Using R for Item Response Theory Model Applications

Insu Paek and Ki Cole

USING R FOR ITEM RESPONSE THEORY MODEL APPLICATIONS

Item response theory (IRT) is widely used in education and psychology and is expanding its applications to other social science areas, medical research, and business as well. *Using R for Item Response Theory Model Applications* is a practical guide for students, instructors, practitioners, and applied researchers who want to learn how to properly use R IRT packages to perform IRT model calibrations with their own data.

This book provides practical line-by-line descriptions of how to use R IRT packages for various IRT models. The scope and coverage of the modeling in the book covers almost all models used in practice and in popular research, including:

- dichotomous response modeling
- polytomous response modeling
- mixed format data modeling
- concurrent multiple group modeling
- fixed item parameter calibration
- modeling with latent regression to include person-level covariate(s)
- simple structure, or between-item, multidimensional modeling
- cross-loading, or within-item, multidimensional modeling
- high-dimensional modeling
- bifactor modeling
- testlet modeling
- two-tier modeling

For beginners, this book provides a straightforward guide to learn how to use R for IRT applications. For more intermediate learners of IRT or users of R, this book will serve as a great time-saving tool for learning how to create the proper syntax, fit the various models, evaluate the models, and interpret the output using popular R IRT packages.

Insu Paek is an associate professor at Florida State University. Before he came to Florida State University, he worked as a psychometrician for large-scale assessment programs and in testing companies for several years. His research interests are educational and psychological measurement and item response modeling and its application.

Ki Cole is an assistant professor at Oklahoma State University. She teaches graduate level educational statistics courses, including item response theory and factor analysis for the behavioral sciences. Her research interests include the theory and applications of psychometrics, scale development, understanding response tendencies, and software evaluation.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

USING R FOR ITEM RESPONSE THEORY MODEL APPLICATIONS

Insu Paek and Ki Cole

Routledge
Taylor & Francis Group
LONDON AND NEW YORK

First published 2020
by Routledge
2 Park Square, Milton Park, Abingdon, Oxon OX14 4RN

and by Routledge
52 Vanderbilt Avenue, New York, NY 10017

Routledge is an imprint of the Taylor & Francis Group, an informa business

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication Data
Names: Paek, Insu (Professor of measurement and statistics), author. | Cole, Ki, author.
Title: Using R for item response theory model applications / Insu Paek and Ki Cole.
Description: Abingdon, Oxon ; New York, NY : Routledge, 2020. | Includes bibliographical references and index.
Identifiers: LCCN 2019018219 | ISBN 9781138542785 (hardback : alk. paper) | ISBN 9781138542792 (pbk. : alk. paper) | ISBN 9781351008167 (ebook : alk. paper)
Subjects: LCSH: Item response theory. | R (Computer program language)
Classification: LCC BF39.2.I84 P43 2020 | DDC 150.28/7—dc23
LC record available at https://lccn.loc.gov/2019018219

ISBN: 978-1-138-54278-5 (hbk)
ISBN: 978-1-138-54279-2 (pbk)
ISBN: 978-1-351-00816-7 (ebk)

Typeset in Bembo
by Apex CoVantage, LLC

Visit the eResources: www.routledge.com/9781138542792

CONTENTS

Preface *vii*

1 Introduction 1
1.1 Basic concepts of item response theory (IRT) 2
1.2 R IRT programs 4

2 Unidimensional IRT with dichotomous item responses 14
2.1 Rasch model application 15
2.2 2-Parameter logistic model (2PLM) application 41
2.3 1-Parameter logistic model (1PLM) application 58
2.4 3-Parameter logistic model (3PLM) application 74
2.5 4-Parameter logistic model (4PLM) application 88

3 Unidimensional IRT with polytomous item responses 102
3.1 Partial credit model (PCM) application 103
3.2 Rating scale model (RSM) application 123
3.3 Generalized partial credit model (GPCM) application 132
3.4 Graded response model (GRM) application 145
3.5 Nominal response model (NRM) application 156

4 Unidimensional IRT with other applications 164
4.1 Mixed format response IRT modeling 164
4.2 Multiple group IRT modeling 172
4.3 Fixed item parameter calibration 184
4.4 Latent regression 188

5 Multidimensional IRT for simple structure 198
5.1 Multidimensional Rasch model application 199
5.2 Multidimensional 2PLM application 203
5.3 Multidimensional 3PLM application 207
5.4 Multidimensional partial credit model (PCM) application 211
5.5 Multidimensional generalized partial credit model (GPCM) application 216
5.6 Multidimensional graded response model (GRM) application 221
5.7 IRT modeling with high dimensionality 226
5.8 IRT modeling for within-item multidimensionality 233

6 Multidimensional IRT for bifactor structure 239
6.1 Bifactor modeling for dichotomous item responses 240
6.2 Bifactor modeling for polytomous item responses 245
6.3 Testlet model application 250
6.4 Two-tier IRT modeling 257

7 Limitations and caveat 266

Index *269*

PREFACE

Item response theory (IRT) has been widely used in research and practice. To this widespread use of IRT, a major contribution was the advancement of the software programs that estimate IRT models. This progression has grown tremendously with the availability of personal computers having fast computational capacity. Most of the well-known IRT programs used in research and practice have been commercial software. In recent years, the open-source statistical computing and programming language R has become very popular. Furthermore, IRT programs, known as packages, have been introduced in the R environment. The R software and its corresponding programs are free. Because of its open-source nature, the details of the programs, including source codes and package documentations, are openly available. Many have learned R through trial and error using the provided R program documentation, but this may be a particularly time-consuming experience depending on the extent of the documentation, the programs' complexities, and possibly the learner's background. This book was written to help minimize this inefficient process and laborious experience of those beginners who want to learn how to use R for IRT analysis. And in the process, readers may learn and understand additional features of the R IRT programs that they may not have discovered on their own.

This book introduces three R IRT programs and provides detailed use of those R IRT programs with a line-by-line approach. Comments on most of the command lines in the text are provided so that readers understand what each line is for and what it produces. The R IRT programs used in this book are "eRm," "ltm," and "mirt," which cover wide varieties of modeling.

This is not a book that elaborates on the theoretical and technical aspects of IRT. For readers who desire a better understanding of the modeling and estimation of IRT, there are several IRT books available. This book is primarily focused on how to use the R IRT programs, providing detailed commands and comments that are helpful and easy to understand. This book is also a learning tool for those who are

interested in R IRT programs but have little experience with R. Students in an introductory or an intermediate IRT course and instructors who teach IRT can utilize this book in the classroom. It can also be useful to any individual wanting to learn how to conduct IRT analysis in R. When using this book, readers are strongly encouraged to sit in front of the computer, open R, and follow the commands, checking the outputs and reading the comments provided in the book. The data sets used in this book are available at the link: www.routledge.com/9781138542792

The book has seven chapters. Chapter 1 is an introduction chapter. Chapters 2, 3, and 4 are unidimensional IRT modeling. Chapters 5 and 6 cover multidimensional modeling. The last chapter is an epilogue chapter describing cautions of which readers should be aware.

To students, instructors, and practitioners interested in learning how to use R to perform IRT analyses, my co-author and I do hope this is achieved to the fullest extent while utilizing this book.

Insu Paek
Tallahassee, Florida

1

INTRODUCTION

Test theory is the "collection of mathematical concepts that formalize and clarify certain questions about constructing and using tests, and then provide methods for answering them" (McDonald, 1999, p. 9). The two primary components of the theory are the test and the persons responding to the test. A test is a set of items that are intended to relate to some unobservable mental trait. A test, also referred to as a survey, scale, or instrument in various contexts, may be used to measure achievement, knowledge, aptitude, psychological traits, or attitude. Just as a ruler is used to measure length or a protractor to measure an angle, a test is used to measure a latent trait. Items that make up a test may be dichotomous, or binary, when there are only two response options, e.g., correct or incorrect, yes or no, etc. Items with more than two response options are polytomous, e.g., a Likert style response scale, multiple-choice responses, or partial-credit scoring.

A latent trait, ability, or construct is an unobservable psychological attribute of a person. Unfortunately, there is no direct way to measure an individual's knowledge of statistics, level of happiness, or readiness for college. Instead, these are most often measured using the person's responses to a collection of test items. Based on the item responses, a score is calculated to represent the underlying latent trait, or ability. The score is theorized to be a function of the person's responses to a group of test items (Lord, 1952).

Two branches of test theory are classical test theory (CTT) and modern test theory. The unit of analysis in CTT is usually the total score, often a sum score, of the person's responses to the set of items. Item response theory (IRT) is considered one of several modern test theory methods, and, as the name implies, the unit of analysis is the individual item response. While CTT methods are easy to use and computationally efficient, CTT has several limitations including its test-level approach and scores dependent upon the test items and sample. IRT may provide

a more informative and thorough evaluation of the items and the person's latent trait. IRT alleviates several limitations of CTT at the cost of greater mathematical computations, the requirement of a large sample size, and the need for stronger assumptions. However, with the advancements of computational power and programming, people in many fields find that the benefits of using IRT outweigh the mathematical and computational complexity.

IRT estimates item characteristics through the modeled item parameters, which permit the calculation of the expected score at the item level (e.g., probability of a yes, or correct answer if responses are binary or dichotomous) and at the test level. The person's latent trait or ability score is also estimated, taking into account specific item characteristics and how the person responded to each of those items. Theoretically, IRT makes it possible to estimate item parameters independently of the specific sample that responded to the items and to estimate the person's latent ability independent of the specific set of items that were answered. This parameter invariance property in IRT is essential in many applications of IRT.

IRT modeling has been used in various contexts of testing, including test development, scaling and test score equating, and computerized adaptive testing, to name a few. IRT is an important technique used by large-scale testing companies internationally, nationally, and at the state level to build tests, evaluate items, score examinees, and develop additional test forms. It is most commonly used in educational and psychological fields, but it is becoming increasingly popular in health and business. Internationally and nationally known assessment programs and organizations who use IRT include the Program for International Student Assessment (PISA; Organisation for Economic Co-operation and Development, 2017), Educational Testing Service (ETS; Educational Testing Service, 2018), National Assessment of Educational Progress (NAEP; Allen, Donoghue, & Schoeps, 2001), Trends in International Mathematics and Science Study (TIMSS; Martin, Gregory, & Stemler, 2000), and Progress in International Reading Literacy Study (PIRLS; Martin, Mullis, & Hooper, 2016).

1.1 Basic concepts of item response theory (IRT)

IRT is a mathematical model that relates a test-taker's latent trait or ability score with the probability of responding in a specific response category of an item. Popular IRT models for dichotomously scored item responses are named according to the number of parameters that models the characteristics of an item and the form of the function. If a single characteristic of the item is modeled (e.g., item difficulty), the IRT model is called the 1-parameter model. If two characteristics of the item are modeled (e.g., item difficulty and item discrimination), the IRT model is called the 2-parameter model. When three characteristics of the item are modeled (e.g., item difficulty, item discrimination, and item pseudo-guessing, which is common for a multiple-choice item test), the IRT model is called the 3-parameter model. The mathematical function that relates a person's latent trait or ability score and the expected item score is called the item response function (IRF). Popular choices of IRFs are the logistic distribution form and the normal ogive form. The complete

naming convention of IRT models combines these two features of the number of item parameters and the IRF form, e.g., the 2-parameter logistic model, in short 2PLM, or the 2-parameter normal ogive model, in short 2PNO model.

Brief history of IRT

In 1952, Frederick Lord (1952) published *A Theory of Test Scores* with the purpose of alleviating limitations of CTT. The intent was to derive a measurement model that could be used to measure latent ability in a way that was invariant of a specific test or set of items. He utilized the normal ogive function, i.e., the 2PNO model, to model the probability of a correct, yes, or positive response (for a dichotomous response) as a function of the test-takers' latent ability and the items' parameters. This model, however, was computationally difficult because it involve the integration of the normal distribution form in the IRF. Birnbaum (1957) proposed a model using the logistic distribution, i.e., the 2PLM, which simplified the computation needs with a more mathematically tractable logistic distribution function. Though there are some discrepancies between the two IRT forms, when a scaling constant ($D = 1.702$) is used in the 2PLM, the difference of the probabilities produced by 2PNO model and 2PLM is less than 0.01 across the entire latent trait scale (see Camilli, 1994).

In Europe, a Danish mathematician, Georg Rasch (1960, 1980), independently developed a probabilistic model that achieved a similar relationship between a person's latent trait and dichotomous item responses using a logistic distribution for the IRF. Rasch's model featured a single item location parameter (i.e., item difficulty) to represent the psychometric property of an item. His model is known as the "Rasch model." The 1-parameter model (1PLM) is sometimes addressed as an alternative to the Rasch model in the IRT literature; however, it should be noted that the Rasch model is different from 1PLM in their origins, history, and philosophy on measurement. The 1PLM stands on the usual model-data fit paradigm and is treated as a submodel of the 2PLM. The Rasch model is based on a different perspective on constructing a measure. See Wright and Stone (1979) and Andrich (2004, 2010) for more details of the Rasch model and the Rasch model proponents' perspective on measurement.

In addition to the Rasch model, 2PNO model, and 2PLM, IRT modeling for dichotomous responses has progressed to more complex modeling, for example, the 3-parameter logistic model (3PLM; Birnbaum, 1968) and 4-parameter logistic model (4PLM; Barton & Lord, 1981). Models have also evolved for polytomous response data. Most notable polytomous IRT models are the partial credit model (PCM; Masters, 1982), rating scale model (RSM; Andrich, 1978), graded response model (GRM; Samejima, 1969), generalized partial credit model (GPCM; Muraki, 1992), and nominal response model (NRM; Bock, 1972).

The recognition of the complexity of underlying traits, or abilities, in educational and psychological testing and the limitations of a single trait, or unidimensional modeling, have led to extensions to account for the multidimensionality of response data. That is, the aforementioned unidimensional dichotomous and polytomous models have been extended to multidimensional models. Furthermore, the fast growth of

the computational efficiency in personal computers and the availability of several multidimensional IRT programs facilitate the applications of multidimensional IRT models, which have become more common than ever before. See Reckase (1997, 2009) for the detailed historical background of multidimensional IRT.

Assumptions of IRT

For a successful application of IRT modeling, three assumptions must hold. First, dimensionality. To apply an IRT model, the number of dimensions of the test data must be clearly defined. Dimensionality is the number of constructs, or latent traits, being measured by a test and required to adequately respond to those test items by the test-taker. Reckase (1990) defines it as the "number of dimensions needed to summarize a data matrix that results from interactions of a set of test items and a group of examinees." Most of the IRT applications are confirmatory, which require prior specification of the number of dimensions and relationships between items and dimensions. One should be clear regarding whether a unidimensional or multidimensional IRT model is appropriate. The use of a unidimensional IRT model does not necessarily dictate that a single construct is sufficient to describe the data from a test.

Second, local independence. Local independence is the conditional independence of item responses on a person's latent traits. Local independence in popular unidimensional IRT model applications assumes that the specification of a single dimension is sufficient to explain all the associations among item responses. If multidimensionality holds, say, two dimensions, then, local independence in popular multidimensional modeling implies that two dimensions are needed to explain all the associations among item responses. For a test measuring a single construct, i.e., that is unidimensional, local independence is achieved if no further association among item responses exists after controlling for the single latent trait or factor. In the same vein, for a 2-dimensional data set, local independence is achieved when no further associations among item responses exist after controlling for the two dimensions, or latent traits or factors.

Third, proper choice of the item response function. When data follow the 3-parameter IRT model, the choice of the 3-parameter model is most appropriate rather than using the 1-parameter model because the 1-parameter model will not fully account for the item's three characteristics. The appropriate choice of the IRF in IRT applications reduces bias in the estimation of the item and the person parameters and subsequent applications. Specifying the proper IRF (i.e., number of parameters and IRF form) may subsume other implicit assumptions such as monotonicity in latent trait, or ability. Note that monotonicity in latent trait modeling is not always used in some IRT modeling, such as in the unfolding model (Roberts & Laughlin, 1996; Roberts, Donoghue, & Laughlin, 2000).

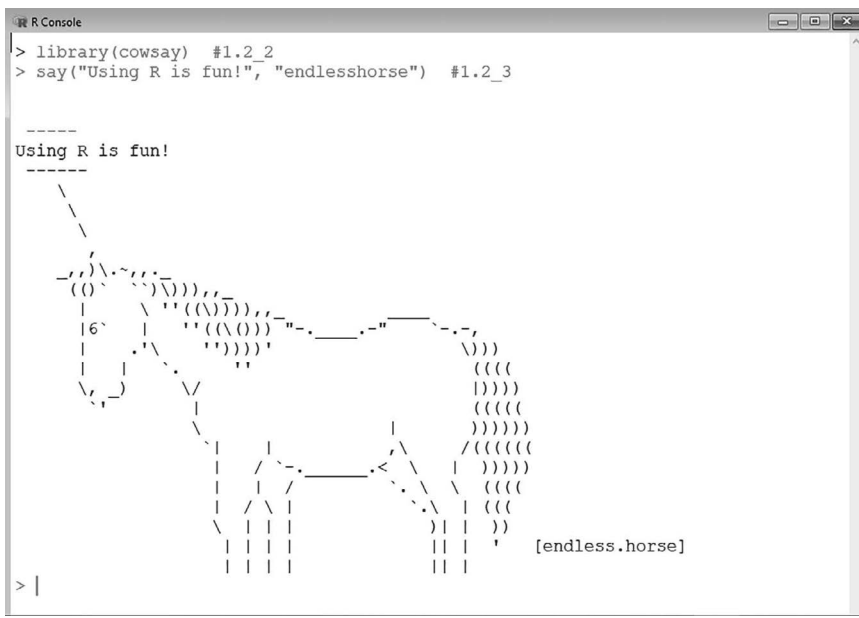
1.2 R IRT programs

There are several software applications for conducting IRT analysis. Popular software include IRTPRO (Cai, Thissen, & du Toit, 2011), flexMIRT (Cai, 2013), and WINSTEPS (Linacre, 2011), to name a few. Distinguishing factors of the software

include the models supported (e.g., dichotomous or polytomous), dimensionality, calibration methods, sample size and test length limitations, availability of specification of priors for parameters, interfacing, output, and cost. One of the primary hindrances of all commercially offered software is cost. The cost for a single user license could be more than a few hundred dollars. One of the benefits of the R software (R Core Team, 2017) is that it is an open-source software, freely available worldwide.

A brief introduction to the R software

The R "environment," as it is called, is both a software and a language. It provides a wide variety of statistical computing, data manipulation, and graph-producing features. Rather than a single software compiled with a collection of all statistical functionalities, it offers various packages that provide specific functional capabilities. For example, there are packages for specific types of analyses: item response theory, structural equation modeling, collections of psychometric methods, factor analysis, and other non-psychometric related analysis, including regression, generalized linear modeling, multilevel modeling, time series analysis, spatial analysis, etc. There is even a "fun" package for playing classic games like Minesweeper and Gomoku (a two-player game similar to traditional Connect Four), or for generating random passwords. One package, "fortunes," shares humorous quotes from the R-help mailing list. Another just-for-fun package is "cowsay," which generates an image of one of 41 animals or characters using character strings with a user-specified quotation. The figure below provides an example.



R is an open source software. This means that anyone can contribute and write a package that can then be made available to the public. There are extreme benefits

and cautions to this. As a benefit, there is a very diverse library into which experts and professionals have contributed useful and valuable resources that are freely available without charge. However, users must be cautious of packages that are performing in beta versions, that are not maintained and updated, or that may contain errors. The packages discussed in this text are well-written, have active maintainers, and are continually updated and improved. Still, there are some features that may not perform as expected or have not been properly evaluated and for which further investigations and updates should take place.

