Modular Open-Source Software for Item Factor Analysis

Joshua N. Pritikin
University of Virginia
Micheal D. Hunter
University of Oklahoma
Steven Boker
University of Virginia

Abstract

This paper introduces an Item Factor Analysis (IFA) module for `OpenMx`, a free, open-source, and modular statistical modeling package that runs within the `R` programming environment on GNU/Linux, Mac OS X, and Microsoft Windows. The IFA module offers a novel model specification language that is well suited to programmatic generation and manipulation of models. Modular organization of the source code facilitates the easy addition of item models, item parameter estimation algorithms, optimizers, test scoring algorithms, and fit diagnostics all within an integrated framework. Three short example scripts are presented for fitting item parameters, latent distribution parameters, and a multiple group model. The availability of both IFA and structural equation modeling in the same software is a step toward the unification of these two methodologies.

Keywords: Item Factor Analysis; Item Response Theory; modular; open-source

---

Joshua N. Pritikin, Department of Psychology, University of Virginia; Michael D. Hunter, Department of Psychology, University of Oklahoma; Steven Boker, Department of Psychology, University of Virginia.

We acknowledge the `OpenMx` team for helpful feedback on the IFA module and the preparation of this manuscript.

Correspondence concerning this article should be addressed to Joshua N. Pritikin, Department of Psychology, University of Virginia, 485 McCormick Rd, Gilmer Hall Room 102 Charlottesville, VA 22904. E-mail: jpritikin@virginia.edu

Introduction

When a test consisting of items is administered, it is desirable to understand what those items measure. Furthermore, if a test is to be summarized as a single score, the items should assess a single latent dimension. This is especially important for computer adaptive testing because each examinee may receive a different set of items intended to measure the same latent ability. These issues have received attention in the literature since at least the 1950s. However, it was the publication of the marginal maximum likelihood algorithm (MML; Bock & Aitkin, 1981) and the increasing availability of computing power that opened the way for efficient analysis of practical problems grounded with a sound theoretical footing. With demonstrations of the flexibility and sensitivity of MML for modeling tests and questionnaires (e.g., Bock, Gibbons, & Muraki, 1988), full-information item factor analysis (IFA) has grown in popularity.

A wide variety of software is available for estimation of IFA models. Examples include ConQuest (M. L. Wu, Adams, Wilson, & Haldane, 2007), EQSIRT (E. J. C. Wu & Bentler, 2012), flexMIRT (Cai, 2012), IRTPRO (Cai, Thissen, & du Toit, 2011), and `mirt` (Chalmers, 2012). In light of the many excellent choices already available, it may be surprising that there is an opportunity for new IFA software. Even so, the present article introduces the availability of IFA software that is substantially different from what is currently available.

Why new software for IFA?

Modular design is an important software attribute that has been neglected by current IFA software. Modular software is written such that each section of code operates independently and is accessed via a well-defined interface. A benefit of modularity is that many programmers can work on the code simultaneously as long as each module maintains the expected behavior of its interface. Our new IFA software is itself a module within `OpenMx` (Boker et al., 2011), a free and open-source software originally designed for structural equation modeling (SEM). `OpenMx` runs inside the `R` statistical programming environment (R Core Team, 2014) in heterogeneous computing environments. Similar to the `OpenMx` style of model specification for SEM, the IFA module offers a novel model specification language.

Open-Source

`OpenMx` is open-source; hence, the source code is available for everybody to view, modify, and use. In order to help organize a community around the project, the `OpenMx` team maintains a web site (http://openmx.psyc.virginia.edu) that hosts binary and source versions of the software and several forms of tutorials and reference documentation. Help with `OpenMx` is available on the web site from discussion forums and a community-maintained Wiki.

`OpenMx` is not the only open-source software for IFA. The R package `mirt` is also open-source. In fulfillment of some of the claimed benefits of open-source software, a friendly exchange of code and ideas has taken place between `OpenMx` and `mirt` (Pritikin, 2014). The `OpenMx` team welcomes contributors and encourages cross-pollination between implementations. Of course, open-source software is peer-reviewed. In many software engineering projects, peer-review has led to more reliable and higher quality software in comparison to a closed-source approach (Aberdour, 2007).

Heterogeneous Computing Environments

`OpenMx` runs on a variety of operating systems including GNU/Linux, Mac OS X, and Microsoft Windows. `OpenMx` scripts that are written on one operating system can be used on other operating systems without modification. This platform-independence is useful in today's heterogeneous computing environments, where each researcher on a team may have a different preferred computing platform. Moreover, since `OpenMx` is free, no license obligations hinder the use of `OpenMx` on computer clusters or exotic supercomputers.

