ik}$; the number of interaction effect parameters is limited by the highest number of required attributes in the Q-matrix. For example, if one test requires five attributes but no item simultaneously requires more than three attributes, then the highest way interaction in the LCDM is three rather than five. Similar to the LLM, the priors on the item parameters can be induced by defining auxiliary parameters (e.g., xlamda1).

By setting different constraints, the LCDM can be transferred into different CDMs. For example, if we set all interaction effect parameters in lines 25–28 to zeros, then the codes in Table 6 are equivalent to the codes in Table 4—namely, the LCDM reduced to the LLM:

```
lamda12[i] <- 0,
lamda13[i] <- 0,
lamda23[i] <- 0,
lamda123[i] <- 0.
```

The Higher Order Latent Structural Model

In CDMs, the number of all possible attribute patterns is typically 2^K. The unstructured latent structural model that was used in previous sections requires $2^K - 1$ structural parameters for such 2^K possible patterns and leads to a substantial computational burden when there are numerous attributes. de la Torre and Douglas (2004) proposed a solution to reduce the calculations related to the estimation of CDM parameters by involving a higher order latent structure beyond the attributes. They proposed a higher order latent structural model in

TABLE 6.
The LCDM

```
1.  model{
2.  for(n in 1:N){
3.    for(i in 1:I){
4.      for (k in 1:K){w[n, i, k] <- alpha[n, k] * Q[i, k]}
5.  eta1[n, i] <- lamda1[i] * w[n, i, 1] + lamda2[i] * w[n, i, 2] + lamda3[i] * w[n,
    i, 3]
6.  eta2[n, i] <- lamda12[i] * w[n, i, 1] * w[n, i, 2] + lamda13[i] * w[n, i, 1]
    * w[n, i, 3] + lamda23[i] * w[n, i, 2] * w[n, i, 3]
7.  eta3[n, i] <- lamda123[i] * w[n, i, 1] * w[n, i, 2] * w[n, i, 3]
8.      logit(p[n, i]) <- lamda0[i] + eta1[n, i] + eta2[n, i] + eta3[n, i]
9.      Y[n, i] ~ dbern(p[n, i])}
10.     for(k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
11.     c[n] ~ dcat(pai[1:C])}
12.     pai[1:C] ~ ddirch(delta[1:C])
13.     for(i in 1:I) {
14.       lamda0[i] ~ dnorm(-1.096, 0.25)
15.       xlamda1[i] ~ dnorm(0, 0.25) T(0, )
16.       xlamda2[i] ~ dnorm(0, 0.25) T(0, )
17.       xlamda3[i] ~ dnorm(0, 0.25) T(0, )
18.       xlamda12[i] ~ dnorm(0, 0.25)
19.       xlamda13[i] ~ dnorm(0, 0.25)
20.       xlamda23[i] ~ dnorm(0, 0.25)
21.       xlamda123[i] ~ dnorm(0, 0.25)
22.       lamda1[i] <- xlamda1[i] * Q[i, 1]
23.       lamda2[i] <- xlamda2[i] * Q[i, 2]
24.       lamda3[i] <- xlamda3[i] * Q[i, 3]
25.       lamda12[i] <- xlamda12[i] * Q[i, 1] * Q[i, 2]
26.       lamda13[i] <- xlamda13[i] * Q[i, 1] * Q[i, 3]
27.       lamda23[i] <- xlamda23[i] * Q[i, 2] * Q[i, 3]
28.       lamda123[i] <- xlamda123[i] * Q[i, 1] * Q[i, 2] * Q[i, 3]}}
```

Note. LCDM = log-linear cognitive diagnosis model.

which all attributes are assumed to be conditionally independent, given a continuous latent trait θ:

$$p_{nk} = P(\alpha_{nk}|\theta_n) = \frac{\exp(\xi_k\theta_n - \beta_k)}{1 + \exp(\xi_k\theta_n - \beta_k)}, \tag{9}$$

where p_{nk} is the probability of person n mastering attribute k, given θ. β_k and ξ_k denote the intercept and slope parameter of the kth attribute, respectively, and θ is assumed as $N(0, 1)$ for model identification. Owing to this higher order structure, the number of attribute parameters to be estimated is only $2K$ (i.e., K attribute intercept parameters and K attribute slope parameters) rather than $2^K - 1$. Because the number of parameters grows linearly, not exponentially, this formulation significantly reduces the computational burden. Theoretically, as a latent structural model, Equation 9 can be employed in any CDM. For the sake of simplicity, the DINA model is used to illustrate how to incorporate the higher order latent structural model into the DINA model to yield the higher order DINA (HO-DINA) model.

We use the codes from lines 8 through 12 to describe the method used to obtain attributes from the higher order latent structural model. Lines 13 through

TABLE 7.
The HO-DINA Model

```
1.  model{
2.    for (n in 1:N) {
3.      for (i in 1:I) {
4.        for (k in 1:K) {w[n, i, k] <- pow(alpha[n, k], Q[i, k])}
5.          eta[n, i] <- prod(w[n, i, 1:K])
6.          p[n, i] <- g[i] + (1 - s[i] - g[i]) * eta[n, i]
7.          Y[n, i] ~ dbern(p[n, i])}}
8.    for(n in 1:N){
9.      for(k in 1:K){
10.       logit(prob.a[n, k]) <- xi[k] * theta[n] - beta[k]
11.       alpha[n, k] ~ dbern(prob.a[n, k])}
12.     theta[n] ~ dnorm(0, 1)}
13.   for(k in 1:K){
14.     beta[k] ~ dnorm(mean.beta, pr.beta)
15.     xi[k] ~ dnorm(mean.xi, pr.xi) T(0, )}
16.   for (i in 1:I) {
17.     s[i] ~ dbeta(1, 1)
18.     g[i] ~ dbeta(1, 1)  T(, 1 - s[i])}}
```

15 are the priors of the latent structural parameters. According to the estimated results in previous studies (e.g., Zhan et al., 2018), the absolute values of β_k and ξ_k may be large and go up to a value of 3 or even 4. Thus, the scale parameters are suggested to be set as `mean.beta` = `0`, `mean.xi` = `0`, `pr.beta` = `0.25`, and `pr.xi` = `0.25`. Assuming higher θ values could lead to higher p_{nk}, which is not strictly necessary (e.g., if one attribute is a misconception rather than a skill; see Bradshaw & Templin, 2014), we could still restrict $\xi_k > 0$. Specifically, a truncated normal distribution is specified for `xi[k]` in Line 15 by using the `T(0, )` operator.

From the seven examples presented in Tables 1 through 7, an obvious advantage of using JAGS is that previously introduced models could be easily extended by altering a few lines of JAGS codes. In the next three sections, we further extend CDMs to address the polytomous attribute, the testlet effect, and the longitudinal data.