There is an open source program, "RStudio," that provides an even more user-friendly interface for using R. (The installation of RStudio requires R.) In this book, the use of the R console version is assumed, although one may use the RStudio to implement all the commands provided in the text without any problems.

An overview of R IRT packages used in the book

To date, there are more than 20 packages that perform some type of IRT analysis. A complete list can be found at the cran.r-project.org website. In this book, three packages are used: "eRm," "ltm," and "mirt." These three are relatively well-known in the measurement research community. They may be distinguished primarily by their foci on Rasch family models, non-Rasch IRT models, and multidimensional IRT models, respectively.

The "eRm: Extended Rasch Modeling" package (Mair, Hatzinger, & Maier, 2018) is used to fit dichotomous and polytomous Rasch family models. It was first released in 2006, and the most recent version, 0.16–1, was released on May 26, 2018. It uses the conditional maximum likelihood (CML) method of estimation.

The "ltm: Latent Trait Models under IRT" package, developed by Rizopoulos (2018), is used to fit the 1PL, 2PL, and 3PL dichotomous models and the GRM and GPCM polytomous IRT models. The package was first released in 2005. It is currently in version 1.1–1, released on April 17, 2018. For model estimations, it uses the marginal maximum likelihood (MML) method of estimation.

The third and final package presented in this text is "mirt: Multidimensional Item Response Theory," developed by Chalmers (2018). It was first released in 2011, and the most recent version, 1.28, was published on May 20, 2018. "mirt" is probably the most flexible R IRT program. "mirt" supports unidimensional and multidimensional modeling with dichotomous and polytomous item response data for popular IRT models. It also allows user-specified constrained versions of most of the models supported in the package, multiple group analysis, extended mixed-effects models, and several other IRT models such as cognitive diagnostic models. Also, "mirt" provides extensive options for the estimation methods for the model parameters and their standard errors, some of which are explored in the multidimensional chapters. The default model estimation method in "mirt" is the MML method. Readers interested in the performance of "mirt" are referred to Xu and Paek (2016) and Lin and Paek (2016), which provide some information

on the relative performance evaluation of "mirt" to the commercial software flexMIRT.

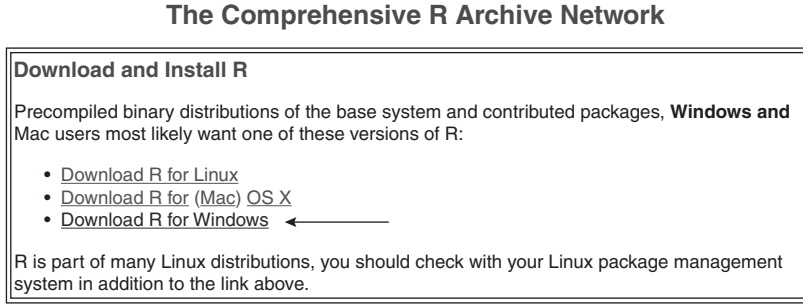
All the analyses in the book were conducted under the R version 3.5.0. The versions of the IRT packages, "eRm," "ltm," and "mirt," were 0.16–1, 1.1–1, and 1.28, respectively.

R and three-IRT package installations

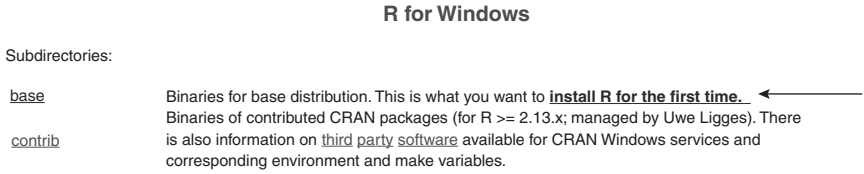
For the R installation,

1 Go to https://cran.cnr.berkeley.edu.
2 Choose "Download R" for Windows or Mac. For Windows, click "Download R for Windows."
3 Click "install R for the first time."
4 Click "Download R 3.5.3 for Windows." The version of the R changes periodically. Readers may see a higher version of R when a new version is available. Some may need to select "Save File" if a window prompt asks "Would you like to save this file?"
5 When the download is complete, click the "R-3.5.3-win.exe" to begin the installation.
6 Follow the on-screen prompts. (Accept the defaults unless you have other particular reasons.)

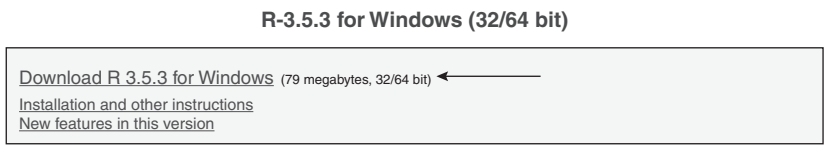
Step 2.



Step 3.

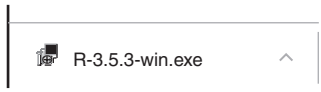


Step 4.

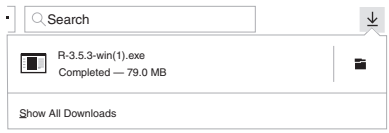


Step 5.

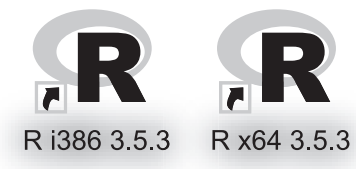
From Chrome:



From Firefox:



Once R is successfully installed, two desktop icons ("R i386 3.5.3" and "Rx64 3.5.3") are created. Double-clicking either of the icons will open the R program. "Rx64 3.5.3" would be the preferred one for most of the users. See the figure below for the R program icons.



Upon opening the R software, an additional "R Console" window is visible. There are two ways to interact with R – through the console or through a script editor.[1] On the console, when a ">" prompt is visible, then R is ready to accept a new command. When a "+" prompt is visible, R is still waiting for more command(s) in order to execute. For example, next to the ">" prompt, users may type "1 + 1" and then "Enter." R supplies the answer to the arithmetic on the next line, followed by another ">" prompt. However, if users type "1 +" and then "Enter," the command is incomplete and the "+" prompt is next to the cursor. This means that R is waiting for more information. Now, typing "1 + 1" and "Enter" completes the command, and R performs the full command of "1 + 1 + 1." The figure below displays the R window and the described arithmetic.

R may also perform with the use of functions. There are several built-in functions in R. For example, the "`sqrt()`" function calculates the square root of the value supplied within the function as an argument, i.e., function(argument). To calculate the square root of 4 type "`sqrt(4)`" and "Enter."

The IRT analysis in this textbook requires three IRT packages. To install each package, use the function "`install.packages()`" and supply the name of the package in quotation marks as the argument. Follow the commands to install each of the three IRT packages.

- Type: `install.packages("eRm")`
  - For "select a CRAN mirror," choose one of the USA ones (e.g., USA (CA1) [https]).
  - You should see a message about a successful download for the package.
- Type: `install.packages("ltm")`
  - Follow the same procedure as earlier.
  - If you are installing these three packages in the same R session, you will not be prompted for "select a CRAN mirror."
- Type: `install.packages("mirt")`
  - Follow the same procedure as earlier.

Both the R program and the packages are periodically updated; thus, users should check if new versions are available because a new version of packages tends to be upgraded with regard to fixing bugs or having new features. Sometimes, to use the full feature of the newest package(s), it may require the newest version of the R version as well.

Once a package is installed, it does not need to be installed when a new R session is opened on the same computer (unless for installing a newer version). An "R session" is a unique launch of the R software. However, each time a new R session is opened, the package must be loaded using the "`library()`" function, supplying the name of the package (without quotation marks) as the argument. For example, to load the "eRm" package after installation, type "`library(eRm)`."

As a start, let us begin with producing the horse picture shown in Figure 1.2_1. Throughout the textbook, R commands (or R scripts) are provided in rectangular boxes, as shown, with a numbering system that defines the (1) chapter of the text, (2) book section within the chapter, and (3) the command line. The first command, "`install.packages("cowsay")`," is numbered "#1.2_1" for the first chapter, second section, first command. We highly encourage readers to open R and follow along with the instructions line-by-line. Each line within the box is fully described and indexed by the line number. For some lines, the output is also provided, but not for all. Once readers execute a line, they should verify the output and understand the interpretations as guided within the text. Typing the line number, e.g., "`#1.2_1`," is not necessary when executing the line.

```
install.packages("cowsay") #1.2_1
library(cowsay) #1.2_2
say("Using R is fun!", "endlesshorse") #1.2_3
?say #1.2_4
q() #1.2_5
```

#1.2_1. This line is to install the "`cowsay`" package using the "`install.packages()`" function. R utilizes functions with arguments for most commands. In this line, "`install.packages ()`" is the function, and "`cowsay`" is supplied as an argument in double quotation marks. Note, not all arguments will be supplied using double quotation marks. This is dependent upon the type of argument defined within the function. After typing the line, press "Enter." Then follow the procedure described earlier for installing a package and selecting the appropriate CRAN mirror.

#1.2_2. Upload the "`cowsay`" package using the "`library()`" function by typing line #1.2_2. Notice in this line that "`cowsay`" is typed inside the function without the double quotation marks. After typing the line, press "Enter" to execute it.

#1.2_3. Use the "`say()`" function to draw the horse seen in Figure 1.2_1. The first argument – "Using R is fun!" – indicates the message to be printed. The second argument, "endlesshorse" in this example, indicates what figure, or animal, is created. After typing the command, press "Enter."

The text for the horse figure may be changed if a different word or phrase is typed as the first argument, e.g., "Nice horse!" instead of "Using R is fun!" The type of figure can be changed by typing a different animal as the second argument, e.g., "cat" or "monkey."

#1.2_4. When a "?" is typed along with the function name, the function documentation is opened in a web browser. Typing "`?say`" and pressing "Enter" opens the description of the "`say()`" function and its options. (See the list of animals available for the figure under the "`by`" arguments.) Whenever one wants to get information about any function, typing the question mark and a package name together (and pressing "Enter") provides information on the package. An equivalent command is "`help`." Typing "`help(say)`" and pressing "Enter" yields the same effect.

#1.2_5. This last line command is for finishing the current R session. Typing "`q()`" and pressing "Enter" will show an window asking "Save workspace image?" Clicking "No" closes the R session. A session can also be closed by clicking the "X" in the top right corner of the window (for a Windows machine).

Note

1 Rather than working directly in the console window, users can also create a script that is a compilation of the commands for a specific project. Creating a script (or a syntax, or a command) file is useful because users can save or edit the script for later uses. To use the R script editor, click "File" and "New script" in R. An additional window titled "Untitled – R Editor" will appear. Users can save the script with a new title (e.g., "R-practice.R") on a designated hard drive. To execute a single line of code, select and highlight the line, or ensure the cursor is on the line, and press "Ctrl+R" at the same time or click the icon on the menu for the "run line or selection" located next to the "save" icon. To open the existing R script file, click "File" and "Open script."

References

Allen, N. L., Donoghue, J. R., & Schoeps, T. L. (2001). *The NAEP 1998 Technical Report, NCES 2001–509*. Washington, DC: Office of Educational Research and Improvement, National Center for Education Statistics.

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, *43*(4), 561–573.

Andrich, D. (2004). Controversy and the Rash model: A characteristic of incompatible paradigm? *Medical Care*, *42*, I-7–1–16.

Andrich, D. (2010). Understanding the response structure and process in the polytomous Rasch model. In M. Nering & R. Ostini (Eds.), *Handbook of polytomous item response theory models* (pp. 123–152). New York, NY: Routledge.

Barton, M. A., & Lord, F. M. (1981). An upper asymptote for the three-parameter logistic item-response model. In *Research bulletin* (pp. 81–20). Princeton, NJ: Educational Testing Service.

Birnbaum, A. (1957). *Efficient design and use of tests of a mental ability for various decision-making problems* (Series Report No. 58–16). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord & M. R. Novick (Eds.), *Statistical theories of mental test scores* (pp. 397–479). Reading, MA: Addison-Wesley.

Bock, R. D. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, *37*, 29–51.

Cai, L. (2013). *flexMIRT: A numerical engine for flexible multilevel multidimensional item analysis and test scoring (Version 2.0)* (Computer software). Chapel Hill, NC: Vector Psychometric Group.

Cai, L., Thissen, D., & du Toit, S. H. C. (2011). *IRTPRO for Windows* (Computer software). Lincolnwood, IL: Scientific Software International.

Camilli, G. (1994). Origin of the scaling constant d=1.7 in item response theory. *Journal of Educational and Behavioral Statistics*, *19*(3), 293–295.

Chalmers, P. (2018). *mirt: A multidimensional item response theory package for the R environment. 1.28*. Retrieved from https://cran.r-project.org/web/packages/mirt/mirt.pdf

Educational Testing Service. (2018). *ETS's research legacy in statistics and psychometrics*. Retrieved from www.ets.org/research/topics/statistics_psychometrics/legacy/

Lin, Z., & Paek, I. (2016, May). *A comparison of recently implemented point and standard error estimation procedures for multidimensional IRT modeling between R package 'mirt' and flexMIRT*. Presentation at the 2016 Modern Modeling Methods Conference, Storrs, CT.

Linacre, J. M. (2011). *Winsteps ® Rasch measurement computer program*. Chicago, IL: Winsteps.com.

Lord, F. (1952). *A theory of test scores* (Psychometric Monograph No. 7). Richmond, VA: Psychometric Corporation. Retrieved from www.psychometrika.org/journal/onlie/MN07.pdf

Mair, P., Hatzinger, R., & Maier, M. (2018). *eRm: Extended rasch modeling. 0.16–1*. Retrieved from https://cran.r-project.org/web/packages/eRm/eRm.pdf

Martin, M. O., Gregory, K. D., & Stemler, S. E. (Eds.) (2000). *TIMSS 1999 technical report*. Boston, MA: International Study Center.

Martin, M. O., Mullis, I., & Hooper, M. (Eds.) (2016). *Methods and procedures in PIRLS 2016*. Boston, MA: International Study Center.

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

McDonald, R. (1999). *Test theory: A unified treatment*. New York, NY: Routledge Taylor & Francis Group.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement*, *14*, 59–71.

Organisation for Economic Co-operation and Development (OECD). (2017). *Pisa 2015 technical guide*. Paris, France: OECD.

R Core Team. (2017). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from www.R-project.org/

Rasch, G. (1960, 1980). *Probabilitistic models for some intelligence and attainment tests*. Chicago, IL: University of Chicago Press.

Reckase, M. D. (1990, April). *Unidimensional data from multidimensional tests and multidimensional data from unidimensional tests*. Paper presented at the Annual Meeting of the American Educational Research Association, Boston, MA.

Reckase, M. D. (1997). The past and future of multidimensional item response theory. *Applied Psychological Measurement*, *21*(25), 25–36.

Reckase, M. D. (2009). *Multidimensional item response theory*. New York, NY: Springer.

Rizopoulos, D. (2018). *ltm: An R package for latent variable modelling and item response theory analysis. 1.1–1*. Retrieved from https://cran.r-project.org/web/packages/ltm/ltm.pdf

Roberts, J. S., Donoghue, J. R., & Laughlin, J. E. (2000). A general item response theory model for unfolding unidimensional polytomous responses. *Applied Psychological Measurement*, *24*, 3–32.

Roberts, J. S., & Laughlin, J. E. (1996). A unidimensional item response model for unfolding responses from a graded disagree-agree response scale. *Applied Psychological Measurement*, *20*, 231–255.

Samejima, F. (1969). *Estimation of latent ability using a response pattern of graded scores* (Psychometrika Monograph Supplement No. 17). Richmond, VA: Psychometric Society.

Wright, B. D., & Stone, M. H. (1979). *Best test design*. Chicago, IL: MESA Press.

Xu, J., & Paek, I. (2016). *The 3PL model recovery with different prior distributions when using flexMIRT and the mirt package in R*. Presentation at the 61st annual meeting of the Florida Educational Research Association, Lakeland, FL.

2

UNIDIMENSIONAL IRT WITH DICHOTOMOUS ITEM RESPONSES

Unidimensional IRT models are appropriate when a test measures a single construct and when one latent trait underlies a test-taker's ability to adequately respond to the set of items in a test. All the models in this chapter assume unidimensionality, and the logistic item response function (IRF) form without the scaling constant, $D = 1.7$, is used.

The Rasch model is presented first, followed by the 2-parameter logistic model (2PLM or 2PL model). Then the 1-parameter logistic model (1PLM or 1PL model) is covered. Some might wonder why the 2PLM is introduced first instead of the 1PLM. Our reasoning was that understanding the 2PLM facilitates understanding the 1PLM as a submodel of the 2PLM because it is easier to see that the 1PLM is a nested form of the 2PLM in the marginal maximum likelihood (MML) estimation, where the standard normal distribution is assumed for ability (θ). Also, unlike the Rasch model, the 1PLM is typically treated as a special case of the 2PLM. When the MML estimation is employed, the 1PLM and the Rasch model are mathematically equivalent in that the model parameter estimates of the former can be transformed to those of the latter or vice versa, and in that both models produce the same degree of the overall model-data fit (e.g., −2loglikelihood, Akaike's information criterion (AIC; Akaike, 1974), or Bayesian information criterion (BIC; Schwarz, 1978)). However, the Rasch model is different from other IRT models in origins, philosophical ground, and constructing a measurement system. (See Andrich (2004) for the Rasch model perspective on measurement.)

Following the Rasch, 2PLM, and 1PLM, the 3-parameter logistic model (3PLM or 3PL model) and the 4-parameter logistic model (4PLM or 4PL model) are explained.

Within each section, the model IRF equation is presented, and the item parameters are defined. The calibration of model is then executed using the "eRm," "ltm,"

and/or "mirt" packages in the R software (R Core Team, 2018). The following steps are instructed for execution in each package, if available:

1 Downloading and loading the package into R.
2 Reading in the appropriate data set.
3 Fitting the model to the data.
4 Checking the convergence of the calibration.
5 Printing the item parameter estimates and their standard errors.
6 Printing the person parameter estimates.
7 Observing global model fit indices, local item fit indices, and person fit indices.
8 Displaying item characteristic curves, item information curves, test characteristic curves, and test information curves.

2.1 Rasch model application

The Rasch model for dichotomously scored responses is sometimes referred to as the simple Rasch model. The simple Rasch model IRF shown in Equation 2.1_1 was defined by Georg Rasch (1960, 1980).

$$P\left(X_{ij}=1|\,\theta_j\right)=\frac{\exp\left(\theta_j-b_i\right)}{1+\exp\left(\theta_j-b_i\right)}\,, \tag{2.1_1}$$

where X_{ij} is the jth person's response to the ith item (0 = incorrect/no; and 1 = correct/yes), θ_j is the jth person's latent trait or ability score, and b_i is the ith item difficulty.

The first R package for the simple Rasch model introduced here is the *Extended Rasch Modeling* package, or "eRm" (Mair & Hatzinger, 2007a, 2007b; Mair, Hatzinger, & Maier, 2018). The estimation of the simple Rasch model in the "eRm" package utilizes the conditional maximum likelihood (CML) estimation. The CML estimation is only possible in the Rasch family models. The distinctive features of the CML estimation are (1) the ability to separate item and person parameters in the likelihood based on the sufficiency of the raw scores for those parameters; and (2) the lack of assumptions of a particular parametric form for the person ability distribution. These are contrasted from the popular MML estimation method for other IRT models, where no sufficient statistic exists and the person ability distribution is typically assumed to be a particular parametric form, such as a normal distribution. Those readers interested in more details of the estimation methods for Rasch and other IRT models are referred to Baker and Kim (2004) and Wright and Masters (1982). The calibration of the simple Rasch model by the MML estimation is also introduced using the "mirt" package (Chalmers, 2012).