Model Specification

While IFA software has grown more capable, model specification generally follows a structure that have changed very little since the 1980s. Specification of IFA models is generally accomplished with a serially ordered script written in a domain specific language unique to a particular software package. One of the key innovations in the design of `OpenMx` was the recognition that models can be specified directly in `R`, leveraging the fact that `R` is interactive and offers a full programming language. One may use `OpenMx` without changing one's conception of model building, engaging in programming minimally. However, consider the challenge of specifying nominal item models for testlets.

The nominal model (e.g., Thissen, Cai, & Bock, 2010) is defined as,

$$\begin{aligned}\boldsymbol{a} &= T_a\boldsymbol{\alpha}\\ \boldsymbol{c} &= T_c\boldsymbol{\gamma}\\ \Pr(\text{pick} = k|\boldsymbol{s}, \boldsymbol{\theta}, a_k, c_k) &= C\ \frac{1}{1+\exp(-(\boldsymbol{s}\boldsymbol{\theta}a_k + c_k))}\end{aligned}$$

where the kth entry of a and c are the result of multiplying two vectors of free parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ by fixed matrices T_a and T_c, respectively; a_0 and c_0 are fixed to 0 for identification; and C is a normalizing factor such that $\sum_k \Pr(\text{pick} = k) = 1$. For locally dependent items that share a common stimulus, Thissen, Steinberg, and Mooney (1989) suggested fitting a nominal item model to their sum-score. Orthogonal polynomial contrasts were used for the T_a and T_c matrices. To avoid overfitting, a proportion of the α_k and γ_k parameters were fixed to zero and the remainder estimated as free parameters, typically obtaining satisfactory fit with a less than full-rank model.

The specification of such nominal items is somewhat labor intensive. The transformation matrices, T_a and T_c, depend on the number of outcomes and the proportion of free parameters may be controlled by freeing or fixing individual parameters. A function is given in Appendix A that takes as arguments a dataset, the item name, the number of factors, the proportion of full-rank of $\boldsymbol{\alpha}$, and proportion of full-rank of $\boldsymbol{\gamma}$. This is a simpler interface because, conceptually, these are the form of the parameters that a user wants to control and the user has direct control of them. Granted, there is nothing that precludes an analyst from writing a similar function for any IFA software package. However, this is often regarded as extra work and is rarely, if ever, done.

Automation of the specification of nominal testlet items is one example of the use of programming to ease a modeling task. One of the main contributions of `OpenMx` has been to encourage treatment of statistical models and their components as things that can be generated and manipulated within a programming environment. The `OpenMx` IFA module also uses this style of model specification. Not all users will prefer to exploit such a low-level user interface. Our goal is to extend the range of utility in both directions, toward both expert and novice users. Sophisticated users are empowered to build non-standard models or to simplify application of routine analyses by developing higher level interfaces. For example, the `metaSEM` package (Cheung, 2014) simplifies the specification of `OpenMx` models for meta-analysis. A similar package could simplify the specification of nominal testlets to make the technique simpler for non-specialists to put into practice.

Modular Components

`OpenMx` is written using modular programming techniques in the `R` and `C++` languages with the intent that it will be maintained and extended by members of the research community. The core programming team helps and encourages statistical and quantitative researchers to add their research projects to the larger `OpenMx` framework. We believe that nurturing a sustainable community around `OpenMx` is essential to the project's long-term prosperity (Nakakoji, Yamamoto, Nishinaka, Kishida, & Ye, 2002). In order to work on part of `OpenMx`, one does not need to understand the inner workings of all other modules; it is only necessary to understand and adhere to the interface of the thing that one wishes to customize or extend. Although `OpenMx` originally focused on SEM, the scope of `OpenMx` is often stretched in new directions, expanding its scope (e.g., state-space modeling, Hunter, 2014). The IFA module is another instance of this happening. In creation of the IFA module, economy of effort was realized by leveraging the existing model specification and data handling methods already built into `OpenMx`. The IFA module, in turn, adds a new set of interfaces that can be customized and extended. Those who practice basic research on IFA are invited to incorporate their research projects into the `OpenMx` IFA module.

Item Models

The IFA module of `OpenMx` relies on the `RPF` package for response probability functions (Pritikin, Chalmers, Weeks, Cai, & Houts, 2014). Three functions are currently available: the 4PL dichotomous response model (Loken & Rulison, 2010), graded response model (Samejima, 1969), and nominal response model (Thissen et al., 2010). Of course, the 1PL, 2PL, 3PL, partial credit model (Masters, 1982), and generalized partial credit model (Muraki, 1992) can be constructed as restrictions of the available models. All models use the logistic metric and a multidimensional parameterization.

The community is invited to expand the list of available models. All that is required to add a new response probability function is to contribute computer code to implement the `RPF` interface (see Table 1). Beyond this modular interface, no additional computer code is required to add a new type of item model. From the point of view of the rest of the software suite, a new item model is indistinguishable from any of the 3 original item models. Item estimation and fit statistics would work seamlessly with a new item model just as they do with the item models that are currently implemented.