A DINA Model for Polytomous Attributes

Previously presented models are limited to binary attributes (i.e., mastery or nonmastery). In such a binary classification, it may be difficult to differentiate persons within the same category who master a specific attribute at different levels. For example, it could be that one person fully masters an attribute, while another person is a borderline master who is slightly above the threshold (e.g., 0.5). Thus, the polytomous attributes and the polytomous Q-matrix (Karelitz, 2004; von Davier, 2008) could be a better option. While a binary attribute is related to only two categories, a polytomous attribute is related to more than two categories (e.g., 0, 1, 2). This fine-grained sizing helps to provide a more

TABLE 8.
The RPa-DINA Model

```
model{
  for(n in 1:N){
    for(i in 1:I){
      for(k in 1:K){
      A[n,i,k] <- step(alpha[n, k] - Q[i, k])
      w[n,i,k] <- pow(A[n, i, k], Q_star[i, k])}
      eta[n,i] <- prod(w[n, i, 1:K])
      p[n,i] <- g[i] + (1 - s[i] - g[i]) * eta[n, i]
      Y[n,i] ~ dbern(p[n, i])}
  for(k in 1:K){alpha[n, k] <- all.patterns[c[n], k]}
  c[n]~dcat(pai[1:C])}
  pai[1:C] ~ ddirch(delta[1:C])
  for(i in 1:I){for(k in 1:K){Q_star[i, k] <- step(Q[i, k] - 1)}}
  for(i in 1:I){
    s[i]~dbeta(1, 1)
    g[i]~dbeta(1, 1) T(,1 - s[i])}}
```

Note. RPa-DINA = reparameterized polytomous attributes deterministic inputs, noisy “and” gate.

informative diagnosis of respondents in terms of their mastery levels. Substantively, the polytomous categories can differ for each attribute with well-defined meanings by content experts. Currently, the ordered category attribute coding (OCAC) framework (Karelitz, 2004) is used to address polytomous attributes (e.g., J. Chen & de la Torre, 2013; Zhan, Bian, & Wang, 2016). In the OCAC framework, the ordinal levels of each attribute are coded as nonnegative integers, beginning from 0 to 1 and going up to the highest level. For illustration, the reparameterized polytomous attributes DINA (RPa-DINA) model (Zhan et al., 2016) is demonstrated as an example here. The RPa-DINA model can be expressed in the following manner:

$$p_{ni} = P(Y_{ni} = 1|\alpha_n) = g_i + (1 - s_i - g_i)\eta_{ni},$$

$$\eta_{ni} = \prod_{k=1}^{K} w_{nik} = \prod_{k=1}^{K} A_{nik}{}^{q^*_{ik}}, \tag{10}$$

where α_{nk} is a polytomous variable for person n on attribute k, $\alpha_{nk} = l - 1$ if person n masters the lth level ($l = 1, \ldots, L_k$) of attribute k; moreover, we let L_k be the number of ordinal levels of attribute k. As the first level of attribute k is labeled as 0, $\alpha_{nk} = l - 1$; the polytomous Q-matrix is an $I \times K$ matrix with element $q_{ik} = l - 1$, thereby indicating the lth level of attribute k is required to answer item i correctly. Further, $A_{nik} = I\{\alpha_{nk} \geq q_{ik}\}$ is the ideal response to item i for person n on attribute k, where $I\{\cdot\}$ is an indicator function. Thus, $A_{nik} = 1$, if person n's attribute mastery level is at or above the specific attribute level that is required by item i and 0 otherwise; $q^*_{ik} = I\{q_{ik} > 0\}$ indicates whether attribute k is required by item i. Typically, the number of possible polytomous attribute patterns is $\prod_{k=1}^{K}(L_k + 1)$. The JAGS codes for the RPa-DINA model are presented in Table 8.

In line 5, `step(x)` equals 1 if `x` ≥ 0 and 0 otherwise. In line 13, `Q_star` is the $I \times K$ binary Q-matrix that was reduced from the polytomous Q-matrix by using $q^*_{ik}= I\{q_{ik} > 0\}$. When $L_k = 2$ for all attributes, the RPa-DINA model is equivalent to the DINA model for binary attributes (see Equations 1 and 2). Therefore, the codes in Table 8 can be used directly to describe the DINA model for binary attributes without any modifications.

Note that q^*_{ik} is useless for the conjunctive condensation rule (e.g., the DINA model) because $\prod_{k=1}^{K} A_{nik}{}^{q^*_{ik}} = \prod_{k=1}^{K} A_{nik}$; however, it is necessary for the disjunctive condensation rule, such as in the DINO model, $\eta_{ni} = 1 - \prod_{k=1}^{K} w_{nik} = 1 - \prod_{k=1}^{K} (1 - A_{nk})^{q^*_{ik}}$, as well as the compensatory condensation rule, such as in the LLM, $\eta_{ni} = \sum_{k=1}^{K} w_{nik} = \sum_{k=1}^{K} A_{nk} q^*_{ik}$.

A DINA Model for Testlet Design

Testlets have been widely adopted in educational and psychological tests. A testlet is a cluster of items that share a common stimulus (Wainer & Kiely, 1987). For example, in a reading comprehension test, a testlet is formed as a bundle of items based on one reading passage. Local item dependence among items within a testlet is called the testlet effect. The testlet effect could be an indication of a noise dimension. In the item response theory (IRT) framework, testlet effects are accounted for by adding a set of additional random effect parameters to standard IRT models: one for each testlet (Wainer, Bradlow, & Wang, 2007) or multiples for each testlet (Zhan, Wang, Wang, & Li, 2014). In practice, testlets can be used in cognitive diagnosis assessment. Although it is not conceptually challenging to add a set of random effect parameters into CDMs, limited efforts have been made for the development of testlet CDMs (Hansen et al., 2016; Liao & Jiao, 2016; Zhan, Li, Wang, Bian, & Wang, 2015; Zhan, Liao, & Bian, 2018).

For illustration, the RDINA model (see Table 2) is used as a template, and this method can be easily extended to the LCDM and other cases. To address the testlet effect, $\gamma_{nd(i)}$, a random effect parameter, is added to the RDINA model:

$$p_{ni} = P(Y_{ni} = 1 | \alpha_n, \gamma_{nd(i)}) = \frac{\exp(\lambda_{0,i} + \lambda_{(K),i}\eta_{ni} + \gamma_{nd(i)})}{1 + \exp(\lambda_{0,i} + \lambda_{(K),i}\eta_{ni} + \gamma_{nd(i)})}, \tag{11}$$

where $\gamma_{nd(i)}$ is assumed from a normal distribution $\gamma_{nd(i)} \sim N(0,\ \sigma^2_{\gamma_d})$, and $\sigma^2_{\gamma_d}$ indicates the magnitude of the testlet effect for testlet d. Other model parameters remain the same as those in the models illustrated above.

The number of testlets (i.e., `M`) must be specified, as does the testlet identifier vector d. The element in vector d (i.e., `d[i]`) is used to indicate the testlet that item i is associated with. It must be noted that if item i is a stand-alone item, `gamma[i, M+1]` is set to be 0, as given in line 13. For example, a test comprises

10 items, 2 testlets with 4 items associated with each testlet, and the last 2 items are stand-alone items; then, vector *d* should be set as

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

In this model, `Sigma_gamma` is the variance of testlet effect, $\sigma^2_{\gamma_d}$, to be monitored and estimated in line 16. The JAGS parameterizes the normal distribution in terms of precision—the inverse of the variance. The JAGS cannot specify an inverse-gamma distribution. Typically, a gamma prior to the inverse of the monitored parameter is specified. Thus, lines 15 and 16 specify an inverse-gamma prior on the `Sigma_gamma[m]` parameter.

In addition to the unstructured latent structural model, the higher order latent structural model (Equation 9) can be introduced (see the next section).

A DINA Model for Longitudinal Data

Providing diagnostic feedback on growth is crucial to formative decisions, such as targeted remedial instructions or interventions. Measuring individual growth or change relies on longitudinal data collected over multiple measures of achievement constructed along the growth trajectory. However, only a few studies focus on measuring growth with regard to several related attributes over multiple occasions (e.g., F. Li et al., 2016; Wang, Yang, Culpepper, & Douglas, 2018; Zhan, Jiao, Liao, & Li, in press). Unlike continuous latent traits in IRT models, the attributes in CDMs are categorical. Therefore, the methods for modeling growth in the IRT framework may not be directly extended to capture growth in the mastery of attributes.