Data

As mentioned before, all the data files used throughout the text are available at www.routledge.com/9781138542792. The data file for the simple Rasch model

application is "c2_rasch.dat." R has several capabilities to read in different types of data formats (e.g., fixed format and comma-delimited format.) Variables (or items when discussing item response data) are separated by a single space in the "c2_rasch.dat." The columns represent different items, and the rows represent different test-takers. Since the focus of this chapter is dichotomous IRT models, the item responses are either 0 (incorrect or no) or 1 (correct or yes). The "c2_rasch.dat" contains item responses of 500 test-takers' responses to 20 dichotomous items.

R command

All the R commands throughout the book are shown in a rectangular box. The output from the execution of commands are shown in a light grey rectangular box. A "#" sign in the command line denotes a comment in R. Users can type their own comments or notes with "#" followed by the comment. The text after the "#" on the command line is not executed.

On the R Console window, users should type the commands on the line next to the ">" prompt. Once a line has been typed, users can press the "Enter" key on a Windows machine (or ⌘ on a Mac) to execute the command. If a command is properly executed without any errors, the appropriate output will be displayed and users will see the ">" prompt to begin a new command.

If the command is incomplete, the "+" prompt will be visible on the line where the command is to be continued. If execution is not successful, R shows a warning message. For example, type the command "`hello`," and R provides the following error message.

```
> hello
Error: object 'hello' not found
```

One of the most frequent unsuccessful executions in the beginning of the R learning experience seems to be a typographical error in typing commands, e.g., not closing a parenthesis or quotation mark, using upper-case instead of lower-case text (or vice versa), not including a comma or period, etc. Thus, beginners should pay attention to typing the commands correctly. Also, note that R is case sensitive and differentiates the capital, upper-case, and the lowercase letters in writing R object names and commands.

Rasch model calibration with the "eRm" package

The instructions shown provide step-by-step guidance for using the "eRm" package to calibrate data to the unidimensional simple Rasch model. We encourage readers to begin a new R session on their own computers and follow along with the commands and verify the output obtained.

```
install.packages("eRm") #2.1_1
library(eRm) #2.1_2
setwd("c:/mystudy/RIRT") #2.1_3
```

#2.1_1. First, the "eRm" package must be installed using the "`install.packages()`" function. Type line #2.1_1. Provide the package name within the parentheses as an argument surrounded by double or single quotes; single and double quotes can be used interchangeably in R, but double quotes are often preferred (R Core Team, 2018). Then choose a CRAN mirror, for example, "USA (CA1) [https]," and click "OK." Once the package is installed on a computer, there is no need to repeat this step when using the same computer (unless the package has been updated and the user wants to install the updated version). If the installation is successful, R will state in the output, "`package 'eRm' successfully unpacked`."

#2.1_2. Once a package is successfully unpacked, it needs to be loaded using the "`library()`" function. R has base packages that are embedded in the initial installation of R and do not need to be loaded. "eRm" is not a base package. In general, R IRT packages are not part of the base package. Downloading and loading new packages with the "`library()`" function is necessary to use these packages and their features in R.

#2.1_3. Set up a working directory and folder using the "`setwd()`" function. In this example, we created a folder called "`RIRT`" under the directory of "`C:/mystudy`." The data files should be under this directory, and the R results can be saved in the specified directory. Note that, in R, the forward slash ("/") is used to specify the directory. ("//" is an alternative to "/".) This line obviates repeated specifications of the directory for reading and exporting data and R results. Readers can create their own directories as desired.

```
ddd <- read.table("c2_rasch.dat") #2.1_4
dim(ddd) #2.1_5
head(ddd) #2.1_6
names(ddd) #2.1_7
new.name <- paste("MC", 1:20, sep="") #2.1_8
new.name #2.1_9
length(new.name) #2.1_10
names(ddd) <- new.name #2.1_11
names(ddd) #2.1_12
```

#2.1_4. Now, read in the data file for the Rasch calibrations using the "`read.table()`" function. In this example, the data file is read and stored in the name called "`ddd`" in the current R session. The "`<-`" is an assignment operator, which assigns the results of the command on the right of the operator to the object name indicated to the left of the operator. Here, "`ddd`" is the name created for the data read from the "c2_rasch.dat" file. By default, this function, "`read.table()`," creates a type of R object called a data frame that is a 2-dimensional array for storing data, similar to a spreadsheet. After executing the line, the data read from "c2_rasch.dat" is stored with the name "`ddd`" as an R data frame object. Users can provide their own name for the

object, for instance, "rasch_dat" or "data_rasch." Again, lower- and upper-case letters are distinguished in the R syntax writing. For example, "rasch_dat" and "Rasch_dat" refer to two different R objects. When exiting R, the R object(s) created in an R session (e.g., "ddd" here) are deleted from R unless users choose an option for saving them.

#2.1_5. The "dim()" function provides the dimensions, or the number of rows and the number of columns, of the object supplied within the function, in this case "ddd." The results of executing the "dim()" command are provided here. The first number "500" represents the 500 rows in the "ddd" data frame, or 500 test-takers, and the second number "20" represents the 20 column variables, or items, in this case.

```
> dim(ddd)
[1] 500 20
```

#2.1_6. The "head()" function is used to show the first six lines of the R data object. The head of the "ddd" object is presented here. The first six rows are the six test-takers' response data. The columns "V1" through "V20" are the default variable names when the data file is read into R, corresponding to items 1 through 20.

```
> head(ddd)
  V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20
1  0  0  0  1  0  1  0  1  0   0   1   1   1   1   0   0   1   0   1   0
2  0  1  0  0  1  0  0  0  1   1   1   1   1   0   1   1   0   0   1   0
3  1  1  0  0  0  1  1  0  0   0   1   1   1   1   1   1   0   0   1   0
4  0  0  1  0  1  0  1  1  0   0   1   1   1   1   0   0   0   0   0   0
5  0  0  1  0  0  0  0  1  0   1   1   0   0   0   0   1   0   0   1   0
6  1  1  1  1  1  1  0  1  0   1   0   0   1   1   0   1   1   0   1   1
```

#2.1_7. The "names()" function provides the labels, or names, of the column variables in the object supplied as the argument – in this case, the data frame "ddd."

```
> names(ddd)
[1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11"
"V12" "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20"
```

#2.1_8 and #2.1_9. Next, create the "new.name" object to replace the default variable (item) names with new character values in the "new.name." The "paste()" function creates a series of characters based on the supplied arguments. In this case, we create a series of characters that starts with "MC" and appends the values 1, 2, etc., up to 20 with no separation between them.

The "new.name" is an R object, a type of data object called "list" in R. To see what its elements are, type and execute "new.name" on the command line, and the 20 new characters are printed.

```
> new.name<-paste("MC", 1:20, sep="")
> new.name
[1] "MC1" "MC2" "MC3" "MC4" "MC5" "MC6" "MC7" "MC8" "MC9"
"MC10" "MC11" "MC12" "MC13" "MC14" "MC15" "MC16" "MC17"
"MC18" "MC19" "MC20"
```

#2.1_10. The "length()" function provides the number of elements in the R data object. "new.name" was created to have 20 character values (for 20 items) because the data frame "ddd" has 20 variables, or items.

```
> length(new.name)
[1] 20
```

#2.1_11. The variable names (item labels) in the "ddd" are replaced using the "<-" assignment operator by the character values in the "new.name."

#2.1_12. To check the new "ddd" variable names, use the "names()" command.

```
> names(ddd)
[1] "MC1" "MC2" "MC3" "MC4" "MC5" "MC6" "MC7" "MC8" "MC9"
"MC10" "MC11" "MC12" "MC13" "MC14" "MC15" "MC16" "MC17"
"MC18" "MC19" "MC20"
```

Item parameter estimation

The "eRm" package uses the CML estimation. Because the CML estimation does not require the specification of the latent trait (θ) distribution, the model identification (ID) constraint is typically given to the item parameters. The most popular model ID constraint is to set the average of all item difficulties equal to 0, that is, $\Sigma b_i = 0$. Another possible constraint is to fix one of the item parameters to an arbitrary value, for example, the first item difficulty equal to 0, $b_1 = 0$. The "eRm" allows either of these model ID constraint, and follows the sum of item difficulties equal to 0 by default (meaning no additional specifications in the code is necessary).

```
mod.R<-RM(ddd) #2.1_13
mod.R$conv #2.1_14
round (cbind(-mod.R$betapar, mod.R$se.beta), 3) #2.1_15
round (cbind(-confint(mod.R)[,2], -confint(mod.R)[,1]), 3)
#2.1_16
summary(mod.R) #2.1_17
```

#2.1_13. Now, calibrate the data to the simple Rasch model calibration with the CML estimation. "`RM()`" is the key function, and the required argument is the object containing the data, e.g., "`ddd`." The calibration result is saved in the R object "`mod.R`" using the "`<-`" assignment operator. The default model identification constraint, $\Sigma b_i = 0$, is used. If one desires the first item difficulty = 0 be constraint, the command "`RM(ddd, sum0=F)`" can be used.

Typing `help(RM)` or `?RM` will provide detailed documentation of options for the "`RM`" function in the "eRm" package. The documentation includes the various arguments that can be supplied into the function, values that are returned when the function is executed, example code using the function, and other details related to the author, reference, and related functions. One should always keep in mind the use of the "`help()`" function or "`?`" when seeking more information about functions. To observe the various components of the `mod.R` object, type `attributes(mod.R)`. Then each of these attributes can be referenced by using the object name, followed by "`$`," and then the attribute name (e.g., line #2.1_14).

#2.1_14. "`mod.R$conv`" requests the convergence attribute of the object, `mod.R`. A returned value of "`1`" indicates the estimation finished within the default stopping criteria without being flagged by particular problems in the "`RM()`" estimation.

#2.1_15. The estimated item difficulties $\left(\hat{b}_i\right)$ and their corresponding standard errors are held within the `mod.R$betapar` and `mod.R$se.beta` attributes. Line #2.1_15 requests the estimated item difficulty and standard error, rounded to three decimal places. The first numeric column titled "`[,1]`" is the item difficulty estimate, $\hat{b}_i$, and the second numeric column titled "`[,2]`" is the standard error of $\hat{b}_i$. The output indicates that the first item's estimated difficulty is $\hat{b}_i = 0.475$, with a standard error of 0.099.

```
round (cbind(-mod.R$betapar, mod.R$se.beta), 3)
            [,1]  [,2]
beta MC1   0.475 0.099
beta MC2  -1.138 0.102
beta MC3  -0.344 0.096
...
beta MC19 -0.420 0.096
beta MC20  0.516 0.099
```

"`mod.R$betapar`" extracts the negative of item difficulty estimates, $-\hat{b}_i$, whereas "`-mod.R$betapar`" converts it into the usual difficulty parameter. "`mod.R$se.beta`" extracts the standard error of the $\hat{b}_i$. The standard error of $\hat{b}_i$

is the same as that of $-\hat{b}_i$. "`round()`" is a function that controls how many decimals the output should contain. It's currently specified as three; thus, the results are rounded to three decimals. Using "`round()`" is not required, but it shows more readable output. The "`cbind()`" function combines the supplied arguments, in this case "`-mod.R$betapar`" and "`mod.R$se.beta`" column-wise.

R reads a complete line of code or command much like a mathematical formula, starting from the inside of the innermost parenthesis and working out. For Line #2.1_15, R first extracts the values of "`mod.R$betapar`" and "`mod.R$se.beta`." Then the two are combined using "`cbind()`." Last, the combined values are rounded to three decimal places.

#2.1_16 The "`confint()`" function provides the 95% confidence interval (CI) for the supplied argument(s), in this case the $\hat{b}_i$ of "`mod.R`." The first numeric column is the lower bound (2.5% quantile) and the second numeric column is the upper bound (97.5% quantile). Again "`round()`" was used to provide the results rounded to the three decimals. The results of item 1 correspond to the item parameter estimate, plus and minus the z-critical value (1.96 for the 95% CI) multiplied by the standard error: $0.475 \pm (1.96 \star 0.099) = [0.281, 0.669]$.

```
> round (cbind(-confint(mod.R)[,2], -confint(mod.R)
  [,1]), 3)
             [,1]    [,2]
beta MC1    0.281   0.669
beta MC2   -1.338  -0.937
beta MC3   -0.531  -0.156
...
beta MC19 -0.608  -0.232
beta MC20  0.321   0.711
```

Typing the command `confint(mod.R)` also provides the 95% CI, but this is the 95% CI associated with the item easiness parameter estimate, $-\hat{b}_i$.

#2.1_17. An alternative way to request the item parameter estimates, their standard errors, and the 95% CI of the estimate is using the "`summary()`" function. The first part of the output, "`Item (Category) Difficulty Parameters (eta)`," is the item difficulty parameterization under the $\Sigma b_i = 0$ constraint. However, the result for the first item is not shown. The first item estimate can be obtained by the negative sum of all the other item difficulty estimates. The second part of the output, "`Item Easiness Parameters (beta)`," is the item easiness parameterization under the same $\Sigma b_i = 0$ constraint. Notice the easiness parameter is equal to the corresponding difficulty parameter obtained from "`- mod.R$betapar`."

```
> summary(mod.R)
Results of RM estimation:
Call: RM(X = ddd)
Conditional log-likelihood: -4530.381
Number of iterations: 16
Number of parameters: 19

Item (Category) Difficulty Parameters (eta): with 0.95 CI:
      Estimate Std. Error lower CI upper CI
MC2   -1.138     0.102     -1.338     -0.937
MC3   -0.344     0.096     -0.531     -0.156
...
MC19  -0.420     0.096     -0.608     -0.232
MC20   0.516     0.099      0.321      0.711

Item Easiness Parameters (beta) with 0.95 CI:
          Estimate Std. Error lower CI upper CI
beta MC1   -0.475   0.099   -0.669   -0.281
beta MC2    1.138   0.102    0.937    1.338
...
beta MC19   0.420   0.096    0.232    0.608
beta MC20  -0.516   0.099   -0.711   -0.321
```

Item characteristic curve (ICC) plot

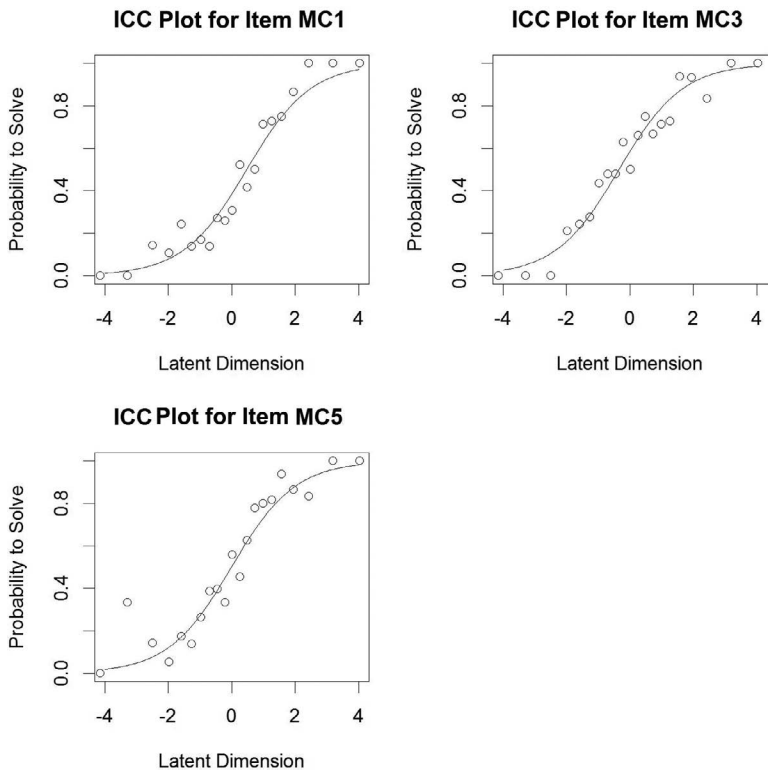
The IRF equation, e.g., Equation 2.1_1, has a corresponding item characteristic curve (ICC) for each item. The ICC is an estimated (or model-based) curve; comparing it with the empirical ICC provides a way to check the goodness of fit for an item in a graphical way. Although the empirical ICC is the term to be distinguished from the model-based ICC, it is not entirely data-based because the empirical ICC is also based on the estimated latent trait score, or $\hat{\theta}$, that is part of the model estimate. Thus, one should consider the use of this empirical vs. model-based ICC comparison plot as an approximate graphical check for the goodness of fit at the item level.

```
plotICC(mod.R, 1:4) # ICCs for items #2.1_18
plotICC(mod.R, item.subset = c(1,3,5),
empICC = list("raw")) #2.1_19
```

#2.1_18. The "`plotICC()`" function provides the ICC(s) based on the supplied arguments. This plots the model-based ICCs. The first argument is the object containing the fitted IRF object; the second argument specifies the items for which the ICC(s) are constructed. In this example, "`1:4`" requests model-based ICC plots for items 1 through 4. Other items can be specified as the second argument to the function (see #2.1_19). The model-based ICC

for each item is drawn in a separate plot. Users must click on the "R Graphics" window to display the plot.

#2.1_19. Comparison plots where the empirical ICC is overlaid onto the model-based ICC plot are reported when "`empICC = list("raw")`" is included. The circles represent the empirical ICCs; these are the proportions of 1's, i.e., correct or yes answers, directly obtained from the observed data. The "`item.subset = c(1,3,5)`" argument inside the "`plotICC()`" function requests the plot of items 1, 3, and 5. The resulting figure of this command is displayed in Figure 2.1_19. A good fit shows the empirical ICC and the model-based ICC to be close or approaching nearly identical shapes; that is, circles are near or on the model-based ICC.



Person latent trait, or ability score estimation

"eRm" provides the MLE for the estimated latent trait or ability score, $\hat{\theta}$. The MLE does not provide finite estimates for the response patterns of all 0s and all 1s, i.e., the perfect raw scores (number-correct scores). To provide estimates for these, "eRm" uses an interpolation technique called spline interpolation.

```
p.R<-person.parameter(mod.R) #2.1_20
p.R #2.1_21
round(cbind(p.R$thetapar$NAgroup1, p.R$se.
theta$Nagroup1), 3) #2.1_22
plot(p.R) #2.1_23
```

#2.1_20. "`person.parameter()`" is used to extract the estimated latent trait scores using the estimated item parameters and data from the previously fitted Rasch object, in this case "`mod.R`." The result is stored in the "`p.R`" object.

#2.1_21. The "`p.R`" object contains the estimated $\hat{\theta}$ that corresponds to each number-correct score. The first numeric column is the "`Raw Score`." The second numeric column is the ML person ability "`Estimate`." The third numeric column is the standard error, "`Std. Error`." The standard error of the zero and perfect number-correct score is not provided, although the ability ML estimate is calculated using the spline interpolation technique. Mathematically, the ML estimates for the zero, and the perfect scores are $-\infty$ and $+\infty$, though the results report "-4.13917665" and "4.03542850," respectively.