Item Parameter Estimation

Item parameters are optimized using MML (Bock & Aitkin, 1981) with an equal interval quadrature. For a convenient performance boost, automatic detection of two-tier covariance structure is implemented (Cai, 2010a). Bayesian priors of any functional form can be placed on any parameter. As per usual SEM practice, a free parameter can be fixed to a constant or equated with other free parameters. Precision of estimates (e.g., standard errors) can be assessed by the direct method (Oakes, 1999), covariance of the row-wise gradients, Richardson extrapolation (Jamshidian & Jennrich, 2000), Supplemented EM (Meng & Rubin, 1991; Tian, Cai, Thissen, & Xin, 2013), sandwich-type covariance (Louis, 1982; Yuan, Cheng, & Patton, 2013), likelihood-based confidence intervals, and bootstrap. Accuracy of parameter recovery and standard errors is indistinguishable from accuracy reported by Cai, Yang, and Hansen (2011). However, users are encouraged to verify our accuracy claims for themselves. A number of ready-to-run simulation studies are included with `OpenMx`, including both simulation studies from Cai, Yang, and Hansen (2011).

Optimizers

IFA is closely related to factor analysis (Kamata & Bauer, 2008; Takane & De Leeuw, 1987). However, IFA makes the simplifying assumption of conditional independence (Equation 1) to achieve higher performance. The Hessian of the completed data model is block diagonal (Cai, 2010b). `OpenMx` takes full advantage with a block-wise matrix inversion routine and Newton-Raphson optimizer. To demonstrate and exercise this code, a test in the `OpenMx` test suite routinely fits a model with 1536

2PL items. This test completes in 17 s on a developer laptop, less than the time it takes to invert a 3072×3072 matrix once on the same hardware. MML offers remarkable efficiency but at the cost of flexibility. To compute likelihood-based confidence intervals, a general non-linear optimizer is required.

`OpenMx` currently offers two non-linear optimizers: NPSOL (Gill, Murray, Saunders, & Wright, 1986) and CSOLNP (Zahery, Gillespie, & Neale, 2014). Since optimizers use a modular interface, a non-linear optimizer can be substituted for the Newton-Raphson optimizer in the M-step of MML or can optimize an IFA model directly without MML. The non-linear optimizers are much slower than Newton-Raphson but can optimize a much broader class of problems. The ability to switch optimizers can also simplify debugging. For example, if Newton-Raphson cannot solve a problem then it is a simple matter to check whether NPSOL can solve it. NPSOL tries to optimize the fit function without the use of analytic derivatives. If NPSOL cannot solve it either then, most likely, there is a mistake in the problem specification.

Test Scores

Examinee scores can be obtained with expected a posteriori (EAP; Bock & Mislevy, 1982), maximum a posteriori (MAP), and sum-score EAP (Thissen, Pommerich, Billeaud, & Williams, 1995) methods. However, the user interface is different than what many traditional IFA packages offer (e.g. menu options or buttons). EAP family scores do not involve optimization whereas MAP scores do. Therefore, EAP family scores are available from the `RPF` package, while MAP scores involve running an `OpenMx` optimization function. This seemingly minor difference is of great philosophical importance to the `OpenMx` design team. `OpenMx` is intended to accurately reflect the underlying mathematical process. We believe that closely following the math is the best way to avoid designing ourselves into a corner. Our users expect their data analysis scripts to continue working year after year, even as we add features and evolve the software. If we design ourselves into a corner then we will be forced to break backward compatibility and users will be forced to update their scripts. This means that some of the `OpenMx` IFA interfaces are not as streamlined as most other IFA software in popular use. One could regard `OpenMx` as a middle layer to support a point-and-click style IFA user interface for analysts who are satisfied with fitting an occasional single factor 2PL model. On the other hand, we believe sophisticated users will appreciate the power and consistency of the `OpenMx` approach when building intricate or non-standard models.

Fit Diagnostics

`OpenMx` offers popular model fit statistics for IFA models including CFI, TLI, RMSEA, AIC, and BIC, consistent with treatment of structural equation models. In addition, the `RPF` package includes IFA specific diagnostic tests. These include a test of the assumption of conditional independence (W.-H. Chen & Thissen, 1997), a sum-score-based test of item fit (Kang & T. T. Chen, 2008; Orlando & Thissen, 2000), and a full-information multinomial fit test (Bartholomew & Tzamourani, 1999). These tests are optimized for two-tier covariance structure. Also available are Rasch residual-based fit statistics (Wright & Masters, 1982). Consistent with our theme of modularity, each diagnostic procedure is an independent `R` function. Easy access to single tests make it straightforward to verify the uniformity of the null distribution or compute a bootstrap *p*-value.