Currently, there are two main approaches for analyzing longitudinal data in cognitive diagnosis. The first approach adopts the latent class modeling perspective (Y. Chen, Culpepper, Wang, & Douglas, 2018; F. Li et al., 2016; Wang et al., 2018), which can all be taken as a particular case or an application of the mixture hidden Markov model (Vermunt, Tran, & Magidson, 2008). The second approach adopts the IRT modeling perspective, such as the longitudinal higher order DINA (Long-DINA) model (Zhan et al., in press), which uses the variance–covariance-based method by assuming multiple continuous higher order latent traits (see Equation 9) that follow a multivariate normal distribution.

Taking into account potential local item dependence among anchor (or repeated) items and also following the description of the testlet-DINA model in Table 9, we introduce the Long-DINA model in this article. Essentially, the Long-DINA model can be taken as an extension of the testlet-DINA model by incorporating a multidimensional higher order latent structure to take into account the correlations among multiple latent attributes that are examined across different occasions. The Long-DINA model can be expressed in the following manner:

TABLE 9.
The Testlet-DINA Model

```
model{
  for(n in 1:N){
    for(i in 1:I){
      for(k in 1:K){w[n, i, k] <- pow(alpha[n, k], Q[i, k])}
      eta[n, i] <- prod(w[n, i, 1:K])
      logit(p[n, i]) <- lamda0[i] + lamdaK[i] * eta[n,i] + gamma[n, d[i]]
      Y[n, i] ~ dbern(p[n, i])}
    for(k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
    c[n] ~ dcat(pai[1:C])}
  pai[1:C] ~ ddirch(delta[1:C])
  for(n in 1:N){
    for(m in 1:M){gamma[n,m] ~ dnorm(0, pr_gamma[m])}
    gamma[n,M+1]<-0}
  for(m in 1:M){
    pr_gamma[m] ~ dgamma(1,1)
    Sigma_gamma[m] <- 1 / pr_gamma[m]}
  for(i in 1:I){
    lamda0[i]~dnorm(-1.096, 0.25)
    lamdaK[i]~dnorm(0, 0.25) T(0, )}}
```

$$\text{First level}: p_{nit} = P(Y_{nit} = 1|\alpha_{nt}, \gamma_{nd(i)}) = \frac{\exp(\lambda_{0,i,t} + \lambda_{(K),i,t}\prod_{k=1}^{K} \alpha_{nkt}^{q_{ikt}} + \gamma_{nd(i)})}{1 + \exp(\lambda_{0,i,t} + \lambda_{(K),i,t}\prod_{k=1}^{K} \alpha_{nkt}^{q_{ikt}} + \gamma_{nd(i)})};$$

$$\text{Second level}: p_{nkt} = P(\alpha_{nkt}|\theta_{nt}) = \frac{\exp(\xi_k\theta_{nt} - \beta_k)}{1 + \exp(\xi_k\theta_{nt} - \beta_k)}, \boldsymbol{\theta}_n = (\theta_{n1}, \ldots, \theta_{nT})';$$

$$\text{Third level}: \boldsymbol{\theta}_n = (\theta_{n1}, \ldots, \theta_{nT})' \sim \text{MVN}_T(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta); \tag{12}$$

where Y_{nit} denotes the response of person n to item i on occasion t; $\alpha_{nt} = (\alpha_{n1t}, \ldots, \alpha_{nKt})'$ denotes person n's attribute profile on occasion t, $\lambda_{0,i,t}$ and $\lambda_{(K),i,t}$ are the intercept and K-way interaction effect parameter for item i on occasion t, respectively, q_{ikt} is the element in the Q-matrix on occasion t, $\gamma_{nd(i)} \sim N(0,\ \sigma^2_{\gamma_d})$ is the specific dimension parameter for person n, used to account for local item dependence among anchor (or repeated) items on different occasions, θ_{nt} is person n's general ability on occasion t, and ξ_{kt} and β_{kt} are the slope and intercept parameters of attribute k on occasion t, respectively. The same attributes and the same underlying latent construct are assumed to be measured on different occasions (Bianconcini, 2012), that is, $K_t = K$. Thus, the slope and intercept parameters of the kth attribute are constrained to be constants across occasions, that is, $\xi_{kt} = \xi_k$ and $\beta_{kt} = \beta_k$. $\theta_n s$ are assumed to be independent of $\gamma_n s$. Further, $\boldsymbol{\mu}_\theta = (\mu_1, \ldots,\ \mu_T)$' is the mean vector of multidimensional higher order latent traits and $\boldsymbol{\Sigma}_\theta$ is a variance and covariance matrix. As a starting and reference point for subsequent occasions, θ_{n1} is constrained to follow a standard normal distribution.

TABLE 10.
The Long-DINA Model

```
1.  model{
2.    for(t in 1:T){
3.      for(n in 1:N){
4.        for(i in 1:I[t]){
5.          for(k in 1:K){w[n,i,k,t] <- pow(alpha[n, k, t], Q[i, k, t])}
6.          eta[n, i, t] <- prod(w[n, i, 1:K, t])
7.          logit(p[n, i, t]) <- lamda0[i,t]+lamdaK[i,t]*eta[n,i,t]+gamma[n,d[i]]
8.          Y[n, i, t] ~ dbern(p[n, i, t])}}
9.      for(n in 1:N){
10.       for(k in 1:K){
11.         logit(prob.a[n, k, t]) <- xi[k] * theta[n,t] - beta[k]
12.         alpha[n, k, t] ~ dbern(prob.a[n, k, t])}}}
13.     for(n in 1:N){theta[n,1:T] ~ dmnorm(mu_theta[1:T], pr_theta[1:T, 1:T])}
14.     for(k in 1:K){
15.       beta[k] ~ dnorm(0, 0.25)
16.       xi[k] ~ dnorm(0, 0.25) T(0, )}
17.     for(n in 1:N){
18.       for(m in 1:M){gamma[n,m] ~ dnorm(0, pr_gamma[m])}
19.       gamma[n, M+1] <- 0}
20.     for(m in 1:M){
21.       pr_gamma[m] ~ dgamma(1, 1)
22.       Sigma_gamma[m] <- 1 / pr_gamma[m]}
23.     for(i in 1:I[1]){
24.       lamda0[i,1] ~ dnorm(-1.096, 0.25)
25.       lamdaK[i,1] ~ dnorm(0, 0.25) T(0, )}
26.     lamda0[1, 2] <- lamda0[1, 1]
27.     lamda0[2, 2] <- lamda0[2, 1]
28.     lamdaK[1, 2] <- lamdaK[1, 1]
29.     lamdaK[2, 2] <- lamdaK[2, 1]
30.     for(i in 3:I[2]){
31.       lamda0[i,2] ~ dnorm(-1.096, 0.25)
32.       lamdaK[i,2] ~ dnorm(0, 0.25) T(0, )}
33.     mu_theta[1] <- 0
34.     for (t in 2:T){mu_theta[t] ~ dnorm(0, 0.5)}
35.     L_theta[1, 1] <- 1
36.     for(tt in 2:T){
37.       L_theta[tt, tt] ~ dgamma(1, 1)
38.       for(ttt in 1:(tt-1)){
39.        L_theta[tt, ttt] ~ dnorm(0, 1)
40.        L_theta[ttt, tt] <- 0}}
41.        Sigma_theta <- L_theta %*% t(L_theta)
42.       pr_theta[1:T, 1:T] <- inverse(Sigma_theta[1:T, 1:T])}
```

Note. Long-DINA = longitudinal higher order deterministic inputs, noisy “and” gate.