The Rasch model has a one-to-one property, where the same raw score yields the same person ability estimate, meaning all test-takers who had a raw score of "1" had an estimated trait score, $\hat{\theta} = \cdot - 3.27618884$, regardless of which item was answered correctly. The results are organized based on this property.

```
> p.R
Person Parameters:
Raw Score     Estimate Std.Error
        0 -4.13917665         NA
        1 -3.27618884  1.0485744
        2 -2.48246503  0.7755980
        3 -1.97443879  0.6609142
        . . .
       18  2.43132323  0.7592744
       19  3.19894491  1.0357701
       20  4.03542850         NA
```

#2.1_22. If the ability estimate for each person is desired, this command provides each person's ML estimate in the first column and its standard error in the second column. "`p.R$thetapar`" returns the estimated trait scores as a list, whereas "`p.R$thetapar$NAgroup1`" returns the estimated trait scores as a vector of numeric values. Because the "`cbind()`" function can only combine vectors or matrices, and not lists, the "`p.R$thetapar$NAgroup1`" is used. If one types "`coef(p.R)`," it produces each person's ML estimate including the zero and the perfect raw scores, but no standard error is included.

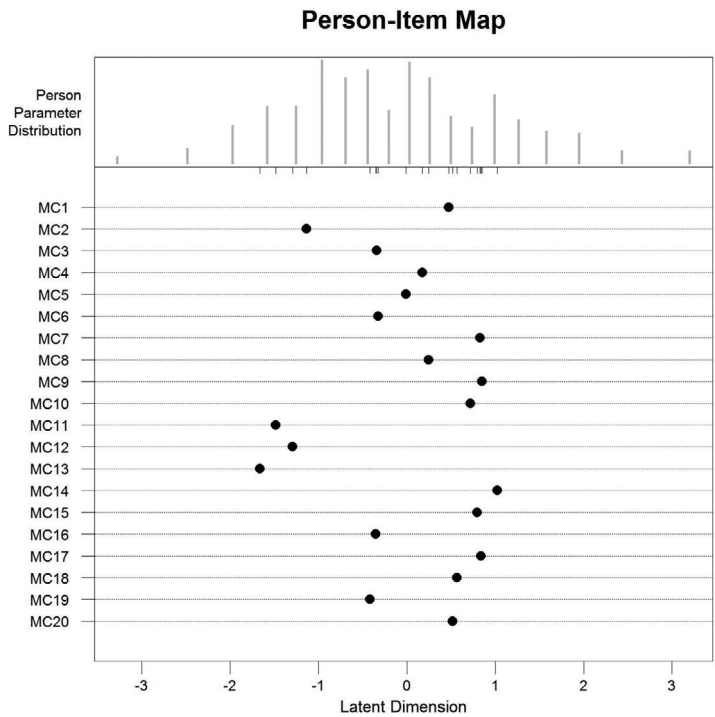
#2.1_23. "plot(p.R)" produces a plot where the x-axis is the person raw score and the y-axis the person ability estimate. The plot delineates the nonlinear relationship between raw score and ability estimate.

Person-item map

```
plotPImap(mod.R) #2.1_24
```

#2.1_24. The person-item map is created by this command, "`plotPImap()`." This is a similar map to what some Rasch model users call a "Wright Map" (Wilson, 2011). In this map, the top panel shows the histogram of person ability estimates, $\hat{\theta}$. The dots on the horizontal-axes in the lower panel are the locations of the item difficulties. In the Rasch model and other IRT models, the item difficulty parameters are located on the same scale as the person ability, θ, so that the item's difficulty can be interpreted on the same scale as the person ability scale. This map is a useful tool to understand how items and persons are spread along the θ scale. Users can also supply an additional argument to sort the items by difficulty using the function, "`plotPImap(mod.R,`

`sorted=TRUE)`." Ideally, item difficulties would be distributed across the meaningful latent trait, or ability, scale so that items are measuring ability across the entire spectrum. In this example, we may prefer more items at a higher difficulty level above 1.



Model-data fit

```
alrt<-LRtest(mod.R) #2.1_25
alrt #2.1_26
MLoef(mod.R, splitcr = "median") #2.1_27
Waldtest(mod.R) #2.1_28
plotGOF(alrt, conf=list()) #2.1_29
itemfit(p.R) # 2.1_30
plotICC(mod.R, item.subset = 1:4, empICC=list("raw"))
#2.1_31
# nonparameteric test
X<-as.matrix(ddd) #2.1_32
NPtest(X, n=500, method = "T11") #2.1_33
personfit(p.R) #2.1_34
```

Though an IRT model may be applied to a data set, and output can be produced, the quality of the model-data fit needs to be evaluated. "eRm" has several available methods for this. Following are the applications of Andersen's likelihood ratio (LR) test (Andersen, 1973), Martin-Löf's LR test (Christensen, Bjørner, Kreiner, &

Petersen, 2002; Fischer & Molenaar, 1995), and the Wald test (Fischer & Molenaar, 1995). There are several different approaches to evaluate the goodness of fit (GOF) for the overall or absolute model-data fit. One should not solely depend on one test alone. Applying several GOF measures and evaluating them together may provide a better understanding of the overall GOF between the model and the data.

#2.1_25 and #2.1_26. An overall, or global, model-data fit test called Andersen's LR test is conducted and saved as "`alrt`." Saving the result is necessary to run a graphical item misfit check by "`plotGOF()`," explained in #2.1_29. The purpose of this test is to check the subgroup model parameter invariance using the median split groups based on the raw score. If the simple Rasch model is a good description of the data, the subgroup calibration results should be nearly identical. Printing "`alrt`" provides the Andersen LR test result, which includes the LR test statistic, its degrees of freedom, and its *p*-value. The *p*-value in this application is 0.468, which is not significant under the nominal significance level $\alpha = 0.05$. Thus, the Rasch model is not rejected for this data regarding the item parameter invariance over the median-split subgroups.

#2.1_27. The Martin-Löf (MLoef) LR test, "`MLoef(),`" employs a similar approach to testing the GOF as the Andersen's LR test shown previously. However, the MLoef test splits data based on subgroups of items and conducts a test of a homogeneous item set, i.e., unidimensionality. Two item groups are made by splitting the items based on the median item raw score (default) and the homogeneity of a test over the subgroups of items is tested. The results include the MLoef test statistic, its degrees of freedom, and its *p*-value. The *p*-value in this application is 0.973, indicating that a homogeneous test described by the Rasch model is not rejected. The chi-square approximation of the MLoef test statistic tends to exhibit low power and a simulation-based approach to evaluate the *p*-value to raise power (Christensen & Kreiner, 2007) was proposed in the literature.

#2.1_28. The subject-splitting approach to testing for the GOF for the entire set of items in terms of parameter invariance can be applied at the item level using the Wald test. Again two subgroups of test-takers are formed by the median raw score (default).

```
> Waldtest(mod.R)
Wald test on item level (z-values):
          z-statistic p-value
beta MC1      -0.461   0.645
beta MC2      -1.132   0.258
beta MC3       2.029   0.042
...
beta MC18     -2.017   0.044
beta MC19      0.911   0.362
beta MC20     -0.743   0.457
```

The results of the Wald test are reported here. Notice, each item is tested in this application, with a "`z-statistic`" and "`p-value`" for each item. Without the adjustment of the family-wise Type I error rate, under $\alpha = 0.05$, for each item, item 3 and item 18 are statistically significant. With the Bonferroni correction (α/number of items), no item is statistically flagged, indicating that the item parameter invariance hypothesis for the items is not rejected.

#2.1_29. The "`plotGOF()`" function provides a graphical check of the equivalence of the two groups' (i.e., the subgroup with raw scores below the median and the subgroup with raw scores above the median) item difficulty estimates. Item difficulty estimates from the two groups are plotted with their confidence ellipses. If the simple Rasch model is a good description of the entire data, the item estimates should be similar or almost identical, and ellipses should be circular and intersect the identity line. Consistent with the results of the Wald test (see #2.1_28) without the Bonferroni correction, the confidence ellipses for items 3 and 18 do not touch the identity line, indicating that these two items are statistically flagged. (To see the ellipses clearly, enlarge the graphical window to the fullest.)

#2.1_30. The item fit indices most well-known in Rasch modeling are the infit and outfit mean-square (MSQ) and the infit and outfit *t*-statistics (Smith & Smith, 2004; Wright & Masters, 1982, 1990). Infit statistics are outlier-robust measures. The closer the infit or outfit MSQ value is to 1.0, the better the goodness of fit is. The *t*-statistics are standardized versions of the MSQ. The larger the absolute value of *t* is the more likely the item misfit is. A large absolute value departed from 0 such as |3| might be considered to flag a potentially misfitting item.

```
> itemfit(p.R)
Itemfit Statistics:
       Chisq  df  p-value  Outfit MSQ  Infit MSQ  Outfit t   Infit t
MC1  496.025 496    0.491       0.998      0.964    -0.01    -0.82
MC2  467.645 496    0.815       0.941      0.960    -0.63    -0.86
MC3  519.186 496    0.228       1.045      1.051     0.73     1.32
...
MC18 440.998 496    0.964       0.887      0.920    -1.60    -1.79
MC19 468.213 496    0.810       0.942      0.986    -0.91    -0.37
MC20 439.163 496    0.968       0.884      0.929    -1.70    -1.61
```

#2.1_31. The "`plotICC()`" function is used for a graphical check where empirical ICC is compared with model-based ICC. See also Figure 2.1_19.

#2.1_32 and #2.1_33. The "eRm" package provides a non-parametric, absolute model check procedure, "`NPtest()`." In order to utilize this function, the data "ddd" must be converted into a matrix using "`as.matrix()`" (#2.1_32). Then the "`NPtest()`" function executes the non-parametric

LR test using a resampling technique. The user must choose their statistic (in this example, "`method="T11"`") and the number of replications ("`n=500`"). "`T11`" is a global model-data check using the model-based and the empirical inter-item correlations. For more methods and options, refer to the documentation using "`?NPtest`."

```
> NPtest(X, n=500, method = "T11")
Nonparametric RM model test: T11 (global test - local
dependence)
(sum of deviations between observed and expected inter-
item correlations)
Number of sampled matrices: 500
one-sided p-value: 0.1
```

The "`one-sided p-value`" in the output is greater than the conventional al $\alpha = 0.05$, indicating that the global model-data fit through the inter-item correlation check does not reject the Rasch model.

#2.1_34. The infit and outfit measures can be applied to evaluate the person fit as well. "eRm" provides the infit and outfit MSQ and *t*-statistic measures in order to check the misfit of an item response string for each person relative to the Rasch model. Note that the previously created "`p.R`" object (see #2.1_20) that contains person abilities is necessary to run "`personfit()`."

```
> personfit(p.R)
Personfit Statistics:
      Chisq df   p-value  Outfit MSQ  Infit MSQ  Outfit t   Infit t
P1   20.427 19     0.369       1.021      1.022      0.17      0.18
P2   17.330 19     0.568       0.866      0.906     -0.64     -0.54
P3   17.742 19     0.540       0.887      0.949     -0.48     -0.28
...
P5   22.028 19     0.283       1.101      1.170      0.42      0.73
P499 19.295 19     0.438       0.965      0.974     -0.11     -0.11
P500 14.843 19     0.732       0.742      0.834     -0.70     -0.81
```

Item information curve and test information curve plots

```
plotINFO(mod.R) #2.1_35
plotINFO(mod.R, type="item") #2.1_36
```

#2.1_35. The "`plotINFO()`" function produces the item and test information curves. The top panel provides all item information curves, and the bottom panel displays the test information curve. The shapes of the Rasch model item information curves are the same since the curves differ only in their

locations (i.e., item difficulty), which provides the peak of the information, and not in their slopes.

#2.1_36. The "`plotINFO()`" function can take additional arguments specifying details of the plot(s) displayed. If only the TIC is desired, the "type = 'item'" argument can be supplied.

This completes the instruction of fitting the simple Rasch model to dichotomous data using the "eRm" package, which utilizes the CML estimation method. In the next section, the simple Rasch model will be fit to the same dichotomous data (i.e., "`ddd`") using the "mirt" package.

Rasch model application with the "mirt" package

The *Multidimensional Item Response Theory* package, called "mirt," contains functions to calibrate dichotomous and polytomous data to unidimensional and multidimensional IRT models (Chalmers, 2012). Though the name implies it is for multidimensional modeling, it also has the capabilities to fit unidimensional models, including the simple Rasch model.

Many of the steps in the previous section will be repeated in this section using the functions specific to the "mirt" package, rather than the "eRm" package. Steps include setting up the package, working directory, and data; then the model is fit, parameter coefficients and ability score estimates are extracted, plots are created, and the model and items are evaluated.

At this point, readers may be wondering which package to utilize, the "eRm" or "mirt," if both can do similar analyses. While that is true, there are fundamental differences between these and other IRT packages. Most notably at this point, the "eRm" package estimates parameters using the CML estimation procedure, whereas the "mirt" package applies the marginal maximum likelihood (MML) estimation method.

```
detach("package:eRm") #2.1_37
install.packages("mirt") #2.1_38
library(mirt) #2.1_39
setwd("c:/mystudy/RIRT") # See 2.1_3
```

#2.1_37. Before working with the "mirt" package, users should detach the "eRm" package, which unloads the "eRm" package from the current R session if you have loaded "eRm" previously, using the "`detach()`" function. This is not necessary if "eRm" was not loaded in the current R session. Detaching a package from an R session does not uninstall the package from the computer, only from the current R session. The reason for this step is that there are some functions with the same name in both the "eRm" and "mirt" packages. If both packages are loaded into the same session, R will mask the

function from one of the packages. This step insures that the correct package function is being used.

#2.1_38. If this is the first time the "mirt" package is used on a computer, it must first be installed using the "`install.packages()`" function. After executing the command, users are prompted to select a CRAN mirror.

#2.1_39. After installing, the "mirt" package should be loaded into the current R session using the "`library()`" function. If "eRm" is not unloaded using step #2.1_37, loading the "mirt" package generates a warning that "`itemfit()`" and "`personfit()`" in the "eRm" package are masked. When a newly loaded package has some overlapping functions, commands, and data sets with an existing package loaded previously for the R session, R sends this "masking" warning to tell the users which functionality and data sets are suppressed in the previously loaded package(s). This means that the "`itemfit()`" and the "`personfit()`" commands in the "eRm" package cannot be executed due to the newly loaded "mirt" package and its overlap with the "eRm." If "mirt" is the main package desired by the user for the current R session, there is no problem in using the "mirt" package, although there is a warning for the masking.

```
ddd <- read.table("c2_rasch.dat") #See 2.1_4
```

If readers have not already loaded the data into their current R session, refer back to lines #2.1_4 through #2.1_12 for instructions on loading the data, extracting characteristics of the data set, and renaming the variables.

Item parameter estimation

```
mod.rasch<-mirt(ddd, model=1, itemtype="Rasch", SE=T)
#2.1_40
mod.rasch #2.1_41
extract.mirt(mod.rasch, what="converged") #2.1_42
extract.mirt(mod.rasch, what="secondordertest") #2.1_43
coef(mod.rasch, IRTpars=T, simplify=T) #2.1_44
coef(mod.rasch, IRTpars=T) #2.1_45
round (sqrt (diag (extract.mirt(mod.rasch, what="vcov")))),
3) #2.1_46
```

#2.1_40. The "`mirt()`" function is used to fit a specified IRT model to the data and save it as an object named "`mod.rasch`." The unidimensional simple Rasch model is specified by "`model=1`" (for a uni-, or 1-dimensional model) and "`itemtype=Rasch`." "`SE=T`" requests the calculation of model parameter standard errors. The default is "`SE=F`," which instructs R to not calculate the standard errors. When standard errors are calculated, the default method is Oakes (Chalmers, 2018). Another popular computationally

efficient standard error calculation method, called "`crossprod`," can be specified by "`mirt(ddd, model=1, itemtype="Rasch", SE=T, SE.type="crossprod").`"

"`mirt()`" provides several item parameter point estimation methods and its standard error calculation methods. Readers interested in understanding more about the standard error calculation methods are referred to e.g., Cai (2008); Chalmers (2018); Paek and Cai (2014); Tian, Cai, Thissen, and Xin (2013). The Oakes method is rather new in the IRT literature. Though limited, the author's experience with the Oakes method was that the method is very efficient and performs well when compared to the "`crossprod`" method in the "mirt" package.

For more details on the arguments supplied in the "`mirt()`" function or on the models fit within the function, readers can type and run "`?mirt`" into R, and the function documentation will open in a web browser.

#2.1_41. Typing and executing the saved R object "`mod.rasch`" provides the output of the estimation convergence check results, including the second-order test result, the log-likelihood value at the final convergence, and information criteria, i.e., Akaike's information criterion (AIC; Akaike, 1974), Bayesian information criterion (BIC; Schwarz, 1978), corrected Akaike's information criterion (AICc; Sugiura, 1978), and sample size adjusted Bayesian information criterion (SABIC; Sclov, 1987). These information criteria are descriptive relative model-data fit measures to compare nested or non-nested model comparisons. When comparing the AIC between two models (or BIC, AICc, or SABIC), a smaller value corresponds to the better fitting model.

The $G2$ statistic for the absolute, overall model-data fit is also provided. However, the use of $G2$ is not recommended unless the number of items is small and the sample size is relatively large (e.g., five items or less with more than a few hundred test-takers) due to the sparsity that occurs in most of the test applications (Maydeu-Olivares & Garcia-Forero, 2010).

#2.1_42 & #2.1_43. Though the convergence check results are shown using "`mod.rasch()`," "mirt" offers another way to extract many aspects of the results. Users can obtain many kinds of results using "`extract.mirt()`," including convergence check results. These two commands provide checks for convergence and the second-order test result. When there is no problem, "`TRUE`" is printed for both checks. Other options include specifying "`what=vcov`" to obtain the variance-covariance matrix of the model parameter estimates: "`extract.mirt(mod.rasch, what="vcov")`." For a list of all options, see the function documentation using the command "`?extract.mirt`."

#2.1_44. The Rasch model item parameter estimates are printed using the "`coef()`" function. The "`IRTpars=T`" argument specifies that

the traditional difficulty IRT item parameter is reported. Otherwise, if "`IRTpars=F`," the model form of the intercept is reported. Under the simple Rasch model, the item intercept parameter is equivalent to the negative of the item difficulty. This relationship varies for other IRT models.

The "`simplify=T`" argument reports the coefficients as a matrix, with the estimated difficulty listed in the second numeric column under the head of "`b`." The "`a`," "`g`," and "`u`" parameters in the heading represent the item slope (or discrimination), the pseudo-guessing (or lower asymptote), and the slip (or upper asymptote) parameters, respectively. "mirt" uses a general IRF that subsumes the Rasch model, the 1PLM, the 2PLM, the 3PLM, and the 4PLM. The item difficulty estimates from the simple Rasch model are presented here. (Note, the item, or variable, names are "`V1`," "`V2`," etc., since the variables were not re-named when the data were read in under this section. If users re-named the variables as they were instructed in the previous section, these are "`MC1`," "`MC2`," etc.)

```
> coef(mod.rasch, IRTpars=T, simplify=T)
$'items'
      a       b   g   u
V1    1   0.607   0   1
V2    1  -1.012   0   1
V3    1  -0.210   0   1
...
V19   1  -0.287   0   1
V20   1   0.648   0   1

$means
F1
 0

$cov
      F1
F1 0.923
```

The Rasch model is obtained by fixing the slope parameter equal to 1, the pseudo-guessing parameter equal to 0, and the slip parameter equal to 1. The person ability, θ, population distribution is assumed to be a normal distribution with its mean fixed to 0 for the model identification purpose, while its variance is a free parameter in the Rasch model MML estimation. These are reported in the "`$means`" and "`$cov`" portions of the output.