Example Scripts

Before we present the scripts, a brief review of IFA models will be helpful. The conditional likelihood of response x_{ij} to item j from person i with item parameters ξ_j and latent ability θ_i is

$$L(x_i|\xi,\theta_i) = \prod_j \Pr(\text{pick} = x_{ij}|\xi_j,\theta_i). \tag{1}$$

One implication of Equation 1 is that items are assumed to be conditionally independent given the latent ability θ. That is, the outcome of one item does not have any influence on another item after controlling for ξ and θ_i. The unconditional likelihood is obtained by integrating over the latent distribution θ_i,

$$L(x_i|\xi) = \int L(x_i|\xi,\theta_i)L(\theta_i)\mathrm{d}\theta_i. \tag{2}$$

With an assumption that examinees are independently and identically distributed, we can sum the individual log likelihoods,

$$\mathcal{L} = \sum_i \log L(x_i|\xi). \tag{3}$$

One important observation about Equation 3 is that there are two kinds of parameters. There are item parameters ξ and latent distribution parameters θ of the participants. For didactic purposes, we will start with a model for item parameters ξ and neglect the latent distribution, assuming that the latent distribution is standard multivariate normal. Once there is some experience with item models, we will fix item parameters and focus on estimating latent distribution parameters. Finally, an example will be given of a model that estimates both item and latent distribution parameters simultaneously.

Readers familiar with prior versions of `OpenMx` know that only one strategy was available to find the maximum likelihood estimate and obtain an estimate of the information matrix. Many more options are available for IFA models. To accommodate the additional choices, the user can attach an explicit compute plan to a model. The compute plan is flexible way to communicate to `OpenMx` which operations to perform.

Item Parameters

Suppose you regularly administer the PANAS (Watson, Clark, & Tellegen, 1988), but instead of scoring participants by adding up the item scores, you want to try IFA. Here is how you might do it. Without loss of generality, we will only consider the positive affect part of the scale.

```
library(OpenMx) # load the OpenMx and RPF packages into R
library(rpf)

spec <- list()  # spec will map item models to data columns
  # rpf.grm creates a graded response item model
spec[1:10] <- rpf.grm(outcomes = 5)
data <- read.csv('demo1.csv')
  # Coerce data to ordered factors
data <- mxFactor(data, levels=1:5)

  # make starting values matrix
startingValues <- matrix(c(1, seq(1,-1,length.out=4)),
                          ncol=length(spec), nrow=5)
  # make an mxMatrix object to hold the free item parameters
imat <- mxMatrix(name='item', values=startingValues, free=TRUE,
                 dimnames=list(names(rpf.rparam(spec[[1]])), colnames(data)))
rownames(imat)[1] <- 'posAff'
  # Labels create equality constraints
imat$labels[1,] <- 'slope'

  # Give instructions on how to optimize the model
computePlan <- mxComputeSequence(steps=list(
    mxComputeEM(expectation='expectation', predict='scores',
      mstep=mxComputeNewtonRaphson()),
    mxComputeOnce(from='fitfunction', what='information', how='meat'),

    mxComputeStandardError()))
  # Construct an mxModel object called 'panas1' containing data, expectation,
  # fit function, and compute plan.
panas1 <- mxModel(model='panas1', imat,
              mxData(observed=data, type='raw'),
              mxExpectationBA81(ItemSpec=spec),
              mxFitFunctionML(), computePlan)
panas1 <- mxRun(panas1, silent=TRUE)   # run the model
summary(panas1)                        # print a summary of the results
```

The item parameters are stored in the `item` matrix with 1 item per column (line 15). Regardless of item model, the first rows contain the factor loadings. The exact parameterization of the graded response model used (lines 6 and 13) is given in the `RPF` package (Pritikin et al., 2014). We use traditional parameter names from the item model except for the first row where we override the name with the name of our factor (line 17). An equality constraint is placed on the parameters in the first row of the item matrix by giving them a label of `slope` (line 19). This causes the estimation of a Rasch model instead of unconstrained 2PL items. The compute plan (line 22) tells `OpenMx` what we want to do with the model. The default compute plan is to optimize the parameters of a model using a non-linear optimizer. This is usually suitable for structural equation models, but for IFA models, it is much more efficient to use the Expectation-Maximization algorithm (Dempster, Laird, & Rubin, 1977). We must request this explicitly with `mxComputeEM` (line 24). The information matrix is approximated by the inverse of the covariance of the row-wise first derivatives (line 25). This estimate is called `meat` because it also forms the inside of a sandwich covariance matrix (White, 1994). The last of step the compute plan, `mxComputeStandardError` (line 26), summarizes the information matrix as standard errors.