For illustration, we assume two occasions ($T = 2$), 10 items on each occasion, and the first 2 items ($M = 2$) on each occasion are used as anchor items. The corresponding JAGS codes are provided in Table 10.

For the test scenario specified above, `I` and `d[i]` should be set as `I = c(10, 10)` and `d = c(1,2,3,3,3,3,3,3,3,3,1,2,3,3,3,3,3,3,3,3)`. Lines 23 through 25 are prior distributions for items on the first occasion, lines 26 through 29 are used for anchor items, and lines 30 through 32 are prior distributions for nonanchor items on the second occasion.

The multivariate normal distribution in JAGS is also parameterized in the precision matrix, which is the inverse of the covariance matrix. Thus, the covariance matrix `Sigma_theta`, which needs to be monitored and estimated, is inverted in line 42; then, the resulting precision matrix `pr_theta` is used in the `dmnorm` function. Typically, inverse Wishart distributions are used to specify priors for the covariance matrices. However, an inverse Wishart prior cannot be used for $\mathbf{\Sigma}_\theta$ (i.e., `Sigma_theta`) because the variance of θ_{n1} is set to 1. To solve this problem, $\mathbf{\Sigma}_\theta$ can be reparameterized in terms of its Cholesky decomposition as $\mathbf{\Sigma}_\theta = \mathbf{\Delta}_\theta\mathbf{\Delta}_\theta'$ (Curtis, 2010; Zhan et al., 2018), where $\mathbf{\Delta}_\theta = \begin{pmatrix} 1 & 0 \\ \varphi & \psi \end{pmatrix}$ is a lower triangular matrix with positive entries on the diagonal and unrestricted off-diagonal entries, and $\mathbf{\Delta}_\theta'$ is the conjugate transpose of $\mathbf{\Delta}_\theta$. Therefore, `L_theta[tt, ttt] ~ dnorm(0, 1)` is used for $\varphi \sim N(0, 1)$ in line 39, and `L_theta[tt, tt] ~ dgamma(1, 1)` is used for $\psi \sim \text{Gamma}(1, 1)$ in line 37.

The expectation–maximization algorithm via flexMIRT (Version 3.51; Cai, 2017) was used in Zhan, Jiao, Liao, and Li (in press).[5] However, due to the restriction of the flexMIRT, multiple Q-matrices and attribute patterns from different occasions must be combined and rebuilt together in an analysis, which may lead to large computing burden. For example, if $T = 4$ and $K = 5$, then $2^{TK} =$ 1,048,576 attribute patterns need to be estimated in the flexMIRT. In contrast, due to the flexibility of the JAGS, multiple Q-matrices and attribute patterns on different occasions are used separately (e.g., `Q[i, k, t]` in Table 10). Thus, only $2^K \times T = 128$ attribute patterns must be estimated.

An Empirical Example: A Tutorial

To demonstrate how to use the JAGS codes presented in the earlier sections to analyze a real data set, fraction subtraction data from de la Torre (2009)—originally used by Tatsuoka (1990)—was analyzed. The data set contained a total of 536 people who responded to 15 items that measure five required attributes. The total number of possible attribute profiles was 32. The Q-matrix can be found in de la Torre (2009). The response data and the Q-matrix can be read in R first:

```
setwd("C:/...") #Set working directory
set.seed(12345)
library(CDM) #CDM package is only used to read the fraction subtraction data.
data(data.fraction1) #Read the fraction subtraction data and Q-matrix.
Y <- data.matrix(data.fraction1$data); View(Y)
Q <- data.matrix(data.fraction1$q.matrix); View(Q)
```

This section illustrates how to employ the R2jags package and JAGS codes to analyze the fraction subtraction data step-by-step. The DINA model and the rRUM were employed and compared. For simplicity, only the DINA model is presented for illustration. Readers can directly adapt the codes in R for other models.

Step 1: Construct `all.patterns`

Within the codes, `all.patterns` is a matrix that contains all possible attribute patterns. The `gapp` (generate all possible patterns) function given below can be used to help readers to quickly generate `all.patterns` on the basis of a self-defined Q-matrix.[6]

```
gapp <- function(q){
   K <- ncol(q)
   q.entries <- as.list(1:K)
   for(k in 1:K){q.entries[[k]] <- sort(unique(c(0,
     q[,k])))}
   attr.patt <- as.matrix(expand.grid(q.entries))}
all.patterns <- gapp(Q); View(all.patterns)
```

Step 2: Load JAGS Codes for the DINA Model

```
DINA <- function(){
  for (n in 1:N) {
    for (i in 1:I) {
      for (k in 1:K){w[n, i, k] <- pow(alpha[n, k], Q[i, k])}
      eta[n, i] <- prod(w[n, i, 1:K])
      p[n, i] <- pow((1 - s[i]), eta[n, i]) * pow(g[i], (1 -
        eta[n, i]))
      Y[n, i] ~ dbern(p[n, i])}
    for (k in 1:K) {alpha[n, k] <- all.patterns[c[n], k]}
    c[n] ~ dcat(pai[1:C])}
  pai[1:C] ~ ddirch(delta[1:C])
  for (i in 1:I) {
    s[i] ~ dbeta(1, 1)
    g[i] ~ dbeta(1, 1) %_% T( , 1 - s[i])}
##the posterior predictive model checking##
for (n in 1:N){
  for (i in 1:I){
    teststat[n,i] <- pow(Y[n, i] - p[n, i], 2)/(p[n, i] *
      (1 - p[n, i]))
    Y_rep[n,i] ~ dbern(p[n,i])
```

```
teststat_rep[n,i] <- pow(Y_rep[n, i] - p[n, i],2)/(p[n, i]
  * (1 - p[n, i]))}}
teststatsum <- sum(teststat[1:N, 1:I])
teststatsum_rep <- sum(teststat_rep[1:N, 1:I])
ppp <- step(teststatsum_rep - teststatsum)}
```

In R, to overcome incompatibility, the dummy operator %_% must be used before `T( , 1 – s[i])`. The dummy operator %_% will be eliminated before the codes are saved as a separate file. In addition, the PPMC (Gelman et al., 2014) is used to evaluate the absolute model–data fit. Posterior predictive probability (PPP) values close to 0.5 indicate that there are no systematic differences between realized and predictive values; thus, there is an adequate fit of the model to the data. In contrast, PPP values close to 0 or 1 (typically PPP value < 0.05 or PPP value > 0.95) suggest an inadequate model fit (Gelman et al., 2014). The sum of the squared Pearson residuals for person *n* and item *i* (Yan, Mislevy, & Almond, 2003) is used as a discrepancy measure to evaluate the overall fit of the model in the following manner:

$$D(Y_{ni};\alpha_n) = \sum_{n=1}^{N}\sum_{i=1}^{I}\left(\frac{Y_{ni}-p_{ni}}{\sqrt{p_{ni}(1-p_{ni})}}\right)^2,$$

where p_{ni} is defined in the same manner as in Equation 1. Note that other kinds of discrepancy measures also can be used for different purposes (Levy & Mislevy, 2016).