The item difficulty parameter estimates from "mirt" are different from those estimated using the "eRm" package. The differences are expected because, although the same data set is used, the two packages ("eRm" and "mirt") use different model identification constraints ($\Sigma b_i = 0$ vs. the mean of the population θ distribution = 0) and different estimation methods (CML vs. MML). Using a different estimation method typically yields practically little differences when the normality assumption

for the θ distribution holds in the data. The mean of the θ normal distribution is shown as 0 because it is fixed for the model identification. Also, the estimated variance of the θ population distribution is shown as 0.923.

#2.1_45. Without the "`simplify=T`" argument, the output reports the item difficulty estimate and its 95% CI for each item. The upper and lower bounds of the 95% CI for the "a," "g," and "u" are "NA" (not available) because they are fixed. The last "`$GroupPars`" shows the population θ distribution parameter variance estimate and its 95% CI.

```
> coef(mod.rasch, IRTpars=T)
$'V1'
          a        b   g   u
par       1    0.607   0   1
CI_2.5   NA    0.392  NA  NA
CI_97.5  NA    0.823  NA  NA

$V2
          a        b   g   u
par       1   -1.012   0   1
CI_2.5   NA   -1.236  NA  NA
CI_97.5  NA   -0.788  NA  NA
...

$V19
          a        b   g   u
par       1   -0.287   0   1
CI_2.5   NA   -0.498  NA  NA
CI_97.5  NA   -0.077  NA  NA

$V20
          a        b   g   u
par       1    0.648   0   1
CI_2.5   NA    0.432  NA  NA
CI_97.5  NA    0.864  NA  NA

$GroupPars
         MEAN_1  COV_11
par           0   0.923
CI_2.5       NA   0.762
CI_97.5      NA   1.085
```

#2.1_46. There are at least two ways to extract the parameter standard errors, which are equal to the square root of the diagonal elements of the variance-covariance matrix of the item parameter estimates. One way is to use "`extract.mirt()`" function, and the other is to use "`coef()`."

When using the "`extract.mirt()`," the (error) covariance matrix is first extracted; then the square root ("`sqrt()`") of the diagonal elements of the extracted covariance matrix ("`diag()`") is calculated. The "`round()`" function is applied to show the results rounded to three decimals for easy readability. The first through the twentieth value (i.e., "`d.1`," "`d.6`," etc., up to "`d.78`") are the standard errors of the item difficulties. The last value, under the heading of "`COV_11.82`," is the standard error of the variance of the population θ distribution. Precisely speaking, the standard error given here is the standard error of $-b_i$, i.e., the standard error of the item easiness, because the general IRF employed in "mirt" is the slope and intercept form. The exponent in the IRF is $a_i\theta_j + d_i$. For the Rasch model, $a_i = 1$ and $d_i = -b_i$, which is the item easiness. Item easiness and difficulty parameters have only the sign difference, and their standard errors are the same.

The other, more straightforward, way to extract the standard errors is to use the following "`coef()`" function. This produces both parameter estimates and their standard errors.

```
coef(mod.rasch, printSE=T)
```

Person latent trait, or ability score estimation

```
fscores(mod.rasch, method="EAP", full.scores=T, full.
scores.SE = T) #2.1_47
```

#2.1_47. The person ability score estimates are obtain using the "`fscores()`" function. "mirt" provides several person ability estimators. The default is the expected a posteriori (EAP) estimator. In addition to EAP, "mirt" provides the maximum likelihood (ML), maximum a posteriori (MAP), which is the Bayes model estimator, weighted likelihood estimator (WLE; Warm, 1989), and more. If "`full.scores=T`" is omitted or "`full.scores=F`" is specified, then the person ability estimates are shown for unique observed response strings rather than for each person.

```
> fscores(mod.rasch, method="EAP", full.scores=T, full.
  scores.SE = T)
                F1       SE_F1
 [1,]  -0.06745438  0.4381910
 [2,]   0.12341358  0.4359166
 [3,]   0.31318773  0.4356889
...
[499,]  0.12341358  0.4359166
[500,]  0.89465174  0.4483522
```

The person's latent trait, or ability estimate, is under the "`F1`" column, and its standard error is under the "`SE_F1`" column (since the "`full.scores.SE=T`" argument

was provided). The "eRm" person ability estimates will be different from the person ability estimates from this "mirt" package. Using a different model identification constraint along with a different item parameter estimation method yields different item parameter estimates, though the same data set is used. Thus, the differences in the item parameter estimates are reflected in the person ability estimation because the estimated item parameter values are used to calculate the person ability estimate.

ICC, TCC, and information plots

```
itemplot(mod.rasch,1) #2.1_48
plot(mod.rasch, type = "trace") #2.1_49
plot(mod.rasch) #2.1_50
plot(mod.rasch, type="info") #2.1_51
itemplot(mod.rasch, item=1, type="info") #2.1_52
```

#2.1_48. The IRF curve, or ICC, is drawn using the "`itemplot()`" function, with the requested item supplied as the second argument, e.g., "`itemplot(mod.rasch,1)`" for item 1. If a different item ICC is desired, for example, item 3, then "itemplot(mod.rasch, 3)" is used.

#2.1_49. The "`plot()`" function can be used to produce various plots of the resulting Rasch model object. "`plot(mod.rasch, type="trace")`" produces the ICCs for all items.

#2.1_50. Calling the function, "`plot(mod.rasch)`," without any additional arguments, produces the test response function, or test characteristic curve (TCC). This plots the estimated ability score $\hat{\theta}$ against the expected total sum score.

#2.1_51. Specifying "`type="traceinfo"`" in the "`plot()`" function produces a test information curve.

#2.1_52. An individual item test information curve is drawn using "`itemplot()`," where "`item=1`" specifies the item requested, and "`type="info"`" specifies that the item information is to be plotted. Readers interested in more features of the "`itemplot()`" can obtain more detailed information by typing "`help(itemplot)`" or "`?itemplot`."

Model-data fit

```
M2(mod.rasch) #2.1_53
itemfit(mod.rasch) #2.1_54
itemfit(mod.rasch, empirical.plot=3, empirical.CI=.95)
  #2.1_55
personfit(mod.rasch, method="EAP") #2.1_56
residuals(mod.rasch, type="LD") #2.1_57
residuals(mod.rasch, type="Q3") #2.1_58
```

#2.1_53. An absolute, overall model-data fit test called the M2 (Maydeu-Olivares and Joes) up to the second-order margins in the data rather than using the full information in the test, 2005, 2006) is computed. M2 is called the limited information test procedure in that it uses up to the second-order margins in the data rather than using the full information in the test. M2 is more powerful to detect the variations in the slope, or discrimination, parameters across items when comparing the Rasch or 1PLM with the 2PLM, or when simple structure or bifactor multidimensionality exists and a unidimensional model is fit to data (see Xu, Paek, & Xia, 2017).

```
> M2(mod.rasch)
           M2  df          p      RMSEA RMSEA_5    MSEA_95      SRMSR       TLI       CFI
stats 219.3782 189 0.06429293 0.01794734       0 0.02747074 0.04639034 0.9858445 0.985919
```

The *p*-value of the M2 test (0.06429) is greater than 0.05 level of significance; thus, we fail to reject the Rasch model. More traditional descriptive fit statistics are also reported, such as the RMSEA, and its 90% CI. The RMSEA is calculated using the M2 statistic, which is approximately a chi-square variate. Note that one cannot simply apply the conventional rules of thumbs (e.g., RMSEA < 0.05) to evaluate the overall model-data fit using these descriptive fit measures when an IRT model is fitted with the categorical ordinal item response data. It appears that those descriptive fit measures should be used even more conservatively, applying a much smaller cutoff, to safely state the model's goodness of fit (see, e.g., Xu et al., 2017) or for relative evaluation of different models.

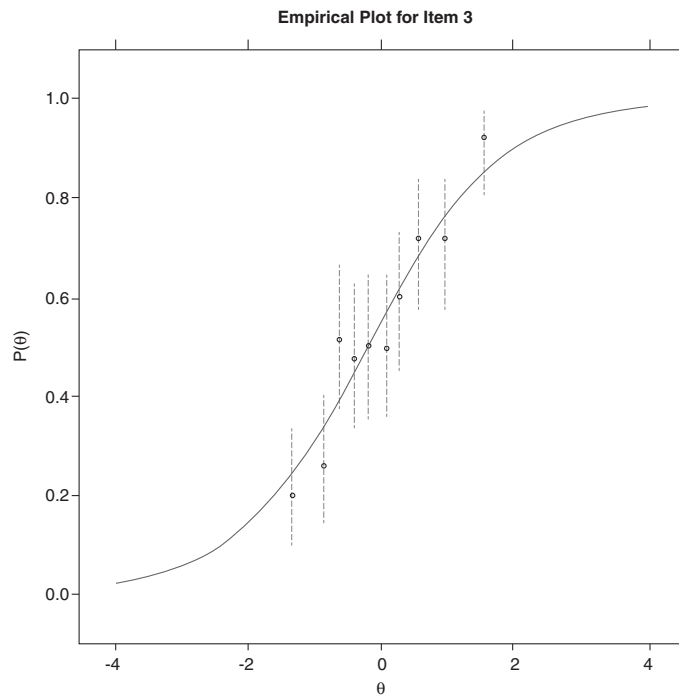
#2.1_54. "`itemfit()`" is a function to produce item level goodness of fit statistics. The item fit index, S-X2 (Orlando & Thissen, 2000, 2003), is calculated for each item, and the degrees of freedom and *p*-value are provided.

```
> itemfit(mod.rasch)
   item   S_X2  df.S_X2  p.S_X2
1    V1 15.475       14   0.346
2    V2 16.925       13   0.203
3    V3 10.413       13   0.660
...
19  V19 16.662       13   0.215
20  V20  7.625       14   0.908
```

The *p*-value of the test is shown in the last numeric column for each item. With and without the Bonferroni correction (α/number of items), there is no statistically flagged item.

"`itemfit()`" provides several item fit statistics, including the Rasch item infit and outfit measures for the unidimensional Rasch model case (recall the discussion of these measures previously in 2.1_30). Typing "`itemfit (mod.rasch, fit_stats = "infit")`" produces the infit and outfit MSQ values and the infit and outfit *t*-test statistics under "`z.outfit`" and "`z.infit`." If users compare the infit and outfit measures from the "eRm" package with those from the "mirt" package, they will notice non-ignorable differences, especially with the standardized *t*-statistic. Currently we recommend the use of infit and outfit measures from the "eRm" package if the Rasch model is being used. If "mirt" is used, we recommend the use of the S-X2 for item evaluation. It is noted that the S-X2 results of "mirt" are consistent with the in- and outfit *t*-results of the "eRm" package using the rule of flagging an item if its $|t| > 3$ or 2.

#2.1_55. "`itemfit()`" can be used to provide an item level graphical descriptive check, including "`empirical.plot=3`" to specify the item, in this case item 3. The model-based ICC (line) and the empirical ICC (circles) for an item are compared. Specifying "`empirical.CI=.95`" adds the 95% CI for the empirical ICC, shown as dotted vertical lines in Figure 2.1_55.



#2.1_56. The "mirt" package produces person fit measures using the "`personfit()`" function ("eRm" does not have this capability). The output contains the infit and outfit measures and corresponding standardized

statistics to the person response strings and the person fit index l_z, called "Zh" at the last column (Drasgow, Levine, & Williams, 1985). See #2.2_18 for further description of l_z.

```
> personfit(mod.rasch, method="EAP")
      outfit      z.outfit      infit      z.infit           Zh
1 1.0198067 0.1649132520 1.0214617 0.179092434 -0.114514871
2 0.8695400 -0.6248630776 0.9060467 -0.530822868 0.597973231
3 0.8912369 -0.4764999407 0.9436888 -0.311057699 0.414218901
...
498 1.1840840 0.9084319735 1.0939236 0.629949032 -0.681873165
499 0.9652691 -0.1118160136 0.9776767 -0.080592855 0.148047489
500 0.7104906 -0.9886655732 0.7880153 -1.244840997 1.138483760
```

For a more user-friendly output, the results of "`personfit()`" can be sorted from lowest to highest Zh statistic using the following code:

```
p.fit <- cbind(1:500, ddd, personfit(mod.rasch,
method="EAP"))
p.fit[order(p.fit[,26]),]
```

#2.1_57. LD-X2 (Chen & Thissen, 1997) is a test for local independence between item pairs. It is calculated for each item pair when "`type = "LD"`" is specified in the "`residuals()`" function. The LD-X2 is typically used as a descriptive index for the local independence violation. The result is a 20 by 20 matrix (since this data set contains 20 items). In the lower triangle, signed LD-X2 values are shown. The upper triangle elements are signed Cramer V coefficient values, which are calculated using the LD-X2 values. The Cramer V coefficient is an extension of the phi coefficient and as a signed coefficient of Cramer V, it ranges from −1 to 1. In general, a larger absolute value of these indices indicates a more severe violation of local independence for the item pair. One should be aware that data sparseness could contribute to a large absolute value of LD-X2 in addition to actual local dependence. Cai, du Toit, and Thissen (2011) treated a standardized LD-X2 value greater than 10 as a rule of thumb for local dependence worthy of more serious attention. Note that the signed LD-X2 values in the output are not standardized LD-X2 values. Assuming LD-X2 is an approximate chi-square variate with one degree of freedom (although it is not as Cai et al. (2011) pointed out, but for the following straightforward approximation), one may use the following two lines of commands and examine the lower tringle elements greater than 10 (from the second command line, "`(abs(rrr)-1)/sqrt(2)`") for possible flagging of an item pair and examining a cluster of items showing local dependence. Or equivalently one can directly examine the elements of the lower tringle in the matrix "`rrr`" and flag an item pair with LD-X2 >

15.142 for dichotomous item response data, which corresponds to the standardize LD-X2 value of 10, which in turn corresponds to |signed Cramer V| > 0.174 for dichotomous item response data. The command line of the standardization of LD-X2, in general, is "`(abs(rrr)-(k-1)*(m-1))/sqrt(2*(k-1)*(m-1))`," where "`rrr`" is a residual matrix generated by the same command line shown here in "mirt," "`k`" is the number of categories for one item and "`m`" is the number of categories for the other item. The standardization that follows is for the dichotomous item case.

```
rrr<-residuals(mod.rasch, type="LD")
(abs(rrr)-1)/sqrt(2)
```

Keeping the results in a matrix form as in "`rrr`" facilitates spotting patterns of clusters of items exhibiting local dependency. However, it may not always be easy to find item pairs quickly that show large values of |signed LD-X2| or |signed CramerV|. We provide an example of code that sorts the pairs of items from the largest value of | signed LD-X2| or |signed CramerV| to the smallest. The code is also an example of how R programming language can be utilized for users' purposes (in this case, a program of looping for the purpose of sorting in the descending order). We do not pursue the description of the details of the code here, which we leave to readers for further learning of the R programming language. Users can simply type the code shown here to obtain the results showing the item pairs listed in their signed LD-X2 or signed Cramer V in the descending order in terms of their absolute values. The code uses the "`rrr`" object created before. The output shows the signed LD-X2 or signed Cramer V in the first column (labeled "`X1`"). The second and the third column (labeled "`X2`" and "`X3`") shows the item numbers for a pair.

```
######## Sorted Signed LD-X2 ########
LDX2<-NULL
for (i in 1:ncol(rrr)){
for (j in i:nrow(rrr)){
if (i != j) {LDX2<-rbind(LDX2, cbind(abs(rrr[j,i]),
  rrr[j,i], colnames(rrr)[i], rownames(rrr)[j]))}
}
}
data.frame(LDX2[order(LDX2[,1], decreasing=T),][,-1])
######## Sorted Signed Cramer V ########
CV<-NULL
for (i in 1:nrow(rrr)){
for (j in i:ncol(rrr)){
if (i != j) {CV<-rbind(CV, cbind(abs(rrr[i,j]), rrr[i,j],
  rownames(rrr)[i], colnames(rrr)[j]))}
}
}
data.frame(CV[order(CV[,1], decreasing=T),][,-1])
```

In the result of the standardized LD-X2 from "`(abs(rrr)-1)/sqrt(2)`," no element is greater than 10. Also, no value of |signed LD-X2| is greater than 15.142. In terms of |signed Cramer V|, no value is greater than 0.174.

#2.1_58. Another descriptive index of the local independence violation is the Q3 statistic by Yen (1984). It is the correlation of the residuals from a pair of items, which are based on the difference between the observed response and the model-predicted response. Q3 is calculated for each item pair using the "`residuals()`" function and specifying "`type="Q3"`." The output is a 20 by 20 symmetric matrix (since this data set has 20 items). Because Q3 is a correlation, it ranges from −1 to 1. A large absolute value indicates a more serious degree of local dependence. When the number of items is greater than or equal to 17, the absolute value of Q3 greater than 0.2 was suggested as a rule of thumb for flagging the item pair (Yen, 1993). Also, de Ayala (2009) addresses another rule of thumb, which is $|Q3| \geq \sqrt{.05} = .2236$ in the sense of a minimum 5% of shared residual variation between two items. However, the behavior of Q3 is influenced by the number of items and the sample size; there is no single uniformly accepted cutoff value of Q3. A resampling technique such as a parametric bootstrap appears to be a better approach to decide the cutoff of Q3 for a given data set under analysis (de Ayala, 2009). The same code supplied to report the largest LD-X2 statistics can be used for the Q3 statistic (by replacing "LDX2" by "Q3" in the code). The output reports no pairs of items having a Q3 value > |0.20|.

2.2 2-Parameter logistic model (2PLM) application

The 2PLM has two features to describe an item's psychometric characteristics – the item difficulty and the item discrimination. The 2PLM IRF is given by Birnbaum (1957) and provided in Equation 2.2_1.

$$P\left(X_{ij}=1\middle|\theta_j\right)=\frac{\exp\left[a_i\left(\theta_j-b_i\right)\right]}{1+\exp\left[a_i\left(\theta_j-b_i\right)\right]}, \tag{2.2_1}$$

where X_{ij} is the jth person's response to the ith item (0 = incorrect or no, and 1 = correct or yes), θ_j is the jth person's latent trait, or ability score, b_i is the ith item difficulty, and a_i is the ith item discrimination. The "eRm" package uses the CML estimation to estimate the simple Rasch model. The CML estimation is only possible for the Rasch model. The most popular estimation method for the 2PLM and other IRT models is the MML estimation, which is the default estimation method used in the "ltm" and "mirt" packages for R IRT modeling. In the 2PLM application, we will use "ltm" and "mirt," both of which assume a normal distribution for the θ population distribution in their MML estimation.

The data file for the 2PLM application is "c2_2pl.dat." This file has a different data format than the previous data set used for the Rasch analysis. The "c2_2pl.dat" has a fixed data file format, where columns 1 through 10 are for person identification and columns 11 through 30 represent item responses to items 1 through 20. Responses are dichotomous, either 0 (incorrect or no) or 1 (correct or yet). The number of test-takers is 1000, and the number of items is 20.

When some basic commands are repeated for the 2PLM application, previous comment numbers (e.g., #2.1_1) are given unless more explanation is necessary.