Latent Distribution Parameters

Suppose an enterprising researcher has administered the PANAS to a large sample from the general population, fit an item model to these data, and published item parameters. You are testing an experimental intervention that should increase positive affect and wish to check whether your sample has significantly more positive affect than the general population mean.

```
35    # Here we are going to reuse the item parameters from our first
36    # example. Instead of estimating them, we are going to fix
37    # them and assume that they are the true population parameters.
38  panas1$item$free[,] <- FALSE
39
40    # set up the matrices to hold our latent free parameters
41  m.mat <- mxMatrix(name='mean', nrow=1, ncol=1, values=0, free=TRUE)
42  rownames(m.mat) <- 'posAff'
43  cov.mat <- mxMatrix(name='cov', nrow=1, ncol=1, values=diag(1), free=TRUE)
44  dimnames(cov.mat) <- list('posAff', 'posAff')
45
46    # build the item part of the model
47  data <- read.csv('demo2.csv')
48    # Coerce data to ordered factors
49  data <- mxFactor(data, levels=1:5)
50  panasModel <- mxModel(model=panas1, m.mat, cov.mat,
51                     mxData(observed=data, type='raw'), name='panas')
52
53   # build the latent distribution part of the model
54 latentModel <- mxModel(model='latent',
55   mxDataDynamic(type='cov', expectation='panas.expectation'),
56   mxExpectationNormal(covariance='panas.cov', means='panas.mean'),
57   mxFitFunctionML())
58
59 computePlan <- mxComputeSequence(steps=list(
60   mxComputeEM(expectation='panas.expectation', predict='scores',
61     mstep=mxComputeGradientDescent(fitfunction='latent.fitfunction')),

62   mxComputeConfidenceInterval()))
63
64   # build the container model
65 e1Model <- mxModel(model='experiment1', panasModel, latentModel,
66   mxFitFunctionMultigroup('panas.fitfunction'),
67   mxCI('panas.mean'),
68   computePlan)
69 e1Model <- mxRun(e1Model, silent=TRUE)  # run the model
70 summary(e1Model)                         # print a summary of the results
```

If you want to run this code on your own computer, note that this example is not self-contained but depends on the previous example. Inside the M-step (line 61), the latent distribution of an IFA model is independent of item parameters (Cai, 2010b). The discrete masses accumulated in the quadrature are converted to a multivariate normal distribution by `mxDataDynamic` (line 55). The model multivariate normal distribution is fit against these dynamic data by `mxExpectationNormal` (line 56). Since the latent model does not have any constraints, the optimizer will find a model distribution that is essentially identical to the data distribution. However, this is not generally the case. Within the M-step, we can fit an arbitrary structural equation model to the data covariance and mean vector. In fact, `mxExpectationNormal` is exactly the same function that we could use to specify a SEM. The compute plan (line 59) tells `OpenMx` what we want to do with the model. It is necessary to specify an explicit compute plan because there are more choices to make than can be accommodated by the default plan. `mxComputeGradientDescent` invokes a non-linear optimizer (line 61). Our use of `mxFitFunctionMultigroup` is not strictly necessary (line 66). This fit function is used because it generalizes to more than 1 group (as in Appendix B). We request a likelihood-based confidence interval around the mean (using `mxCI`) to obtain a more robust test of our hypothesis than we could expect from a Wald test (line 67).

Item and Latent Distribution Parameters

The reader may be aware that other IFA software is more succinct when it comes to model specification. Only a single line of `R` code is required to fit our first example with `mirt`. The benefits of the `OpenMx` approach become apparent in the specification of more intricate models. Multiple group models often involve one group anchored to a standard normal distribution with latent distribution parameters of the other groups estimated relative to the anchoring group. An example is given in Appendix B that is approximately a combination of our first two examples. The specification is lengthy, but the user has control over every detail. As few as another 10-20 lines of `R` code are required to turn this example into a Monte Carlo study to predict statistical power at a range of candidate sample sizes (e.g., Kelly et al., in press).

Discussion

`OpenMx` is an open-source statistical modeling software package with IFA and SEM modules that runs in the `R` statistical computing environment. The IFA module builds on a strength of the `OpenMx` design wherein a full programming language is used for model specification. The IFA engine (Bock & Aitkin, 1981) has been carefully tuned to perform well on multicore hardware (Levon, 2014; Valgrind Developers, 2014) and can fit a wide variety of multidimensional IFA models. Once an IFA model is fit, a selection of factor scores, fit statistics, and diagnostic tests are available.

Projects like `OpenMx` are never fully completed and the IFA module is no exception. For higher-dimensional models, the Metropolis-Hastings Robbins-Monro algorithm (Cai, 2010b) would make a useful addition since Bock and Aitkin (1981) becomes impractically slow as the number of dimensions increase (the curse of dimensionality). Once we can efficiently estimate models with more factors then it would be a straightforward extension to allow RAM (McArdle & McDonald, 1984) or LISREL (Jöreskog & Van Thillo, 1972) notation to model the mean and covariance matrix. The assumption that the latent distribution is multivariate normal could be relaxed using splines (Woods, 2006) or empirical histograms (Woods, 2007). A limited-information goodness of fit test (e.g. Cai & Hansen, 2013) would be helpful since the full-information test performs poorly as the number of items increase. Scripts to automate exploration of differential item function and test linking would ease these common tasks.