Step 3: Load the R2jags Package and Data

```
library(R2jags)
N = nrow(Y) #as an exercise, N = 100 can be used to reduce the
  time cost
I = nrow(Q)
K = ncol(Q)
C = nrow(all.patterns)
delta = rep(1, C)
jags.data = list("N", 'I", "K", "Y", "Q", "C", "all.pat-
  terns", "delta")
```

Step 4: Preliminary Study for Parameter Convergence

```
pre.parameters <- c("s", "g", "pai")
#s: slipping parameter
#g: guessing parameter
#pai: posterior mixing proportion
jags.inits <- NULL #Initial values are not specified.
```

```
pre.sim <- jags(data = jags.data, inits = jags.inits,
           parameters.to.save = pre.parameters, mod-
           el.file = DINA, n.chains = 2, n.iter = 1000,
           n.thin = 1, DIC = TRUE)
R_convergence <- sum(pre.sim$BUGSoutput$summary[ , 8]
  >= 1.2) == 0; R_convergence

if(R_convergence == 0){pre.sim.c <- autojags(pre.sim,
  Rhat = 1.2, n.update = 30)
pre.sim.c$n.iter}; pre.sim.c$n.iter
if(R_convergence == 1){pre.sim$n.iter}; pre.sim$n.iter
```

A preliminary study was conducted to obtain a necessary number of iterations to achieve convergence. In Bayesian CDMs, item parameters and mixing proportions are typically checked for convergence (Culpepper, 2015a; Zhan et al., 2018). In the preliminary study, two Markov chains (`n.chains = 2`) were used with `n.iter = 1000` iterations per chain, with the first half of iterations in each chain as burn-in (in default); the thinning interval was set to be `n.thin = 1` (i.e., without thinning).[7] Finally, the remaining half of the iterations were used for model parameter inferences. The potential scale reduction factor, $\hat{R}$, as modified by Brooks and Gelman (1998), was computed to assess the convergence of every parameter. Values of $\hat{R}$ less than 1.1 or 1.2 indicate convergence (Brooks & Gelman, 1998; de la Torre & Douglas, 2004). If any parameter estimate does not reach convergence (i.e., `R_convergence` is `FALSE`), updating would automatically continue until all values of $\hat{R}$ become less than 1.2. More stringent rules for convergence can be used by setting `Rhat = 1.1` or `1.05` in the `autojags` function. `pre.sim$n.iter` or `pre.sim.c $n.iter` is used to show how many iterations are needed to converge. In this example, based on the rule of $\hat{R} < 1.2$, 1,000 iterations are necessary for convergence.

Step 5: Parameter Estimation

```
jags.parameters <- c("s", "g", "c", "pai", "ppp")
#c: estimated attribute pattern of each respondent
#ppp: posterior predictive probability
jags.inits <- NULL #Initial values are not specified.

time1 = as.POSIXlt(Sys.time())
sim <- jags(data = jags.data, inits = jags.inits, parame-
       ters.to.save = jags.parameters, model.file =
       DINA, n.chains = 2, n.iter = 10000, n.burnin =
       5000, n,thin = 1, DIC = TRUE)
```

```
time2 = as.POSIXlt(Sys.time())
use.time = difftime(time2, time1, units="secs")
```

Although the preliminary study indicates that a burn-in of 1,000 iterations is adequate, we employed a burn-in of 5,000 iterations in this study to ensure the stability of the results. Two Markov chains were used with 10,000 iterations per chain, and the first 5,000 iterations in each chain were excluded as burn-in. The thinning interval was set to be 1. Finally, 10,000 iterations were used for model parameter inferences. The `use.time` function was used to compute the overall running time for parameter estimation.[8]

Step 6: Save Estimated Parameters

```
sim1 <- sim$BUGSoutput
E.pattern <- cbind(sim1$median$c)
E.itempar <- cbind(sim1$mean$g, sim1$mean$s, sim1$sd$g,
  sim1$sd$s)
write.table(E.pattern, "pattern_DINA.txt")
write.table(E.itempar, "itempar_DINA.txt")
write.table(sim1$summary, "summary_DINA.txt")
write.table(sim1$DIC, "DIC_DINA.txt")
write.table(use.time, "time_DINA.txt")
write.table(sim1$mean$deviance, "deviance_DINA.txt")
write.table(sim1$mean$ppp, "ppp_DINA.txt")
```

The file "summary_DINA.txt" presents the summary statistics based on the sampled values of all monitored parameters. Taking the mixing proportions as an example, Table 11 presents the first three estimated mixing proportions, `pai`, of 32 possible patterns. Note that `pai[1]` to `pai[32]` correspond to the 32 rows in `all.patterns`, respectively. The posterior mean and the standard deviation can be used as the point estimates of the mixing proportions and their standard errors. The values corresponding to the column labeled 2.5% and the column labeled 97.5% can be used as the 95% credible interval. Further, the column labeled `Rhat` lists the $\hat{R}$, and the column labeled `n.eff` lists the effective number of simulation draws, which can be viewed as the effective sample size for a posterior distribution upon which inferences were based. In addition, the trace plots for the mixing proportions can be requested by inputting `traceplot (sim, varname = "pai").`

The file "ppp_DINA.txt" contains the PPP value of the DINA model for these data. In addition, the file "deviance_DINA.txt" extracts the posterior mean of `deviance` in the MCMC samples, that is, -2 log likelihood (-2LL), which can be used to compute a certain relative model–data fit—for example, the Akaike's (1974) information criterion (AIC) index. The file "DIC_DINA.txt" provides the

TABLE 11.
Sample Output Results of Mixing Proportions for the DINA Model

Parameters	Mean	*SD*	2.5%	25%	50%	75%	97.5%	Rhat	n.eff
pai[1]	.018	.016	.000	.006	.013	.024	.060	1.004	440
pai[2]	.009	.008	.000	.003	.007	.013	.028	1.001	3,200
pai[3]	.017	.015	.000	.006	.013	.025	.058	1.004	1,000

Note. *SD* = standard deviation; Rhat = potential scale reduction factor; n.eff = the effective sample size.

TABLE 12.
Model Fit and Computing Time in the Empirical Example

Model	ppp	NP	−2LL	AIC	DIC	Time
DINA	.553	91	5,451.89	5,542.89	6,336.40	1,534.97
rRUM	.701	107	4,843.06	4,950.06	6,302.98	3,369.94

Note. NP = number of estimated parameters; −2LL = −2 log likelihood; AIC = Congdon's version of Akaike's information criterion; DIC = deviance information criterion; Time = overall computing time for two Markov chains (in seconds).

deviance information criterion (DIC) index (Spiegelhalter, Best, Carlin, & Van der Linde, 2002), where DIC $= \bar{D} + p_e = \bar{D} + \mathrm{var}(D)/2$—namely, the effective number of parameters (p_e) was computed by $p_e = \mathrm{var}(D)/2$ (Gelman, Carlin, Stern, & Rubin, 2003; Su & Yajima, 2015), where D is the deviance and $\bar{D}$ is the posterior mean of deviance (i.e., −2LL, the value in the file of "deviance_DINA.txt"). Note that in the Bayesian analysis, the AIC can be defined as AIC $= \bar{D} + p$ (Congdon, 2003), where p is the number of estimated parameters. In addition, as mentioned by Gelman et al. (2014), the Bayesian information criterion (BIC; Schwarz, 1978) has a different goal from AIC and DIC—"BIC is not intended to predict out-of-sample model performance but rather is designed for other purposes, we do not consider it further here" (p. 175).[9] In addition, the file "pattern_DINA.txt" is for the estimated attribute patterns. As a categorical value, the posterior mode of `c` is treated as the estimated value in this study, and the value of `c`—that is, 1 to 32—corresponds to the 32 rows in `all.patterns`, respectively. The file "time_DINA.txt" summarizes the overall computing time.