2PLM calibration with the "ltm" package

```
install.packages("ltm") # see 2.1_1
library(ltm) #see 2.1_2
setwd("c:/mystudy/RIRT") #see 2.1_3
```

The R IRT package used in this application is the *Latent Trait Models for IRT*, or "ltm" (Rizopoulos, 2006). As in the previous simple Rasch model case, the package should be downloaded using the "`install.packages()`" function and loaded using the "`library()`" command. The "`setwd()`" function is used again to set up the working directory. For details, see 2.1_1 through 2.1_3.

```
ddd<-read.fwf("c2_2pl.dat", widths=c(10,rep(1,20)))
  #2.2_1
dim(ddd) #2.2_2
ddd <- ddd[,-1] # 2.2_3
names(ddd) <- paste("It",1:20, sep="") #2.2_4
```

#2.2_1. The data file has a fixed width data format where variables are located at the designated columns. To read the fixed type data file, use the "`read.fwf()`" function. The "`widths=`" argument is used to specify the width of each variable. The first ten columns are for the person identification index. The next 20 items, i.e., columns 11 through 30, represent items. Each item response takes one column. For this, the "`widths=c(10, 1, 1, ... ,1)`" may be used where the number 1 is repeated 20 times. This tells R that the first ten characters of each line in the file are for a single variable, and the next 20 characters of the line each represent a unique variable. A simplified specification is done using the "`rep()`" command, which is for repetition: "`widths=c(10, rep(1,20))`." The file is read and stored as an R object (data frame) called "ddd."

#2.2_2. The dimensions of the data are checked using "`dim()`," and the output displays "`1000`" by "`21`." By default, the variable names are "V1" through "V21." The first variable, or column, is the person identification index.

#2.2_3. Because the person index variable is not necessary in the IRT model estimation, the first column, or variable, is removed from "`ddd`." This is

done by overwriting the "ddd" variable with the original "ddd" matrix, excluding the first column, "[,-1]." The user may type "head(ddd)" or "names(ddd)" after this command to verify that only the variables "V2" through "V21" are shown.

#2.2_4. Using the "paste()" command, the variable names of "V2" through "V21" in "ddd" are replaced by the new names of "It1" through "It20."

Item parameter estimation

Item parameter estimation in the "ltm" package is done using the MML estimation. In the MML estimation, the person latent trait, or ability, parameter, θ, is typically assumed to follow a normal distribution, and its mean and variance are fixed as 0 and 1, respectively, to resolve the model identification or the latent trait scale indeterminacy issues.

```
mod.2pl <- ltm(ddd~z1, IRT.param=T) #2. 2_5
mod.2pl$conv #2. 2_6
```

#2.2_5. The "ltm()" function is used to fit response data to a specific IRT model. Inside the "ltm()" command, the first argument contains the data object – "~" – and the specified model. In this example, the R data object "ddd" is specified, and "~z1" imposes the use of a unidimensional model. "IRT.param = T" is specified to produce the item difficulty, $\hat{b}_i$. If "IRT.param = F" is used, the resulting item parameters from the slope and intercept form are produced. The calibration result is saved in the R object "mod.2pl."

Notice the exponent of the 2PLM in Equation 2.2_1 is $a_i(\theta_j - b_i)$; the a_i and b_i are the item discrimination and difficulty parameters, respectively. In the slope and intercept form, the exponent is of the form $a_i\theta + b_i$. While the discrimination or slope parameters stay the same in both forms, the item difficulty parameter, b_i, is equivalent to $-a_i b_i$ from the slope and intercept form. In the IRT literature, the discrimination parameter (a_i) is sometimes called the slope parameter due to their equivalence in either form. Technically a_i is not the slope of the IRF, but it is linearly proportional to the slope. These two terminologies, slope and discrimination, are used interchangeably in research and practice.

#2.2_6. The convergence check is supplied in "mod.2pl$conv." A similar check was done using the "eRm" package (see 2.1_14). In the "eRm" application, "TRUE" indicates complete convergence; in the "ltm" package, the value "0" indicates normal termination. If the value of one is printed, users should carefully go through possible reasons why the convergence was not achieved (e.g., checking data coding, missing values, or sufficient number of

estimation iterations for the particular data set under analysis or checking the model complexity and data size, etc.).

```
summary(mod.2pl) #2.2_7
coef(mod.2pl) #2.2_8
```

#2.2_7. The "`summary()`" function provides the log-likelihood value at the final convergence, AIC and BIC, item parameter estimates and corresponding standard errors. AIC and BIC are descriptive model-data fit indices. They are useful for comparing different models, where a smaller value indicates a better model-data fit. In this application of a single model calibration, they do not have usage.

The "`Coefficients:`" in the output lists the item difficulty estimates and their standard errors for all the items first ("`Dffclt.It1`" through "`Dffclt.It20`"). Right after the last item's difficulty estimate and its standard error, item discrimination estimates and their standard errors are listed for all the items ("`Dscrmn.It1`" through "`Dscrmn.It20`").

```
> summary(mod.2pl)
Call:
ltm(formula = ddd ~ z1, IRT.param = T)

Model Summary:
 log.Lik       AIC       BIC
-10853.7    21787.39   21983.7

Coefficients:
               value  std.err     z.vals
Dffclt.It1   -0.4560   0.0641    -7.1119
Dffclt.It2   -0.7904   0.0604   -13.0862
...
Dffclt.It19  -1.2712   0.0905   -14.0447
Dffclt.It20  -1.0685   0.1018   -10.4922
Dscrmn.It1    1.5075   0.1274    11.8320
Dscrmn.It2    2.2296   0.1935    11.5196
...
Dscrmn.It19   1.6730   0.1569    10.6615
Dscrmn.It20   1.1574   0.1116    10.3683

Integration:
method: Gauss-Hermite
quadrature points: 21

Optimization:
Convergence: 0
max(|grad|): 0.021
quasi-Newton: BFGS
```

#2.2_8. The "`coef()`" command provides a more succinct summary of the item parameter estimates. Each row represents an item. The first numeric column is the item difficulty estimate $(\hat{b}_i)$, and the second numeric column is the item discrimination, or slope parameter, estimate $(\hat{a}_i)$. As mentioned in the previous section, users may also use the "round()" function for a cleaner display: "`round(coef(mod.2pl),3)`."

```
> coef(mod.2pl)
           Dffclt     Dscrmn
It1   -0.45597511  1.5075340
It2   -0.79038680  2.2296120
...
It19  -1.27117265  1.6730142
It20  -1.06848021  1.1574098
```

Person latent trait, or ability score estimation

```
factor.scores(mod.2pl, method="EAP") #2.2_9
```

#2. 2_9. The "`factor.scores()`" function produces latent trait, or ability, score estimates. In this command, the EAP estimator was requested. EAP can be viewed as a natural choice for person ability estimation in the MML estimation, where the person θ is assumed to follow a particular distribution, such as a normal distribution, because EAP takes advantage of the assumed population θ distribution for the individual person ability. The first column in the output is an index for unique response patterns; it does not represent each person's identification index. In the output, the last value of 924 in the first column means that there are 924 unique observed response patterns, or strings, in the data out of theoretically possible 2^{20} = 1,048,576 number of response patterns.

```
> factor.scores(mod.2pl, method="EAP")
Call:
ltm(formula = ddd ~ z1, IRT.param = T)
Scoring Method: Expected A Posteriori
Factor-Scores for observed response patterns:
 It1 It2 It3 It4 ... It18 It19 It20 Obs   Exp     z1 se.z1
1   0   0   0   0 ...    0    0    0   2 3.256 -2.445 0.544
2   0   0   0   0 ...    1    0    0   2 1.118 -2.263 0.506
3   0   0   0   0 ...    1    1    0   1 0.294 -1.891 0.445
...
923 1   1   1   1 ...    1    1    0   1 0.331  1.613 0.567
924 1   1   1   1 ...    1    1    1  10 9.271  2.025 0.628
```

The "`Obs`" and "`Exp`" columns are observed and model-predicted frequencies, respectively. For example, two test-takers had a response pattern with all zeros

in this data. The "`z1`" and "`se.z1`" columns are the person ability estimate and the standard error. If the option "`resp.patterns = ddd`" is specified inside the "`factor.scores()`" command (shown here), then all person observed response patterns and their ability estimates with the standard errors are displayed in the same order as they are observed in the original data.

```
factor.scores(mod.2pl, method="EAP", resp.patterns = ddd)
```

In the 2PLM (or more complex models such as 3PLM or 4PLM), person θ estimation is sometimes called pattern scoring. Unlike the one-to-one correspondence between observed raw score and $\hat{\theta}$ in the Rasch model, the same raw scores could lead to different $\hat{\theta}$ values if their response patterns are not the same. For example, in the output of #2.2_9, line 2 corresponds to the response pattern with all zeros and a "`1`" for only item 18; line 7 corresponds to the response pattern with all zeros and a "`1`" for only item 13. Though these test-takers would have the same total sum score, the person answering only item 18 correctly had an estimated score of $\hat{\theta} = -2.263$. The person answering only item 13 correctly had an estimated score of $\hat{\theta} = -2.244$.

ICC plot

```
plot(mod.2pl, legend=T) #2.2_10
plot(mod.2pl, item=1:2, legend=T) #2.2_11
plot(mod.2pl, item=c(1,3,5), legend=T) #2.2_12
```

#2.2_10. The "`plot()`" function is used to display the ICCs for one or more items. Without any specification, all items are drawn in one plot. The logical argument "`legend=T`" displays the legend; if not specified, items are identified on the curves themselves.

#2.2_11. The second argument, "`item=1:2`," is used to specify which item(s) to plot in the "`plot()`" function. Here, ICCs for items 1 and 2 are shown in one plot.

#2.2_12. The "`item=c(1,3,5)`" argument in the "`plot()`" function specifies ICCs for items 1, 3, and 5 drawn on a single plot.

Item and test information plots

```
plot(mod.2pl, type="IIC", item=3, legend=T) #2.2_13
plot(mod.2pl, type="IIC", item=c(1,3,5), legend=T) #2.2_14
plot(mod.2pl, type="IIC", item=0, legend=T) #2.2_15
```

#2.2_13. The same "`plot()`" function used for ICCs can be used for item and test information curves by specifying the desired type. With "`type="IIC"`,"

the item information curve is requested. Still, the item (item 3 in this example) can be specified.

#2.2_14. This is equivalent to #2.2_12 with the addition of "`type="IIC"`" to produce item information curves for items 1, 3, and 5 drawn in one plot.

#2.2_15. By specifying "`item=0`" in the "`plot()`" function, the test information curve is provided.

Item fit

```
item.fit(mod.2pl) #2.2_16
item.fit(mod.2pl, FUN = mean) #2.2_17
```

#2.2_16. The fit statistic produced in the "`item.fit()`" function compares observed with fitted proportions of correct items across ten θ intervals and summarizes the differences. The test is Yen's Q_1 statistic (1981). The observed and fitted proportions are calculated at the median point in each θ interval.

```
> item.fit(mod.2pl)

Item-Fit Statistics and P-values

Call:
ltm(formula = ddd ~ z1, IRT.param = T)

Alternative: Items do not fit the model
Ability Categories: 10

          X^2 Pr(>X^2)
It1  22.1568   0.0046
It2  15.1375   0.0565
It3  11.6439   0.1678
...
It18 10.0816   0.2593
It19 16.7680   0.0326
It20 11.7102   0.1646
```

With the Bonferroni correction for the family-wise Type I error rate of 0.05, two items (items 6, and 15) are statistically flagged, indicating potential misfit of those items to the 2PLM. Without the Bonferroni correction, 13 items are statistically flagged.

In addition to the issue of setting up the number of ability intervals, the interval construction is based on the estimated abilities, not the true ability that is unobserved. Therefore, the test statistic and its *p*-value based on a chi-square sampling distribution should be interpreted approximately. Also, the behavior of the test is negatively affected if the number of items in a test is small. It appears more than 40 items provide a better behavior of this item fit test in terms of Type I error rates (Orlando & Thissen, 2000).

#2.2_17. If one wants to use the mean point instead of the median within each θ interval to calculate the observed and fitted proportions in each interval, the "`FUN=mean`" option can be specified inside the "`item.fit()`" function.

With the "`FUN = mean`" option, items 6 and 15 are statistically flagged again under the Bonferroni correction for the family-wise Type I error rate of 0.05. "`item.fit()`" also provides options to control the number of θ intervals and to implement a simulation-based item fit calculation that takes a much longer computation time. Use "`?item.fit`" for more information.

Person fit

```
person.fit(mod.2pl) #2.2_18
```

#2.2_18. Person fit measures of l_0 (Levine & Rubin, 1979) and l_z (Drasgow et al., 1985) are provided by the "`person.fit()`" function. The person fit measures are used to detect aberrant, or unusual, response patterns relative to the fitted model, which is the 2PLM here. Both l_0 and l_z are based on the person response pattern's likelihood. l_z is a standardized version of l_0. The output includes the unique response patterns observed in the data, l_0, l_z, and the p-value of l_z. The last column is the p-value or the probability of observing the l_z or lesser value of the l_z. The small p-value (e.g., less than 0.01) may be used to flag potential aberrant response patterns relative to the 2PLM.

```
> person.fit(mod.2pl)

Person-Fit Statistics and P-values

Call:
ltm(formula = ddd ~ z1, IRT.param = T)

Alternative: Inconsistent response pattern under the
estimated model
  It1 It2  It3 ... It18 It19 It20      L0      Lz Pr(<Lz)
1   0   0    0 ...    0    0    0 -2.3921  1.3715  0.9149
2   0   0    0 ...    1    0    0 -3.7714  1.1085  0.8662
3   0   0    0 ...    1    1    0 -5.6272  1.0110   0.844
...
920 1   1    1 ...    1    1    1 -9.6233 -0.1432  0.4431
921 1   1    1 ...    1    1    1 -3.9674  0.9335  0.8247
922 1   1    1 ...    0    1    1 -4.4717  0.6034  0.7269
923 1   1    1 ...    1    1    0 -6.2754  0.0401   0.516
924 1   1    1 ...    1    1    1 -2.3457  1.2317   0.891
```

For the approximate normality of the l_z, the true value of θ is required, but the l_z statistic is calculated using the estimated ability $\hat{\theta}$. Snijders (2001) proposed the l_z^* index, which behaves better when using estimated abilities, $\hat{\theta}s$, but it is not executed in the "ltm" package.

If the person fit result for each person's response pattern is desired, "`resp.patterns = ddd`" should be specified inside the "`person.fit`":

```
person.fit(mod.2pl, resp.patterns=ddd)
```

Model-data fit and model comparison

A submodel of the 2PLM that may be of interest to users is a model where all the item discrimination, or slope parameters, are equal. This model, with the equality constraint of the slopes, is called the 1PLM, which is described in a later section. A well-known statistical test procedure for the comparison of the two nested model is the likelihood ratio test. For this, one needs to fit a reduced model and a full model. The reduced model is the one that can be found by constraining some parameters of the full model. When testing one against the other, the null hypothesis is that the reduced model, e.g., the 1PLM, underlies the data while the alternative hypothesis is that the full model, e.g., 2PLM, underlies the data. a better fitting model.

```
mod.1pl<-rasch(ddd) #2.2_19
anova(mod.1pl, mod.2pl) #2.2_20
```

#2.2_19. The "`rasch()`" command in the "ltm" package is used to fit the 1PLM to the data. (Again the details of the 1PLM and its application are shown later.) The 1PLM calibration is saved under the name "`mod.1pl`."

#2.2_20. The likelihood ratio test is conducted using the "`anova()`" function. The reduced model name is specified first, followed by the full model name.

```
> anova(mod.1pl, mod.2pl) #2.2_20
Likelihood Ratio Table
             AIC      BIC    log.Lik    LRT df  p. value
mod.1pl 22128.04 22231.1 -11043.02
mod.2pl 21787.39 21983.7 -10853.70 378.65 19 <0.001
```

The likelihood ratio statistic is 378.65 and its degree of freedom is 19, which is equal to the difference in the number of model parameters in the two fitted models. The 1PLM has 20 item difficulties and one slope parameter, for a total of 21 parameters. The 2PLM has 20 item difficulties and 20 slope/discrimination parameters, for a total of 40. The difference, $40 - 21 = 19$, is the degree of freedom. The p-value

of the likelihood ratio test is less than 0.001; thus, rejecting the null hypothesis of the 1PLM in favor of the 2PLM.

The output also shows two information criteria, AIC and BIC. The values of the information criteria for the 2PLM are smaller than those for the 1PLM, again supporting the 2PLM for this data set.

2PLM calibration with the "mirt" package

The "ltm" and the "mirt" packages can both be used to fit the 2PLM to dichotomous data. Following are instructions for the use of the "mirt" package.

```
install.packages("mirt") #2.2_21
library("mirt") #2.2_22
setwd("c:/mystudy/RIRT") #2.2_23
```

#2.2_21. If the "mirt" package has already been installed on the current computer, users do not need to execute the "`install.packages()`" command.

#2.2_22. The "mirt" package does need to be loaded for each new R session. If "ltm" was loaded prior to loading "mirt" in the same R session, R sends the following warning.

```
> library(mirt)
Loading required package: stats4
Loading required package: lattice
Attaching package: 'mirt'
The following object is masked from 'package:ltm':
 Science
```

The warning indicates that the data set called "Science" in the "ltm" package is no longer usable (i.e., it is "masked") because "mirt" contains the same data set "Science," and the features of the most recent package installed overwrites any other package features with the same name. If you type "`Science`" after loading the "mirt" package, the data set from the "mirt" package is called. One can remove the "ltm" package within the current session by typing "`detach("package:ltm")`." This "masking" of the data set by uploading "mirt" is not a problem in this application.

#2.2_23. The working directory where data sets are stored should be specified in each new R session. If this "`setwd()`" function was executed previously within the same session, there is no need to specify this again.

```
ddd<-read.fwf("c2_2pl.dat", widths=c(10,rep(1,20))) #2.2_24
dim(ddd) #2.2_25
ddd <- ddd[,-1] # 2.2_26
names(ddd) <- paste("It",1:20, sep="") #2.2_27
```

#2.2_24. The same data set, "c2_2pl.dat," used for the 2PLM with "ltm" is read for the "mirt" 2PLM application.

#2.2_25, #2.2_26, and #2.2_27. See #2.2_3, #2.2_4, and #2.2_5 for details on setting up the "c2_2pl.dat" data set.

Item parameter estimation

```
mod.2pl<-mirt(ddd, model=1, itemtype="2PL", SE=T) #2.2_28
mod.2pl #2.2_29
extract.mirt(mod.2pl, what="converged") #2.2_30
extract.mirt(mod.2pl, what="secondordertest") #2.2_31
coef(mod.2pl, IRTpars=T, simplify=T) #2.2_32
coef(mod.2pl, IRTpars=T) #2.2_33
coef(mod.2pl, IRTpars=T, printSE=T) #2.2_34
round (sqrt (diag (extract.mirt(mod.2pl, what="vcov")) ),
3) #2.2_35
```

#2.2_28. The 2PLM is calibrated to data using the "`mirt()`" function, and specifying "`itemtype =2PL`. "`SE=T`" is specified to obtain the item parameter standard errors. If the argument is not included, or if "`SE=F`" is specified, the standard errors are not calculated, by default. As mentioned in #2.1_40, "`mirt()`" provides several standard error calculation methods. The default standard error calculation method is Oakes.