The current article only briefly covers the many features and facilities of `OpenMx`. To learn more, obtain a free download of the software, and participate in the forums please go to http://openmx.psyc.virginia.edu (Boker et al., 2011).

Conclusions

`OpenMx`, a freely available open-source statistical software package, is now capable of estimating IFA models via MML. However, datasets often include other indicators measured on an interval or ratio scale. For example, in an educational context, students are often recorded with an age and a socioeconomic score. Educators want to understand how school performance is influenced by demographic factors in a general SEM setting. There is some work in this direction (e.g., Adams, Wilson, & Wang, 1997). However, at the time of writing, there is still no consensus on how to best combine IFA and SEM approaches without resorting to Monte Carlo methods (Cai, 2013). The availability of an IFA algorithm within capable, open-source SEM software is a step toward the unification of IFA and SEM. We hope `OpenMx` will spur research into a more satisfactory reconciliation of these two methodologies.

References

Aberdour, M. (2007). Achieving quality in open-source software. *Software, IEEE*, *24*(1), 58–64.

Adams, R. J., Wilson, M., & Wang, W.-c. (1997). The multidimensional random coefficients multinomial logit model. *Applied Psychological Measurement*, *21*(1), 1–23.

Bartholomew, D. J. & Tzamourani, P. (1999). The goodness of fit of latent trait models in attitude measurement. *Sociological Methods & Research*, *27*(4), 525–546.

Bock, R. D. & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika*, *46*, 443–459.

Bock, R. D., Gibbons, R., & Muraki, E. (1988). Full-information item factor analysis. *Applied Psychological Measurement*, *12*(3), 261–280.

Bock, R. D. & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement*, *6*(4), 431–444.

Boker, S. M., Neale, M. C., Maes, H., Wilde, M., Spiegel, M., Brick, T. R., . . . Bates, T., et al. (2011). OpenMx: An open source extended structural equation modeling framework. *Psychometrika*, *76*(2), 306–317.

Cai, L. (2010a). A two-tier full-information item factor analysis model with applications. *Psychometrika*, *75*(4), 581–612. doi:10.1007/s11336-010-9178-0

Cai, L. (2010b). High-dimensional exploratory item factor analysis by a Metropolis-Hastings Robbins-Monro algorithm. *Psychometrika*, *75*, 33–57.

Cai, L. (2012). flexMIRT: A numerical engine for multilevel item factor analysis and test scoring [Computer software]. Version 1.88. Vector Psychometric Group.

Cai, L. (2013, October). *Flexible multidimensional item analysis, test scoring, and model fit evaluation.* Talk presented at the annual conference of the Society of Multivariate Experimental Psychology, St. Petersburg, Florida.

Cai, L. & Hansen, M. (2013). Limited-information goodness-of-fit testing of hierarchical item factor models. *British Journal of Mathematical and Statistical Psychology*, *66*(2), 245–276.

Cai, L., Thissen, D., & du Toit, S. (2011). IRTPRO [Software manual]. Version 2.1. Scientific Software International.

Cai, L., Yang, J. S., & Hansen, M. (2011). Generalized full-information item bifactor analysis. *Psychological Methods*, *16*(3), 221–248.

Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, *48*(6), 1–29. Retrieved from http://www.jstatsoft.org/v48/i06/

Chen, W.-H. & Thissen, D. (1997). Local dependence indexes for item pairs using Item Response Theory. *Journal of Educational and Behavioral Statistics*, *22*(3), 265–289.

Cheung, M. W.-L. (2014). *metaSEM: Meta-analysis using structural equation modeling*. R package version 0.9-0. Retrieved from http://courses.nus.edu.sg/course/psycwlm/Internet/metaSEM/

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1–38.

Gill, P. E., Murray, W., Saunders, M. A., & Wright, M. H. (1986). *User's guide for NPSOL (version 4.0): A fortran package for nonlinear programming*. DTIC Document.

Hunter, M. D. (2014, May). *Extended structural equations and state space models when data are missing at random*. Talk presented at the Annual Meeting of the Association for Psychological Science, San Francisco, CA.

Jamshidian, M. & Jennrich, R. I. (2000). Standard errors for EM estimation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *62*(2), 257–270.

Jöreskog, K. G. & Van Thillo, M. (1972). LISREL: A general computer program for estimating a linear structural equation system involving multiple indicators of unmeasured variables. [Computer software].

Kamata, A. & Bauer, D. J. (2008). A note on the relation between factor analytic and Item Response Theory models. *Structural Equation Modeling*, *15*(1), 136–153.

Kang, T. & Chen, T. T. (2008). Performance of the generalized S-X^2 item fit index for polytomous IRT models. *Journal of Educational Measurement*, *45*(4), 391–406.

Kelly, G., Mobbs, S., Pritikin, J. N., Mayston, M., Mather, M., Rosenbaum, P., . . . Forsyth, R. (in press). Gross Motor Function Measure-66 trajectories in children recovering after severe acquired brain injury. *Developmental Medicine and Child Neurology*.