Table 12 presents the model fit comparison between the DINA and rRUM models to the fraction subtraction data. The rRUM model was identified as having a better fit based on AIC, DIC, and a PPP value of 0.701. It took approximately 1,535 and 3,370 seconds to run the DINA and rRUM models, respectively.

Table 13 presents the estimates of item parameters for two models. Table 14 presents the estimates of the mixing proportions for the two models. It must be

TABLE 13.
Estimates of Item Parameters in the Empirical Example

	DINA		rRUM					
Item	g	s	π^*	r_1^*	r_2^*	r_3^*	r_4^*	r_5^*
$\frac{3}{4}-\frac{3}{8}$	.011 (.012)	.277 (.024)	.894 (.020)	.019 (.016)				
$3\frac{1}{2}-2\frac{3}{2}$	.214 (.025)	.121 (.021)	.890 (.020)	.539 (.107)	.861 (.106)	.842 (.113)	.254 (.050)	
$\frac{6}{7}-\frac{4}{7}$	.146 (.053)	.039 (.010)	.969 (.010)	.498 (.039)				
$3-2\frac{1}{5}$	.127 (.019)	.137 (.029)	.894 (.030)	.039 (.040)	.806 (.153)	.608 (.212)	.769 (.119)	.150 (.079)
$3\frac{7}{8}-2$	.217 (.063)	.249 (.023)	.746 (.023)			.419 (.063)		
$4\frac{4}{12}-2\frac{7}{12}$	.036 (.012)	.229 (.027)	.801 (.027)	.110 (.083)	.421 (.211)	.645 (.219)	.114 (.043)	
$4\frac{1}{3}-2\frac{4}{3}$	.076 (.016)	.080 (.018)	.939 (.016)	.614 (.151)	.314 (.177)	.527 (.226)	.116 (.036)	
$\frac{11}{8}-\frac{1}{8}$	.174 (.045)	.051 (.014)	.948 (.014)	.940 (.050)	.060 (.046)			
$3\frac{4}{5}-3\frac{2}{5}$	.108 (.037)	.062 (.014)	.957 (.012)	.786 (.058)		.093 (.048)		
$2-\frac{1}{3}$	.170 (.023)	.075 (.021)	.930 (.023)	.132 (.068)		.338 (.152)	.738 (.106)	.362 (.088)
$4\frac{5}{7}-1\frac{4}{7}$	.123 (.035)	.102 (.017)	.915 (.017)	.843 (.062)		.073 (.039)		
$7\frac{3}{5}-\frac{4}{5}$	.034 (.013)	.137 (.022)	.881 (.022)	.661 (.170)		.026 (.028)	.088 (.032)	
$4\frac{1}{10}-2\frac{8}{10}$	.137 (.021)	.161 (.024)	.856 (.023)	.374 (.123)	.320 (.230)	.160 (.157)	.447 (.070)	
$4-1\frac{4}{3}$	.025 (.010)	.203 (.033)	.820 (.036)	.090 (.101)	.313 (.260)	.276 (.244)	.228 (.079)	.120 (.075)
$4\frac{1}{3}-1\frac{5}{3}$	.015 (.007)	.185 (.025)	.838 (.024)	.693 (.180)	.112 (.131)	.256 (.198)	.015 (.014)	

Note. Posterior standard deviations (i.e., standard errors) are in parentheses. DINA = deterministic inputs, noisy "and" gate; rRUM = reduced reparameterized unified model.

TABLE 14.
Estimates of Mixing Proportions in the Empirical Example

Pai[i]	Attribute Patterns	DINA	rRUM
pai[1]	0 0 0 0 0	.017 (.012)	.032 (.025)
pai[2]	1 0 0 0 0	.009 (.007)	.005 (.005)
pai[3]	0 1 0 0 0	.017 (.015)	.006 (.005)
...	...	...	...
pai[30]	1 0 1 1 1	.005 (.004)	.003 (.003)
pai[31]	0 1 1 1 1	.009 (.009)	.010 (.008)
pai[32]	1 1 1 1 1	.350 (.022)	.328 (.024)

Note. Posterior standard deviations (i.e., standard errors) are in parentheses; the middle 26 patterns are omitted. DINA = deterministic inputs, noisy "and" gate; rRUM = reduced reparameterized unified model.

noted that the comparison between these two models is beyond the scope of this article. Thus, no further explanation of the results is provided.

Summary

This article presents a systematic introduction to using JAGS for Bayesian CDM estimation. Several JAGS codes are presented to fit a few representative CDMs. The unstructured latent structural model and the higher order latent structural model are both introduced. Further, this article demonstrates how to extend these models to polytomous attributes, the testlet effect, and longitudinal data. Finally, an empirical example is presented to illustrate how the R2jags package must be illustrated to run the JAGS codes.

As a tutorial, this article has its limitations. First, only selected CDMs were demonstrated in this tutorial. Thus, the readers are encouraged to consult other sources for other model extensions. Second, certain emerging research topics are not included, such as the Q-matrix estimation (Y. Chen, Culpepper, Chen et al., 2018; Chung & Johnson, 2018), joint CDMs for response accuracy and response times (e.g., Zhan et al., 2018), and CDMs for polytomous scoring items (Ma & de la Torre, 2016b; von Davier, 2008). Third, only the R2jags package was used to call JAGS; however, other R packages such as the rjags (Plummer, Stukalov, & Denwood, 2016) and jagsUI (Kellner, 2017) can also be used. Fourth, the computing time tended to be rather long, particularly for large-scale tests with a large sample size. Thus, it is desirable to develop more effective Bayesian estimation programs to increase the efficiency in model parameter estimation for the new models, such as the dina (Culpepper, 2015b) package in R. In addition, a new Bayesian software package named Stan (Carpenter et al., 2016) has been developed. Stan uses the no-U-turn sampler (Hoffman & Gelman, 2014), an extension to the Hamiltonian Monte Carlo (Neal, 2011) algorithm. The Hamiltonian Monte

Carlo is considerably faster than the Gibbs sampler, which is used in JAGS. Further exploration of using Stan to fit Bayesian CDMs would be valuable (e.g., Lee, 2016).

Overall, given the increasing number of applications of the Bayesian MCMC algorithm in fitting numerous CDMs, JAGS has the potential of becoming a popular tool in the field. Further, it is hoped that researchers can adapt the codes presented in this article for their own testing situations.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Natural Science Foundation of Zhejiang Province, China (Grant No. LY16C090001).

ORCID iD

Peida Zhan https://orcid.org/0000-0002-6890-7691

Notes

1. A few minor differences between just another Gibbs sampler (JAGS) and OpenBUGS (or WinBUGS) can be found in the JAGS manual (Plummer, 2015).
2. Uninformative priors are designed to provide minimal information regarding the parameter being estimated and to allow the data to dominate the analysis or determination of the posterior distribution, which deems them less "subjective."
3. To a certain extent, informative priors reflect data analysts' beliefs regarding estimated parameters. Thus, the specification of the scale parameters of the β prior distribution still depends on the situation.
4. A description of the gamma distribution can be found in the comments in Table 9.
5. To our knowledge, currently, among existing stand-alone software and packages, only the flexMIRT can be utilized to fit the longitudinal higher order deterministic inputs, noisy "and" gate model.
6. The `gapp` function can be used for both binary and polytomous attributes.
7. Occasionally, in order to avoid high autocorrelations between the sampling distributions or to take up less space in memory for large-scale data, the thinning interval can be set to 5 or a larger number.
8. All runs reported in this article were on an msi GT72VR 6RD DOMINATOR laptop with a 2.6GHz Intel Core i7 6700HQ CPU, 2133MHz 32GB of memory, and 256GB SanDisk z400s Solid State Drive.