#2.2_29. Typing the saved "`mod.2pl`" provides the convergence check results. See also #2.1_41.

```
> mod.2pl
Call:
mirt(data = ddd, model = 1, itemtype = "2PL", SE = T)

Full-information item factor analysis with 1 factor(s).
Converged within 1e-04 tolerance after 40 EM iterations.
mirt version: 1.28
M-step optimizer: BFGS
EM acceleration: Ramsay
Number of rectangular quadrature: 61
Latent density type: Gaussian

Information matrix estimated with method: Oakes
Condition number of information matrix = 35.46826
Second-order test: model is a possible local maximum

Log-likelihood = -10853.61
Estimated parameters: 40
AIC = 21787.23; AICc = 21790.65
BIC = 21983.54; SABIC = 21856.5
G2 (1048535) = 8175.32, p = 1
RMSEA = 0, CFI = NaN, TLI = NaN
```

#2.2_30 and #2.2_31. The "`extract.mirt()`" is used to perform separate extractions of the convergence check results. When the output is "`TRUE`," it indicates that the estimation went smoothly within the specified (default) stopping criteria and the possible local maxima for the final estimates. See also #2.1_42 and #2.1_43.

#2.2_32. Item coefficients are extracted using the "`coef()`" function. The slope (or discrimination) parameter (`a`) and difficulty parameter (`b`) estimates are printed when "`IRTpars=T`," along with "`g`" and "`u`," which are the pseudo-guessing parameter (or lower asymptote) and slip parameter (or upper asymptote). The "`mirt()`" function uses a general 4-parameter IRF, and the 2PLM is estimated by fixing "`g`" to 0 and "`u`" to 1. "`IRTpars=T`" is necessary to obtain the conventional slope or discrimination and the difficulty estimates. Otherwise, the results from the slope and intercept parameterization (i.e., $a_i\theta_j + d_i$) are printed, which contains the slope (a_i) and the intercept ($d_i = -a_i b_i$ or $b_i = -d_i/a_i$). The parameters of the ability θ population distribution assumed to be normal are fixed to be 0 and 1 for the mean and the variance, which are shown under "`$means`" and "`$cov`."

```
> coef(mod.2pl, IRTpars=T, simplify=T)
$'items'
           a       b  g  u
It1    1.507  -0.453  0  1
It2    2.231  -0.787  0  1
...
It19   1.676  -1.267  0  1
It20   1.158  -1.065  0  1

$means
F1
0

$cov
 F1
F1 1
```

#2.2_33. Using the "`coef()`" function without the argument "`simplify=T`" prints the discrimination and the difficulty parameter estimates with their 95% CIs. Again, for the 2PLM, "`g`" and "`u`" are fixed to 0 and 1, respectively; thus, their CIs are not available.

```
> coef(mod.2pl, IRTpars=T)
$'It1'
              a       b   g   u
par       1.507  -0.453   0   1
CI_2.5    1.258  -0.579  NA  NA
CI_97.5   1.757  -0.328  NA  NA
...
```

```
$It20
              a       b   g   u
par       1.158  -1.065   0   1
CI_2.5    0.939  -1.265  NA  NA
CI_97.5   1.377  -0.866  NA  NA

$GroupPars
        MEAN_1 COV_11
par          0      1
CI_2.5      NA     NA
CI_97.5     NA     NA
```

#2.2_34. Lastly, the "`coef()`" function is used with the argument "`printSE=T`" to print the item parameter estimates and their standard errors. Since "`IRTpars=T`" is specified, the traditional a_i (discrimination) and b_i (difficulty) estimates are printed. Internally, the "`mirt()`" function uses the slope and intercept parameterization, $a_i\theta + d_i$. The point estimate of b_i is computed by a transformation from the intercept parameterization, i.e., $b_i = -d_i/a_i$. For the standard error of the b_i estimate, the delta method approximation is used.

```
> coef(mod.2pl, IRTpars=T, printSE=T)
$'It1'
          a       b    g    u
par   1.507  -0.453    0    1
SE    0.127   0.064   NA   NA
...
$It2
          a       b    g    u
par   2.231  -0.787    0    1
SE    0.193   0.060   NA   NA
...
$It20
          a       b    g    u
par   1.158  -1.065    0    1
SE    0.112   0.102   NA   NA

$GroupPars
        MEAN_1 COV_11
par          0      1
SE          NA     NA
```

#2.2_35. The standard errors of the slope parameter (a_i) and the intercept parameter ($d_i = -a_i b_i$) estimates are extracted directly from the estimated error covariance matrix. A similar command was provided for the Rasch model in #2.1_46. Readers may reference #2.1_46 for details.

The first and the second values of the output ("a1.1" and "d.2") are the standard errors of the slope and the intercept for the first item. The second and the third values ("a1.5" and "d.6") are the standard errors of the slope and the intercept for the second item. The order follows the standard error of the slope first and of the intercept parameter estimates for items. The labels (e.g., a1.1, d.2, a1.5, d.6) are for internal use in the model estimation, and no particular attention is needed for them.

```
> round (sqrt (diag (extract.mirt(mod.2pl, what="vcov")) ),
3)
 a1.1   d.2  a1.5   d.6 ...  a1.73  d.74  a1.77   d.78
0.127 0.095 0.193 0.151 ...  0.157 0.147  0.112  0.096
```

Person latent trait, or ability score estimation

```
fscores(mod.2pl, method="EAP", full.scores=T, full.
 scores.SE = T) #2.2_36
```

#2.2_36. The "fscores()" function extracts the person ability estimates (similar to #2.1_47 for the Rasch model with "mirt"). The EAP ability estimator is requested with the "method="EAP"." "mirt" offers more options for the person ability estimators such as MAP (or Bayesian modal estimator) and WLE. Unique person's scores are requested with "full.scores=T," otherwise, person scores are provided for unique response strings. "full.scores.SE = T" requests the standard errors of the latent trait scores. See #2.1_47 for more details and options.

```
> fscores(mod.2pl, method="EAP", full.scores=T, full.
 scores.SE = T)
                    F1       SE_F1
  [1,]  -0.0363535078  0.3515893
  [2,]   0.7379961115  0.4356442
  [3,]  -1.3324520798  0.3597927
       ...
 [999,]  -0.2367553395  0.3392012
[1000,]  -1.5956994589  0.3908837
```

The ability estimates are under the first column named as "F1" and their standard errors are in the last column titled "SE_F1." If "full.scores=F" is specified, the ability estimates are printed for each unique observed response string. Similar to the person ability output in the "ltm" application, the output with the "full.scores=F" has unique response strings, ability estimates, and their standard errors in the first, the second, and the third columns, respectively.

ICC, TCC, and information plots

The following plotting functions for the 2PLM are similar to those presented in the Rasch model section using the "mirt" package; see #2.1_48 through #2.1_52.

```
itemplot(mod.2pl, 3) #2.2_37
plot(mod.2pl, type = "trace") #2.2_38
plot(mod.2pl) #2.2_39
plot(mod.2pl, type="info") #2.2_40
itemplot(mod.2pl, 3, type="info") #2.2_41
```

#2.2_37. The 2PLM ICC for item 3 is printed using the "`itemplot()`" function.

#2.2_38. To request the ICCs for all items, the "`plot()`" function is used, with the "`type="trace"`" argument. Now, all ICCs do not have the same slopes since each item is allowed to have its own estimated discrimination, or slope parameter. Readers may refer back to the estimated item parameter values from #2.2_32. Those items with a lower discrimination estimate, "`a`," have flatter curves, whereas those items with a higher discrimination estimate, "`b`," have a steeper curve.

#2.2_39. Executing the "`plot()`" function without any additional argument produces the TCC.

#2.2_40. The test information curve is produced when the "`plot()`" function is used with the "`type="info"`" argument.

#2.2_41. Item information curve for item 3 is drawn in the plot. Produced by "`itemplot(mod.2pl, 3, type="info"`." This command is similar to #2.2_38, which produced the ICC, but the addition of "`type="info"`" produces the item information curve. Unlike the Rasch model case, the shapes of the item information curves for the 2PLM, or more complex models such as the 3PLM, are not the same across items having different discrimination, or slope parameter values. Item information shape is affected heavily by the item slope parameter values. The higher the slope parameter is, the more peaked the ICC is at the item difficulty location on the θ scale. For example, item 17 has the highest discrimination estimate $(\hat{a} = 2.613)$; at $\theta = \hat{b} = -1.374$, the peak of the item information curve is approximately at 1.707. Item 18 has the lowest discrimination estimate $(\hat{a} = 0.664)$; at $\theta = \hat{b} = -0.734$, the peak of the item information curve is at approximately 0.110. This maximum item information can be obtained by the following code.

```
iteminfo(extract.item(mod.2pl, 17), -1.374)
iteminfo(extract.item(mod.2pl,18), -0.734)
```

Model-data fit

```
M2(mod.2pl) #2.2_42
itemfit(mod.2pl) #2.2_43
itemfit(mod.2pl, empirical.plot=3, empirical.CI=.95) #2.2_44
personfit(mod.2pl, method="EAP") #2.2_45
residuals(mod.2pl, type="LD") #2.2_46
residuals(mod.2pl, type="Q3") #2.2_47
```

#2.2_42. The "`M2()`" function reports the absolute, overall model-data fit M2 statistic with descriptive model-data fit measures such as the RMSEA. See #2.1_53 for more details.

```
> M2(mod.2pl)
            M2  df         p     RMSEA RMSEA_5  RMSEA_95      SRMSR       TLI       CFI
stats 189.6136 170 0.1443064 0.01074661       0 0.01838579 0.02510003 0.9975107 0.9977727
```

The *p*-value of M2 is .1443, greater than $\alpha = 0.05$, indicating the fitted 2PLM is not rejected.

#2.2_43. The model-data fit is investigated at each item level using the "`itemfit()`" function. The item fit index, S-X2, is the default method for item fit calculation. The *p*-value of the S-X2 statistic is provided in the last column.

```
> itemfit(mod.2pl)
   item    S_X2 df.S_X2   p.S_X2
1   It1  17.374      14    0.237
2   It2   8.460      13    0.812
3   It3   5.753      16    0.991
...
19 It19  12.239      14    0.587
20 It20  18.290      15    0.248
```

With the Bonferroni correction for the family-wise Type I error control under $\alpha = 0.05$, there is no statistically flagged item within this data set. See #2.1_54 for more about the "`itemfit()`" function.

#2.2_44. The "`itemfit(mod.2pl, empirical.plot=3, empirical.CI=.95)`" command requests the model-based ICC for item 3, compared with the empirical ICC represented by the circles. The 95% CIs for the circles (empirical ICC) are shown as dotted vertical bars. See also #2.1_54.

#2.2_45. The statistic "`Zh`" person fit statistic is reported using the "`personfit()`" function. This is the l_z statistic developed by Drasgow, Levin, and Williams (1985). See #2.2_18.

```
> personfit(mod.2pl, method="EAP")
              Zh
1    -0.380600027
2     0.033937248
3     1.269778096
...
998  -0.383638876
999   0.428624558
1000  0.289496333
```

In the Rasch model fit using the "mirt" package, the "`personfit()`" provided infit and outfit indices specialized for the Rasch model (see #2.1_56), but only the "`Zh`" measure is produced by this function if the 2PLM is fitted to data. A large value of "`Zh`" (e.g., Zh < −3) needs attention for a possibility of important aberrant response patterns. Users may use the following code to save and sort the estimated Zh values for the 1,000 persons, including their original responses, in the example data set for easier observations.

```
p.fit <- cbind(1:1000, ddd, personfit(mod.2pl,
method="EAP"))
p.fit[order(p.fit[,22]),]
```

There is one person (person 110) that may be flagged as having a possible aberrant response, with a Zh less than −3.0, and 12 persons with −3 < Zh < −2.

#2.2_46. The index of local independence violation, LD-X2 (Chen & Thissen, 1997), is calculated using the "`residuals()`" function and "`type="LD"`." See #2.1_57 for a more detailed description of the output. The indices are provided for every pair of items in the 20 by 20 matrix (for a data set of 20 items). A large absolute value of the standardized LD-X2 is a strong piece of evidence for the violation of local independence. After standardizing the LD-X2 as shown (elements in the lower triangle from "`(abs(rrr)-1)/sqrt(2)"`"), item pairs showing greater than 10 may be flagged for their potential local dependence.

```
rrr<-residuals(mod.2pl, type="LD")
(abs(rrr)-1)/sqrt(2)
```

#2.2_47. Yen's Q3 index (Yen, 1984) is calculated when "`type="Q3"`," which is another index of the violation of the local independence. As in the LD-X2, Q3 is calculated for every item pair. The greater the absolute value of the Q3 is, the stronger the violation of local independence is. See #2.1_58.

2.3 1-Parameter logistic model (1PLM) application

The 1PLM is a special case of the 2PLM, where the item discrimination, or slope, parameter (a) is constrained to be the same across all items. The IRF for the 1PLM is presented in Equation 2.3_1.

$$P\left(X_{ij}=1|\,\theta_j\right)=\frac{\exp\left[a\left(\theta_j-b_i\right)\right]}{1+\exp\left[a\left(\theta_j-b_i\right)\right]}, \tag{2.3_1}$$

All the notations are the same as in the 2PLM. The constraint in the 1PLM IRF compared to the 2PLM IRF is the restriction of the a_i parameter to be equal for all items, i.e., $a_i = a$. Because the 1PLM is nested within the 2PLM, a model comparison test, such as the likelihood ratio test, can be conducted. The comparison between 2PLM and 1PLM was introduced in 2.2_20 and is further explained in this section. Note that the comparison of the 1PLM with the 2PLM is equivalent to investigating the relative model-data fit of the Rasch model with the 2PLM.

The 1PLM application is done using the "ltm" and "mirt" packages. Both packages use the MML estimation and assume θ follows a normal distribution. The mean and the standard deviation of the normal distribution are fixed as 0 and 1, respectively, for the model identification. This is a different estimation and model ID constraint from the "eRm" Rasch model application, which uses the CML estimation.

1PLM calibration with the "ltm" package

```
install.packages("ltm") # 2.3_1
library(ltm) # 2.3_2
setwd("c:/mystudy/RIRT") #2.3_3
```

#2.3_1. If the "ltm" package has not been installed on the computer, it must be installed. If it has previously been installed, there is no need to execute this. See #2.1_1 for details.

#2.3_2. Upon opening a new session in R, the "ltm" package must be loaded into the current R session using the "`library()`" function, similar to #2.1 _2. If any other package, e.g., "eRm" or "mirt," has been loaded in the current R session, it is recommended to users to detach those using "`detach("package:eRm")`" and "`detach("package:mirt")`."

#2.3_3. Within each new R session, set the working directory using the "`setwd()`" function. See #2.1_3.

```
ddd <- read.table("c2_rasch.dat") #2.3_4
dim(ddd) #2.3_5
```

#2.3_4. For the 1PLM analysis, use the same data that was used previously in the simple Rasch section, "`c2_rasch.dat`" (see #2.1_4). To read the data file, users can use the "`read.table()`" function to load the data into the current R session and save it as an R data frame called "ddd."

#2.3_5. The "`dim()`" command shows the number of examinees (rows) and the number of items (columns) in the data frame "`ddd`." The number of test-takers is 500 and the number of items is 20. Typing "`head(ddd)`" shows the first six rows of the data set. The default column (variable) names are "V1" through "V20." See also #2.1_5 through 2.1_12 on renaming the variables.

Item parameter estimation

```
mod.1pl <- rasch(ddd) #2.3_6
mod.1pl$conv #2.3_7
```

#2.3_6. The "`rasch()`" command executes the 1PLM where the item difficulty and common slope, or discrimination, across items is estimated; the mean and the standard deviation of the population theta (ability) distribution are fixed as 0 and 1, respectively. While it may appear that the simple Rasch model is being fit to the data with the "`rasch()`" command, this command fits the 1PLM with the discrimination freely estimated (i.e., not necessarily to the value 1.0), but constrained to be equal across all items.

#2.3_7. To verify that the calibration reached convergence, observe the value held in the object "`mod.1pl$conv`." A value of 0 indicates the estimation was completed within the specified stopping rules in the estimation process. See also #2.2_6 for more details.

```
summary(mod.1pl) #2.3_8
coef(mod.1pl) #2.3_9
```

#2.3_8. The values of the final converged log-likelihood, AIC, and BIC are shown with the item difficulties and the common discrimination across items using the "`summary()`" function. The common discrimination, or slope value is 0.9606 in this application. The "`std.err`" and "`z.vals`" are also provided. The former is the standard error of the difficulty and the common slope estimates and the latter is the ratio of the "`value`" divided by "`Std.err`."

```
> summary(mod.1pl)
Call:
rasch(data = ddd)
Model Summary:
   log.Lik       AIC         BIC
-5936.795   11915.59     12004.1
Coefficients:
                value    std.err     z.vals
Dffclt.V1      0.6316     0.1170     5.3962
Dffclt.V2     -1.0540     0.1253    -8.4135
...
Dffclt.V19    -0.2993     0.1123    -2.6650
Dffclt.V20     0.6743     0.1178     5.7242
Dscrmn         0.9606     0.0427    22.4901

Integration:
method: Gauss-Hermite
quadrature points: 21

Optimization:
Convergence: 0
max(|grad|): 0.052
quasi-Newton: BFGS
```

#2.3_9. The "coef()" function provides item parameter estimates column-wise. The first numeric column is the item difficulty estimate, and the second numeric column is the common item discrimination estimate.

```
> coef(mod.1pl)
 Dffclt Dscrmn
V1   0.6316227 0.9605987
V2  -1.0540209 0.9605987
...
V18  0.7280589 0.9605987
V19 -0.2993346 0.9605987
V20  0.6743477 0.9605987
```

Person latent trait, or ability score estimation

```
factor.scores(mod.1pl, method="EAP") #2.3_10
```

#2.3_10. EAP ability scores are requested using the "`factor.scores()`" function and the "`method="EAP"`" argument. "ltm" does not provide the ML ability estimator. See also #2. 2_9 from the 2PLM for more details.

```
> factor.scores(mod.1pl, method="EAP")
Call:
rasch(data = ddd)
Scoring Method: Expected A Posteriori
Factor-Scores for observed response patterns:
     V1  V2 V3 V4 ... V18 V19 V20 Obs   Exp     z1  se.z1
1     0   0  0  0 ...   0   0   0   1 1.753 -2.321  0.614
2     0   0  0  0 ...   0   1   0   1 0.177 -1.405  0.523
3     0   0  0  0 ...   0   0   0   1 0.051 -1.405  0.523
...
494   1   1  1  1 ...   0   1   1   1 0.239  2.149  0.559
495   1   1  1  1 ...   1   1   1   2 1.089  2.469  0.595
```

Each row in the output is a unique observed response pattern from the data, its frequency ("`Obs`"), its expected frequency ("`Exp`"), the EAP ability estimate ("`z1`"), and its standard error ("`se.z1`"). Theoretically, 2^{20} = 1,048,576 unique response patterns are possible, but the number of unique observed response patterns in this data set is 495. The 1PLM also keeps the one-to-one correspondence between the latent trait score estimate and the raw score as in the Rasch model. The second and the third response patterns in the output have the same raw sum-score of three and they have the same $\hat{\theta}$ (EAP) latent trait, or ability estimate, −1.405.

ICC plot

```
plot(mod.1pl, legend=T) #2.3_11
plot(mod.1pl, item=1:2, legend=T) #2.3_12
plot(mod.1pl, item=c(1,3,5), legend=T) #2.3_13
```

#2.3_11. The "`plot()`" function is used to plot ICCs for all items in a single plot, with item labels presented in a legend, similar to #2.2_10 for the 2PLM application using the "ltm" package.

#2.3_12. Including the additional second argument, "`item=1:2`," presents the ICCs for items 1 through 2 (i.e., 1:2) in a single plot. Like the Rasch model, the shapes of the ICCs of the 1PLM are the same, or parallel, due to the same item slope parameterization (see #2.2_11).

#2.3_13. Including the second argument, "`item=c(1,3,5)`," presents the ICCs for items 1, 3, and 5 on a single plot (see #2.2_12).