Levon, J. (2014). OProfile [Computer software]. Retrieved from http://oprofile.sourceforge.net/

Loken, E. & Rulison, K. L. (2010). Estimation of a four-parameter item response theory model. *British Journal of Mathematical and Statistical Psychology*, *63*(3), 509–525.

Louis, T. A. (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 226–233.

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, *47*, 149–174.

McArdle, J. J. & McDonald, R. P. (1984). Some algebraic properties of the reticular action model for moment structures. *British Journal of Mathematical and Statistical Psychology*, *37*(2), 234–251.

Meng, X.-L. & Rubin, D. B. (1991). Using EM to obtain asymptotic variance-covariance matrices: The SEM algorithm. *Journal of the American Statistical Association*, *86*(416), 899–909.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, *16*, 159–176.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. (2002). Evolution patterns of open-source software systems and communities. In *Proceedings of the international workshop on principles of software evolution* (pp. 76–85).

Oakes, D. (1999). Direct calculation of the information matrix via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *61*(2), 479–482.

Orlando, M. & Thissen, D. (2000). Likelihood-based item-fit indices for dichotomous Item Response Theory models. *Applied Psychological Measurement*, *24*(1), 50–64.

Pritikin, J. N. (2014). *Item Factor Analysis: A primer and new open-source implementation* (Master's thesis, University of Virginia).

Pritikin, J. N., Chalmers, R. P., Weeks, J., Cai, L., & Houts, C. (2014). *Response Probability Functions* [Computer software]. Version 0.36. Retrieved August 25, 2014, from http://cran.r-project.org/web/packages/rpf/index.html

R Core Team. (2014). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. Retrieved from http://www.R-project.org

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement*, *34*(4), 100.

Takane, Y. & De Leeuw, J. (1987). On the relationship between item response theory and factor analysis of discretized variables. *Psychometrika*, *52*(3), 393–408.

Thissen, D., Cai, L., & Bock, R. D. (2010). The nominal categories item response model. In M. L. Nering & R. Ostini (Eds.), *Handbook of Polytomous Item Response Theory Models* (pp. 43–75). Routledge.

Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. (1995). Item Response Theory for scores on tests including polytomous items with ordered responses. *Applied Psychological Measurement*, *19*(1), 39–49.

Thissen, D., Steinberg, L., & Mooney, J. A. (1989). Trace lines for testlets: A use of multiple-categorical-response models. *Journal of Educational Measurement*, *26*(3), 247–260.

Tian, W., Cai, L., Thissen, D., & Xin, T. (2013). Numerical differentiation methods for computing error covariance matrices in Item Response Theory modeling: An evaluation and a new proposal. *Educational and Psychological Measurement*, *73*(3), 412–439.

Valgrind Developers. (2014). Valgrind [Computer software]. Retrieved from http://valgrind.org/

Watson, D., Clark, L. A., & Tellegen, A. (1988). Development and validation of brief measures of positive and negative affect: The PANAS scales. *Journal of Personality and Social Psychology*, *54*(6), 1063.

White, H. (1994). *Estimation, inference and specification analysis*. Cambridge University Press.

Woods, C. M. (2006). Ramsay-curve Item Response Theory (RC-IRT) to detect and correct for nonnormal latent variables. *Psychological methods*, *11*(3), 253.

Woods, C. M. (2007). Empirical histograms in Item Response Theory with ordinal data. *Educational and Psychological Measurement*, *67*(1), 73–87.

Wright, B. D. & Masters, G. N. (1982). *Rating scale analysis*. Chicago: MESA Press.

Wu, E. J. C. & Bentler, P. M. (2012). *EQSIRT - A comprehensive Item Response Theory program*. Multivariate Software, Inc. Encino, CA.

Wu, M. L., Adams, R. J., Wilson, M. R., & Haldane, S. A. (2007). *ACER ConQuest: Generalised item response modelling software*. Victorial, Australia.

Yuan, K.-H., Cheng, Y., & Patton, J. (2013). Information matrices and standard errors for MLEs of item parameters in IRT. *Psychometrika*, 1–23. doi:10.1007/s11336-013-9334-4

Zahery, M., Gillespie, N., & Neale, M. C. (2014, May). *High-performance constrained optimization in R: Applications to substance use*. Presented at the poster session of the 26th Annual Convention of Association for Psychological Science, San Francisco, CA.