9. Note that if the reader wishes to report the Bayesian information criterion (BIC) in the Bayesian analysis, the BIC can be defined as $\mathrm{BIC} = \bar{D} + (\log N - 1)p$.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723. doi:10.1109/TAC.1974.1100705

Bianconcini, S. (2012). A general multivariate latent growth model with applications to student achievement. *Journal of Educational and Behavioral Statistics*, *37*, 339–364.

Bolt, D., Chen, H., DiBello, L., Hartz, S., Henson, R., Roussos, L., . . . Templin, J. (2008). *The Arpeggio Suite: Software for cognitive skills diagnostic assessment* [Computer software and Manual]. St. Paul, MN: Assessment Systems.

Box, G. E. P., & Tiao, G. C. (1973). *Bayesian inference in statistical analysis*. Reading, MA: Addison-Wesley.

Bradshaw, L., & Templin, J. (2014). Combining item response theory and diagnostic classification models: A psychometric model for scaling ability and diagnosing misconceptions. *Psychometrika*, *79*, 403–425.

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*, 434–455. doi:10.2307/1390675

Cai, L. (2017). *flexMIRT® version 3.51: Flexible multilevel multidimensional item analysis and test scoring* [Computer software]. Chapel Hill, NC: Vector Psychometric Group.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., . . . Riddell, A. (2016). Stan: A probabilistic programming language. *Journal of Statistical Software*, *20*, 1–37.

Chen, J., & de la Torre, J. (2013). A general cognitive diagnosis model for expert-defined polytomous attributes. *Applied Psychological Measurement*, *37*, 419–437.

Chen, Y., Culpepper, S. A., Chen, Y., & Douglas, J. (2018). Bayesian estimation of the DINA Q matrix. *Psychometrika*, *83*, 89–108.

Chen, Y., Culpepper, S. A., Wang, S., & Douglas, J. (2018). A hidden Markov model for learning trajectories in cognitive diagnosis with application to spatial rotation skills. *Applied Psychological Measurement*, *42*, 5–23.

Congdon, P. (2003). *Applied Bayesian modelling*. New York, NY: John Wiley.

Chung, M., & Johnson, M. (2018). An MCMC algorithm for estimation the Q-matrix in a Bayesian framework. *arXiv preprint arXiv:1802.02286*. Retrieved from https://arxiv.org/abs/1802.02286

Culpepper, S. A. (2015a). Bayesian estimation of the DINA model with Gibbs sampling. *Journal of Educational and Behavioral Statistics*, *40*, 454–476.

Culpepper, S. A. (2015b). *dina: Bayesian estimation of DINA model* (R package version 1.0.1). Retrieved from http://CRAN.R-project.org/package=dina

Culpepper, S. A., & Hudson, A. (2018). An improved strategy for Bayesian estimation of the reduced reparameterized unified model. *Applied Psychological Measurement*, *42*, 99–115.

Curtis, S. M. (2010). BUGS code for item response theory. *Journal of Statistical Software*, *36*, 1–34.

DeCarlo, L. T. (2012). Recognizing uncertainty in the Q-matrix via a Bayesian extension of the DINA model. *Applied Psychological Measurement*, *36*, 447–468. doi:10.1177/0146621612449069

de la Torre, J. (2009). DINA model and parameter estimation: A didactic. *Journal of Educational and Behavioral Statistics*, *34*, 115–130.

de la Torre, J. (2011). The generalized DINA model framework. *Psychometrika*, *76*, 179–199. doi:10.1007/s11336-011-9207-7

de la Torre, J., & Douglas, J. (2004). Higher-order latent trait models for cognitive diagnosis. *Psychometrika*, *69*, 333–353. doi:10.1007/BF02295640

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis* (3rd ed.). Boca Raton, FL: CRC Press.

Gelman, A., Carlin, J.B., Stern, H.S., & Rubin, D.B. (2003). *Bayesian data analysis* (2nd ed.). Boca Raton, FL: CRC Press.

George, A. C., Robitzsch, A., Kiefer, T., Gross, J., & Uenlue, A. (2016). The R package CDM for cognitive diagnosis models. *Journal of Statistical Software*, *74*, 1–24.

Haertel, E. H. (1989). Using restricted latent class models to map the skill structure of achievement items. *Journal of Educational Measurement*, *26*, 301–321. doi:10.1111/j.1745-3984.1989.tb00336.x

Hansen, M., Cai, L., Monroe, S., & Li, Z. (2016). Limited-information goodness-of-fit testing of diagnostic classification item response models. *British Journal of Mathematical and Statistical Psychology*, *69*, 225–252. doi:10.1111/bmsp.12074

Hartz, S. M. (2002). *A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality*. (Unpublished doctoral dissertation). University of Illinois at Urbana-Champaign, Champaign, IL.

Henson, R., Templin, J., & Willse, J. (2009). Defining a family of cognitive diagnosis models using log-linear models with latent variables. *Psychometrika*, *74*, 191–210.

Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*, 1593–1623.

Huang, H.-Y., & Wang, W.-C. (2014). The random-effect DINA model. *Journal of Educational Measurement*, 51, 75–97.

Jiang, H. (1996). *Applications of computational statistics in cognitive diagnosis and IRT modeling*. (Unpublished doctoral dissertation). University of Illinois at Urbana-Champaign, Champaign, IL.

Junker, B. W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement*, *25*, 258–272.

Karelitz, T. M. (2004). *Ordered category attribute coding framework for cognitive assessments*. (Unpublished doctoral dissertation). University of Illinois at Urbana-Champaign, Champaign, IL.

Kellner, K. (2017). *jagsUI: A wrapper around "rjags" to streamline "JAGS" analyses* (R package version 1.4.9). Retrieved from http://CRAN.R-project.org/package=jagsUI

Köhn, H.-F., & Chiu, C.-Y. (2016). A proof of the duality of the DINA model and the DINO model. *Journal of Classification*, *33*, 171–184.

Lee, S. Y. (2016). *Cognitive diagnosis model: DINA model with independent attributes*. Retrieved from http://mc-stan.org/documentation/case-studies/dina_independent.html

Levy, R., & Mislevy, R. J. (2016). *Bayesian psychometric modeling*. Boca Raton, FL: CRC Press.

Li, F., Cohen, A., Bottge, B., & Templin, J. (2016). A latent transition analysis model for assessing change in cognitive skills. *Educational and Psychological Measurement*, *76*, 181–204.

Li, X. M., & Wang, W.-C. (2015). Assessment of differential item functioning under cognitive diagnosis models: The DINA model example. *Journal of Educational Measurement*, *52*, 28–54.

Liao, M., & Jiao, H. (2016, April). *A combination of diagnostic classification model and item response theory model with testlet effect*. Paper presented at the International Objective Measurement Workshop, Washington, DC.

Lunn, D. J., Thomas, A., Best, N., & Spiegelhalter, D. (2000). WinBUGS-a Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, *10*, 325–337.

Ma, W., & de la Torre, J. (2016a). *GDINA: The generalized DINA model framework* (R package version 1.2.1). Retrieved from http://CRAN.R-project.org/package=GDINA

Ma, W., & de la Torre, J. (2016b). A sequential cognitive diagnosis model for polytomous responses. *British Journal of Mathematical and Statistical Psychology*, *69*, 253–275. doi:10.1111/bmsp.12070

Macready, G. B., & Dayton, C. M. (1977). The use of probabilistic models in the assessment of mastery. *Journal of Educational and Behavioral Statistics*, *2*, 99–120.