Appendix A
Nominal Item Model, Testlet Construction

```
library(OpenMx) # load the OpenMx and RPF packages into R
library(rpf)
nominalTestlet <- function(data, name, factors, aRank, cRank) {
  lev <- length(levels(data[[ name ]]))  # find the number of outcomes
  spec1 <- rpf.nrm(lev, factors)   # create a nominal item model
  nthresh <- spec1$outcomes - 1

  # determine which parameters are free
  free <- rep(FALSE, spec1$factors + 2 * nthresh)
  free[1] <- TRUE
  base <- spec1$factors+1
  if (nthresh * aRank >= 1) {
    free[(1+base):(base + nthresh * aRank - 1)] <- TRUE
  }
  base <- base + nthresh
  if (nthresh * cRank >= 1) {
    free[base:(base + nthresh * cRank - 1)] <- TRUE
  }

  # suggest feasible starting values
  sv <- free * c(rep(1,spec1$factors), rep(1, nthresh), rep(0, nthresh))
  sv[ c(1,spec1$factors+1) ] <- 1

  list(spec1, free=free, startingValues=sv)   # return results
}

# create some demonstration data and items
data <- data.frame(fruitfly=mxFactor(sample.int(5, 10, replace=TRUE), 1:5),
                   lincoln=mxFactor(sample.int(6, 10, replace=TRUE), 1:6))
# What the user would do to create a nominal testlet once they had data
t1 <- nominalTestlet(data, 'fruitfly', 1, .1, .5)
t2 <- nominalTestlet(data, 'lincoln', 1, .5, .3)
```

Appendix B
A Two Group Model, Estimating Item and Latent Distribution Parameters

```
library(OpenMx) # load the OpenMx and RPF packages into R
library(rpf)
spec <- list()   # spec will map item models to data columns
  # rpf.grm creates a graded response item model
spec[1:10] <- rpf.grm(outcomes = 5)

g1Data <- read.csv('demo1.csv')
  # Coerce data to ordered factors
g1Data <- mxFactor(g1Data, levels=1:5)

  # make starting values matrix
startingValues <- matrix(c(1, seq(1,-1,length.out=4)),
                         ncol=length(spec), nrow=5)
  # make an mxMatrix object to hold the free item parameters
```

```r
imat <- mxMatrix(name='item', values=startingValues, free=TRUE,
            dimnames=list(names(rpf.rparam(spec[[1]])), colnames(g1Data)))
rownames(imat)[1] <- 'posAff'
  # label item parameters to equate them across groups
for (rx in 1:nrow(imat)) for (cx in 1:ncol(imat)) {
  imat$labels[rx,cx] <- paste('i',rx,',',cx,sep='')
}

  # The latent distribution of this group is fixed to standard normal.
anchor <- mxModel(model='anchor', imat,
                  mxData(observed=g1Data, type='raw'),
                  mxExpectationBA81(ItemSpec=spec), mxFitFunctionML())

  # set up the matrices to hold our latent free parameters
m.mat <- mxMatrix(name='mean', nrow=1, ncol=1, values=0, free=TRUE)
rownames(m.mat) <- 'posAff'
cov.mat <- mxMatrix(name='cov', nrow=1, ncol=1, values=diag(1), free=TRUE)
dimnames(cov.mat) <- list('posAff', 'posAff')

  # build the item part of the 2nd group
g2Data <- read.csv('demo2.csv')
  # Coerce data to ordered factors
g2Data <- mxFactor(g2Data, levels=1:5)
panasModel <- mxModel(model='panas2', m.mat, cov.mat, imat,
    mxData(observed=g2Data, type='raw'),
    mxExpectationBA81(ItemSpec=spec), mxFitFunctionML())

  # build the latent distribution part of the 2nd group
latentModel <- mxModel(model='latent',
    mxDataDynamic(type='cov', expectation='panas2.expectation'),
    mxExpectationNormal(covariance='panas2.cov', means='panas2.mean'),
    mxFitFunctionML())

  # build the container model
groups <- c('anchor', 'panas2')
mstepPlan <- list(
    mxComputeNewtonRaphson(freeSet=paste(groups,'item',sep='.')),
    mxComputeGradientDescent(fitfunction='latent.fitfunction',
                             freeSet=c('panas2.mean','panas2.cov')))

cModel <- mxModel(model='container', anchor, panasModel, latentModel,
157      mxFitFunctionMultigroup(c('anchor.fitfunction', 'panas2.fitfunction')),
158      mxComputeEM(expectation=paste(groups, 'expectation', sep='.'),
159        predict='scores', mstep=mxComputeSequence(steps=mstepPlan)))
160 cModel <- mxRun(cModel)            # run the model
161 summary(cModel)                    # print a summary of the results
```

This example uses the same data as the first two examples. However, instead of using only the first dataset to estimate item parameters (line 30), both datasets combined influence item parameters. In the `mstepPlan` (line 152), `freeSet` is used to partition the free parameters into item parameters and latent distribution parameters (lines 153 and 155). These two sets of parameters are optimized separately every E-M cycle.

Table 1
Interface for response probability functions.

Method
Descriptive:
Count of non-estimated parameters
Count of estimated parameters
Default upper & lower bounds (if any)
Imperative:
Plausible random parameters
Response probability function
First and second derivatives:
Estimable parameters with respect to the log likelihood
Outcome probabilities with respect to examinee ability