Martin, A. D., Quinn, K. M., & Park, J. H. (2011). Mcmcpack: Markov chain Monte Carlo in R. *Journal of Statistical Software*, *42*, 1–21.

Maris, E. (1999). Estimating multiple classification latent class models. *Psychometrika*, *64*, 187–212.

Muthén, L. K., & Muthén, B. O. (2010). *Mplus: Statistical analysis with latent variables: User's guide* (pp. 1998–2007). Los Angeles, CA: Author.

Neal, R. M. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, *2*, 113–162.

Plummer, M. (2015). *JAGS Version 4.0.0 user manual*. Lyon, France. Retrieved from http://sourceforge.net/projects/mcmc-jags/

Plummer, M., Stukalov, A., & Denwood, M. (2016). *rjags: Bayesian graphical models using MCMC* (R package version 4-6). Retrieved from http://CRAN.R-project.org/package=rjags

R Core Team. (2016). *R: A language and environment for statistical computing*. Vienna, Austria: The R Foundation for Statistical Computing.

Rupp, A., Templin, J., & Henson, R. (2010). *Diagnostic measurement: Theory, methods, and applications*. New York, NY: Guilford Press.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*, 461–464.

Sinharay, S., & Almond, R. G. (2007). Assessing fit of cognitive diagnostic models: A case study. *Educational and Psychological Measurement*, *67*, 239–257.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society, Series B*, *64*, 583–616.

Spiegelhalter, D., Thomas, A., Best, N., & Lunn, D. (2014). *OpenBUGS Version 3.2.3 user manual*. Helsinki, Finland. Retrieved from http://www.openbugs.net/w/Manuals/

Su, Y.-S., & Yajima, M. (2015). *R2jags: Using R to run "JAGS"* (R package version 0.5-7). Retrieved from http://CRAN.R-project.org/package=R2jags

Tatsuoka, K. K. (1983). Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, *20*, 345–354.

Tatsuoka, K. K. (1990). Toward an integration of item-response theory and cognitive error diagnosis. In N. Frederiksen, R. Glaser, A. Lesgold, & M. Safto (Eds.), *Monitoring skills and knowledge acquisition* (pp. 453–488). Hillsdale, NJ: Erlbaum.

Templin, J., & Henson, R. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological Methods*, *11*, 287–305.

Templin, J., & Hoffman, L. (2013). Obtaining diagnostic classification model estimates using Mplus. *Educational Measurement: Issues and Practice*, *32*, 37–50.

Vermunt, J. K., Tran, B., & Magidson, J. (2008). Latent class models in longitudinal research. In S. Menard (Ed.), *Handbook of longitudinal research: Design, measurement, and analysis* (pp. 373–385). Burlington, MA: Elsevier.

von Davier, M. (2005). *mdltm: Software for the general diagnostic model and for estimating mixtures of multidimensional discrete latent traits models* [Computer software]. Princeton, NJ: ETS.

von Davier, M. (2008). A general diagnostic model applied to language testing data. *British Journal of Mathematical and Statistical Psychology*, *61*, 287–307.

Wagenmakers, E.-J., Lee, M. D., Lodewyckx, T., & Iverson, G. (2008). Bayesian versus frequentist inference. In H. Hoijtink, I. Klugkist, & P. A. Boelen (Eds.), *Bayesian evaluation of informative hypotheses* (pp. 181–207). New York, NY: Springer.

Wainer, H., Bradlow, E. T., & Wang, X. (2007). *Testlet response theory and its applications*. New York, NY: Cambridge University Press.

Wainer, H., & Kiely, G. (1987). Item clusters and computerized adaptive testing: A case for testlets. *Journal of Educational Measurement*, *24*, 185–202.

Wang, S., Yang, Y., Culpepper, S. A., & Douglas, J. A. (2018). Tracking skill acquisition with cognitive diagnosis models: A higher-order, hidden Markov model with covariates. *Journal of Educational and Behavioral Statistics*, *43*, 57–87.

Yan, D., Mislevy, R. J., & Almond, R. G. (2003). *Design and analysis in a cognitive assessment* (ETS Research Report Series, RR-03-32). Princeton, NJ: ETS.

Zhan, P., Bian, Y., & Wang, L. (2016). Factors affecting the classification accuracy of reparametrized diagnostic classification models for expert-defined polytomous attributes. *Acta Psychologica Sinica*, *48*, 318–330.

Zhan, P., Jiao, H., & Liao, D. (2018). Cognitive diagnosis modelling incorporating item response times. *British Journal of Mathematical and Statistical Psychology*, *71*, 262–286.

Zhan, P., Jiao, H., Liao, M., & Bian, Y. (2018). Bayesian DINA modeling incorporating within-item characteristic dependency. *Applied Psychological Measurement*. Advance online publication. doi:10.1177/0146621618781594

Zhan, P., Jiao, H., Liao, D., & Li, F. (in press). A longitudinal higher-order diagnostic classification model. *Journal of Educational and Behavioral Statistics*.

Zhan, P., Li, X., Wang, W.-C., Bian, Y., & Wang, L. (2015). The multidimensional testlet-effect cognitive diagnostic models. *Acta Psychologica Sinica*, *47*, 689–701.

Zhan, P., Liao, M., & Bian, Y. (2018). Joint testlet cognitive diagnosis modeling for paired local item dependence in response times and response accuracy. *Frontiers in Psychology*, *9*, 607.

Zhan, P., Wang, W.-C., Wang, L., & Li, X. (2014). The multidimensional testlet-effect Rasch model. *Acta Psychologica Sinica*, *46*, 1208–1222.

Authors

PEIDA ZHAN is an assistant professor at College of Teacher Education, Zhejiang Normal University, No. 688 Yingbin Road, Jinhua, Zhejiang Province, 321004, China; email: pdzhan@gmail.com. His research interests involve theoretical and applied development in latent variable modeling such as cognitive diagnosis modeling, item response theory modeling, and response times modeling. Currently, he focuses on the assessment for learning in scientific literacy and developing methods of process data analysis in technology-enhanced innovative assessment.

HONG JIAO is an associate professor at Department of Human Development and Quantitative Methodology, 1230C Benjamin Building, University of Maryland, College Park, MD 20742, USA; email: hjiao@umd.edu. Her research interests are Rasch modeling, multilevel and mixture item response theory (IRT) modeling, testlet response theory models, process data in educational and psychological measurement, Bayesian estimation of IRT models, cognitive diagnosis, technology-enhanced innovative assessment, and computerized classification test.

KAIWEN MAN is a PhD candidate at Department of Human Development and Quantitative Methodology, 1230 Benjamin Building, University of Maryland, College Park, MD 20742, USA; email: kman@umd.edu. His primary research interests include data mining methods, test security, response time modeling, eye-tracking modeling, and multidimensional IRT.

LIJUN WANG is a professor at College of Teacher Education, Zhejiang Normal University, No. 688 Yingbin Road, Jinhua, Zhejiang Province, 321004, China; email: frankwlj@163.com. Her research interests involve the measurement models such as cognitive diagnosis models, Rasch models in educational and psychological measurement, and psychometrics in large-scale tests.

Manuscript received August 12, 2017
First revision received May 28, 2018
Second revision received October 1, 2018
Third revision received December 24, 2018
Accepted December 27, 2018