Item and test information plots

The following plots for the 1PLM application using the "ltm" package are similar to those of the 2PLM application using "ltm" illustrated in #2.2_13 through #2.2_15.

```
plot(mod.1pl, type="IIC", item=3, legend=T) #2.3_14
plot(mod.1pl, type="IIC", item=c(1,3,5), legend=T)
#2.3_15
plot(mod.1pl, type="IIC", item=0, legend=T) #2.3_16
```

#2.3_14. The "`plot()`" function is also used to plot item information curves (IIC). In this case, the IIC for item 3 is drawn in a plot using the argument "`type="ICC"`."

#2.3_15. Item information curves for items 1, 3, and 5 are drawn in a single plot using the arguments "`type="ICC"`" and "`item=c(1,3,5)`." Because of the same slope parameterization, as in the Rasch model case, the shape (spread and height) of the item information curves are the same for the three items, reaching the amount of information on the y-axis.

#2.3_16. By specifying "`item=0`," the test information curve is displayed.

Item fit

```
item.fit(mod.1pl) #2.3_17
```

#2.3_17. The item fit statistic, Q_1, which compares empirical proportions and model-produced proportions on the ten θ intervals (by default), is calculated.

```
> item.fit(mod.1pl)
Item-Fit Statistics and P-values
Call:
rasch(data = ddd)

Alternative: Items do not fit the model
Ability Categories: 10

       X^2 Pr(>X^2)
V1 17.4490   0.0421
V2 11.6972   0.2309
V3 7.1655    0.6199
...
V18 21.8781  0.0093
V19 17.3173  0.044
V20 16.7189  0.0533
```

As mentioned earlier with the 2PLM application using the "ltm" package, "`item.fit()`" offers options for the number of θ intervals, using different summary values to represent each θ interval (e.g., mean or median; median is the default), and a simulation-based item fit calculation. For details, see #2.2_16 and #2.2_17.

Person fit

```
person.fit(mod.1pl) #2.3_18
```

#2.3_18. Two person fit measures, l_0 ("L0") and l_z ("Lz"), which is the standardized version of l_0, are calculated. The last column ("Pr(<Lz)") is the probability of observing the current l_z value or lesser values based on the normal approximation. A small value of "Pr(<Lz)" (e.g., less than .001) or a small "Lz" value (e.g., Lz < −3) may be used for flagging a potential aberrant response pattern relative to the 1PLM. The first column is not a person identification index but an index for a unique response pattern. See 2.2_18.

```
> person.fit(mod.1pl)
Person-Fit Statistics and P-values
Call:
rasch(data = ddd)

Alternative: Inconsistent response pattern under the
estimated model

    V1 V2 V3 V4 ...  V18 V19 V20       L0     Lz  Pr(<Lz)
1    0  0  0  0 ...    0   0   0  -2.6216 1.4257   0.923
2    0  0  0  0 ...    0   1   0  -6.3572 1.2417   0.8928
3    0  0  0  0 ...    0   0   0  -7.6127 0.7226   0.765
...
493  1  1  1  1 ...    0   1   0  -6.6292 1.0405   0.8509
494  1  1  1  1 ...    0   1   1  -4.8699 1.2785   0.8995
495  1  1  1  1 ...    1   1   1  -2.7284 1.6020   0.9454
```

Users can use the following code to inspect the reported *p*-values by sorting the data according to the *p*-values reported.

```
p.fit <- person.fit(mod.1pl, resp.patterns=ddd)
p.fit <- cbind(1:500, ddd, p.fit$p.values)
p.fit[order(p.fit[,22]),]
```

Model-data fit and model comparison

The 1PLM is nested within the 2PLM. Constraining the item slope parameters in the 2PLM to be the same makes the 1PLM. With this hierarchical relationship, the likelihood ratio test for a relative model-data fit compares the 1PLM with the 2PLM.

```
mod.2pl <- ltm(ddd~z1, IRT.param=T) #2.3_19
anova(mod.1pl, mod.2pl) #2.3_20
```

#2.3_19. The 2PLM is fit to the data using the "`ltm()`" function. See also #2.2_5.

#2.3_20. The "`anova()`" function is used to compare the nested 1PLM to the 2PLM by supplying the objects into the function. The order of the supplied objects is important, where the smaller model is listed first. The null hypothesis of the test is the smaller model – the 1PLM in this example; the alternative hypothesis is the larger model – 2PLM in this case. The result of the likelihood ratio test ("`LRT`," "`df`," and "`p.value`"), AIC, and BIC are printed.

```
> anova(mod.1pl, mod.2pl)
Likelihood Ratio Table
             AIC      BIC  log.Lik     LRT  df  p. value
mod.1pl 11915.59 12004.10 -5936.79
mod.2pl 11931.55 12100.14 -5925.78   22.04  19     0.282
```

The *p*-value of the test is 0.282 and fails to reject the null hypothesis. This supports the decision of using the 1PLM over the 2PLM. See also #2.2_20.

```
GoF.rasch(mod.1pl, B = 100) #2.3_21
```

#2.3_21. The "ltm" package provides another type of goodness of fit which addresses the absolute model-data fit for the 1PLM using a parametric bootstrap approach taken with the Pearson's χ^2 test. "`GoF.rasch`" is only for the model evaluation of an objected created from the "`rasch()`" function. For illustration purposes, the number of replications for the bootstrapping is set to be 100, "B = 100."

```
> GoF.rasch(mod.1pl, B = 100)
Bootstrap Goodness-of-Fit using Pearson chi-squared
Call:
rasch(data = ddd)
Tobs: 1063400
# data-sets: 101
p-value: 0.287
```

If the *p*-value is less than, say, 0.05 under $\alpha = 0.05$, then the decision is to reject the fitted model. The "`p-value`" in this example is 0.287, failing to reject the reject the model. This absolute goodness of fit procedure may take some time due to the computation-intensive nature of the bootstrapping. The usefulness of this parametric resampling procedure in the case of the IRT model applications appears to be subject to further research (see, e.g., Tollenaar & Mooijart, 2003).

1PLM calibration with the "mirt" package

Prior to using the "mirt" package for calibrations, users are encouraged to detach the "ltm" package using "`detach("package:ltm")`." This is not necessary for

this section of the 1PLM calibrations, since no components of the two packages overlap, but it is good practice.

```
library(mirt) #2.3_22
setwd("c:/mystudy/RIRT") #2.3_23
```

#2.3_22. The "mirt" package is loaded using the "`library()`" function. For details on this and installing the "mirt" package, see #2.1_38 and #2.1_39.

#2.3_23. If the working directory for this R session has not been established, do this using the "`setwd()`" function, similar to #2.1_3.

```
ddd<-read.fwf("c2_2pl.dat", widths=c(10,rep(1,20)))
#2.3_24
dim(ddd) #2.3_25
ddd <- ddd[,-1] # 2.3_26
names(ddd) <- paste("It",1:20, sep="") #2.3_27
```

Though the "c2_rasch.dat" was used for the 1PLM calibrations with the "ltm" package, the "c2_2pl.dat" will be used for the 1PLM calibrations with the "mirt" package.

#2.3_24. Read in the "c2_2pl.dat," using the "`read.fwf()`" function specifying the fixed-width format. Refer to #2.2_1 through #2.2_4 for details of this function and reading in the data.

#2.3_25. The "`dim()`" command provides the number of rows and columns in the supplied object. In this example data, the number of rows, representing persons, is 1000, and the number of columns, representing variables, is 21, including the person index column.

#2.3_26. The first column, which is the person index, is removed from the data by overwriting the original object "ddd" with the same data excluding the first column.

#2.3_27. The default variable (or column) names ("V2," "V3," ..., "V21") in "ddd" are replaced by "`It1", "It2", ..."It20`"." For details, see #2.2_4. Typing "names(ddd)" after executing this command will show the new item (or variable) names.

Item parameter estimation

To run the 1PLM IRT model in "mirt," a set of specific syntax that defines the model, describing the dimensions (or factors) and the loading structure of items across the dimensions may be constructed prior to fitting the model to the data. This provides clear commands to the "`mirt()`" function about the constraints employed in the IRT model. Defining the model also provides flexibility to specify a variety of user-constrained IRT models, especially in multidimensional modeling. For the 1PLM estimation, the syntax specifies a single dimension, or factor (for the

unidimensional modeling), on which all items are loaded, and that all item slope, or discrimination, parameters are constrained to be the same.

```
spec<-'F = 1-20
CONSTRAIN=(1-20,a1)' #2.3_28
mod.1pl<-mirt(ddd, model=spec, itemtype="2PL", SE=T)
#2.3_29
mod.1pl #2.3_30
extract.mirt(mod.1pl, what="converged") #2.3_31
extract.mirt(mod.1pl, what="secondordertest") #2.3_32
coef(mod.1pl, IRTpars=T, simplify=T) #2.3_33
coef(mod.1pl, IRTpars=T) #2.3_34
coef(mod.1pl, IRTpars=T, printSE=T) #2.3_35
```

#2.3_28. This two-line command specifies the dimensional structure and item loadings of the model. The first line defines a single factor, or dimension, on which item 1 through item 20 load. The use of "`F`" is arbitrary, and users can utilize a different letter or word (e.g., "`D`" or "`D1`") if desired. The second line imposes the equality constraint that item slope parameters, "`a1`," for all items, "`1-20`," are the same. The two lines of the model specification are enclosed in single quotes and are saved under the name "`spec`," which is also arbitrary. Users can give any desired name for the object.

#2.3_29. In the 2PLM estimation by "`mirt()`," the "`model=1`" argument was used (see #2.2_28), but it is now replaced by "`model=spec`" in the 1PLM estimation that assigns the model being fit to the one defined in #2.3_28. The type of item passed to the argument is the 2PL, "`itemtype=2PL`"; the "`mirt()`" function does not have the option to specify the item type according to the 1PLM. For that reason, the 2PLM is specified with the additional model constraint defined in the model argument.

Again, "`SE=T`" is supplied to calculate the standard errors of the model parameter estimates. The default is "`SE=F`," which does not calculate the standard errors. The default standard error method when "`SE=T`" is specified is Oakes. See also #2.1_40 for more on these and other methods.

The model calibrations are stored in the object named "`mod.1pl`."

#2.3_30. Executing the "`mod.1pl`" object reports the convergence check results and other data fit indices such as AIC and BIC. See also #2.1_41.

```
> mod.1pl
Call:
mirt(data = ddd, model = spec, itemtype = "2PL", SE = T)

Full-information item factor analysis with 1 factor(s).
Converged within 1e-04 tolerance after 13 EM iterations.
mirt version: 1.28
```

```
M-step optimizer: BFGS
EM acceleration: Ramsay
Number of rectangular quadrature: 61
Latent density type: Gaussian

Information matrix estimated with method: Oakes
Condition number of information matrix = 29.25918
Second-order test: model is a possible local maximum

Log-likelihood = -11043.02
Estimated parameters: 21
AIC = 22128.03; AICc = 22128.98
BIC = 22231.1; SABIC = 22164.4
G2 (1048554) = 8554.13, p = 1
RMSEA = 0, CFI = NaN, TLI = NaN>
```

#2.3_31 and #2.3_32. Convergence check results can be extracted using the "`extract.mirt()`" function. When the "`what="converged"`" argument is supplied, the output of "`TRUE`" indicates that the iterations in the estimation ended smoothly within the specified (default) stopping criteria. When the "`what="secondordertest"`" argument is supplied, a "`TRUE`" output reports that the solutions are possible local maxima. For more on these, refer back to #2.1_42 and #2.1_43.

#2.3_33. The estimated item coefficients are reported with the "`coef()`" function. The 1PLM has a common slope parameter across all items. The common slope, or discrimination, estimate (under "`a`") is shown in the first numeric column; the estimate is 1.204 for this example data. The second numeric column (under "`b`") holds the item difficulty estimates for each item. The columns of "`g`" and "`u`" are for the pseudo-guessing and the slip parameters, respectively. They are fixed as 0 and 1, respectively, for the 1PLM.

The "`IRTpars=T`" argument in the command reports the item parameterization under the usual IRT form (i.e., discrimination and difficulty) as compared to the slope and intercept form. For more on this refer to #2.2_32 through #2.2_34. The "`simplify=T`" argument prints only the slope and difficulty estimates, without standard errors or the estimate's CI. The model identification is imposed on the person latent trait (θ) population, which is shown under "`$means`" and "`$cov`."

```
> coef(mod.1pl, IRTpars=T, simplify=T)
$'items'
         a      b  g  u
It1  1.204 -0.508  0  1
It2  1.204 -1.063  0  1
...
It19 1.204 -1.529  0  1
It20 1.204 -1.036  0  1

$means
F
0

$cov
  F
F 1
```

#2.3_34. If "`simplify=T`" is not used (which is the default), 95% CIs for the discrimination and the difficulty estimates are shown. The columns of "g" and "u" are for the pseudo-guessing and the slip parameters, respectively. They are fixed as 0 and 1, respectively, for the 1PLM identification. The last part of the output is for the latent trait, or ability population parameter estimates. Because they are fixed as 0 and 1, "`NA`" is shown under the "`$GroupPars`" 95% CI.

```
> coef(mod.1pl, IRTpars=T)
$'It1'
             a      b   g   u
par      1.204 -0.508   0   1
CI_2.5   1.133 -0.646  NA  NA
CI_97.5  1.274 -0.370  NA  NA

$It2
             a       b   g   u
par      1.204  -1.063   0   1
CI_2.5   1.133  -1.217  NA  NA
CI_97.5  1.274  -0.908  NA  NA
...
$It19
             a       b   g   u
par      1.204  -1.529   0   1
CI_2.5   1.133  -1.705  NA  NA
CI_97.5  1.274  -1.353  NA  NA
```

```
$It20
             a       b   g   u
par      1.204  -1.036   0   1
CI_2.5   1.133  -1.189  NA  NA
CI_97.5  1.274  -0.883  NA  NA

$GroupPars
     MEAN_1 COV_11
par       0      1
CI_2.5   NA     NA
CI_97.5  NA     NA
```

#2.3_35. The model item parameter estimates and their standard errors are printed together when the "`printSE=T`" argument is supplied in the "`coef()`" function. "`NA`" is printed for the standard errors of "`g`" and "`u`," which are fixed pseudo-guessing and slip, or lower and upper asymptote parameters.

```
> coef(mod.1pl, IRTpars=T, printSE=T)
$'It1'
         a       b  g    u
par  1.204  -0.508  0    1
SE   0.036   0.070  NA  NA

$It2
     a           b   g   u
par  1.204  -1.063   0   1
SE   0.036   0.079  NA  NA
...

$It20
         a       b   g   u
par  1.204  -1.036   0   1
SE   0.036   0.078  NA  NA

$GroupPars
 MEAN_1 COV_11
par   0      1
SE   NA     NA
```

In the previous report, standard errors were reported for the IRT discrimination "`a`" and difficulty "`b`" parameters; the standard errors are reported for the slope "`a`" and intercept "`d`" parameters. For more on this relationship, see #2.2_32 and #2.2_34. The standard errors of the slope and the intercept parameter estimates can

be extracted from the calibration results using the "`extract.mirt()`" command, including "`what="vcov"`."

```
round (sqrt (diag (extract.mirt(mod.1pl, what="vcov"))), 3)
```

The number of items in this application is 20. The 1PLM has 21 item parameters – the common slope and 20 intercept parameters. If this command using "`extract.mirt()`" is executed, due to the constraints in the item slope parameters, the label for the output does not follow an easy-to-understand format, as shown here. The common slope parameter label is "`a1.1.5.9.13. . . . 73.77`." The standard error the common slope is 0.036. The labels "`d2`," "`d6`," . . ., "`d70`," "`d74`," and "`d78`" are for the intercept parameters (d_i which is $-b_i/a$). For example, the standard error of the intercept for the first item is 0.083 under "`d.2`."

```
> round (sqrt (diag (extract.mirt(mod.1pl, what="vcov"))), 3)
a1.1.5.9.13.17.21.25.29.33.37.41.45.49.53.57.61.65.69.73.77   d.2
                                                            0.036 0.083
                                                              d.6  d.10
                                                            0.089 0.082
                                                             d.14  d.18
                                                            0.081 0.084
...
                                                             d.62  d.66
                                                            0.083 0.111
                                                             d.70  d.74
                                                            0.083 0.098
                                                             d.78
                                                            0.089
```

Person latent trait, or ability score estimation

```
fscores(mod.1pl, method="EAP", full.scores=T, full.
scores.SE = T) #2.3_36
```

#2.3_36. EAP ability estimator ("`method="EAP"`") is used for the person latent trait estimation using the "`fscores()`" function. "`full.scores=T`" requests the scores for all persons (as compared to unique item response patterns), and "`full.scores.SE=T`" requests the estimated standard errors. Person latent trait or ability estimates and their standard errors are under "`F1`" and "`SE_F1`," respectively. This is similar to #2.1_47 for the Rasch model calibration and #2.2_36 for the 2PLM calibration.

```
> fscores(mod.1pl, method="EAP", full.scores=T, full.
  scores.SE = T)
                 F1       SE_F1
   [1,]  0.02730556  0.3898809
   [2,]  0.83486582  0.4383009
   [3,] -1.45739924  0.4234633
   [4,] -0.50767315  0.3833241
   [5,]  0.83486582  0.4383009
   [6,]  1.35208299  0.4933259
...
 [999,] -0.15339150  0.3853798
[1000,] -1.68293428  0.4430836
```

ICC, TCC, and information plots

The "`itemplot()`" and "`itemplot()`" functions in "mirt" are used similarly for the Rasch model (#2.1_48 through #2.1_52), 2PLM (#2.2_37 through #2.2_41), here for the 1PLM, and later for the 3PLM and 4PLM.

```
itemplot(mod.1pl, 5) #2.3_37
plot(mod.1pl, type = "trace") #2.3_38
plot(mod.1pl) #2.3_39
plot(mod.1pl, type="info") #2.3_40
itemplot(mod.1pl, 3, type="info") #2.3_41
```

#2.3_37. The "`itemplot()`" function is used to print the ICC for item 5 by requesting "5" as the second argument.

#2.3_38. The "`plot()`" function is used to print the ICCs for all items using the argument, "`type="trace"`."

#2.3_39 and #2.3_40. The "`plot()`" function is used to print the TCC when no additional arguments are supplied. When "`type="info"`," the test information curve is printed.

#2.3_41. Item information curve for item 3 is drawn using the "`itemplot()`" function, where the second argument is the item requested, and "`type="info"`" requests the information curve. As in the Rasch model, the shapes of the item information curves across all items follow the same shape and only differ in location because the item slope parameters are constrained to be the same.

Model-data fit and model comparison

The model-data fit check in "mirt" are used similarly for the Rasch model (#2.1_53 through #2.1_58), 2PLM (#2.2_42 through #2.2_47), here for the 1PLM, and later for the 3PLM and 4PLM.

```
M2(mod.1pl) #2.3_42
itemfit(mod.1pl) #2.3_43
itemfit(mod.1pl, empirical.plot=2, empirical.CI=.95) #2.3_44
personfit(mod.1pl, method="EAP") #2.3_45
residuals(mod.1pl, type="LD") #2.3_46
residuals(mod.1pl, type="Q3") #2.2_47
```

#2.3 _42. The "`M2()`" function reports the absolute model-data fit test statistic, M2, in addition to descriptive fit measures such as RMSEA. The *p*-value of M2 is virtually 0, rejecting the fitted 1PLM model under $\alpha = 0.05$. See #2.1_53 for more details.

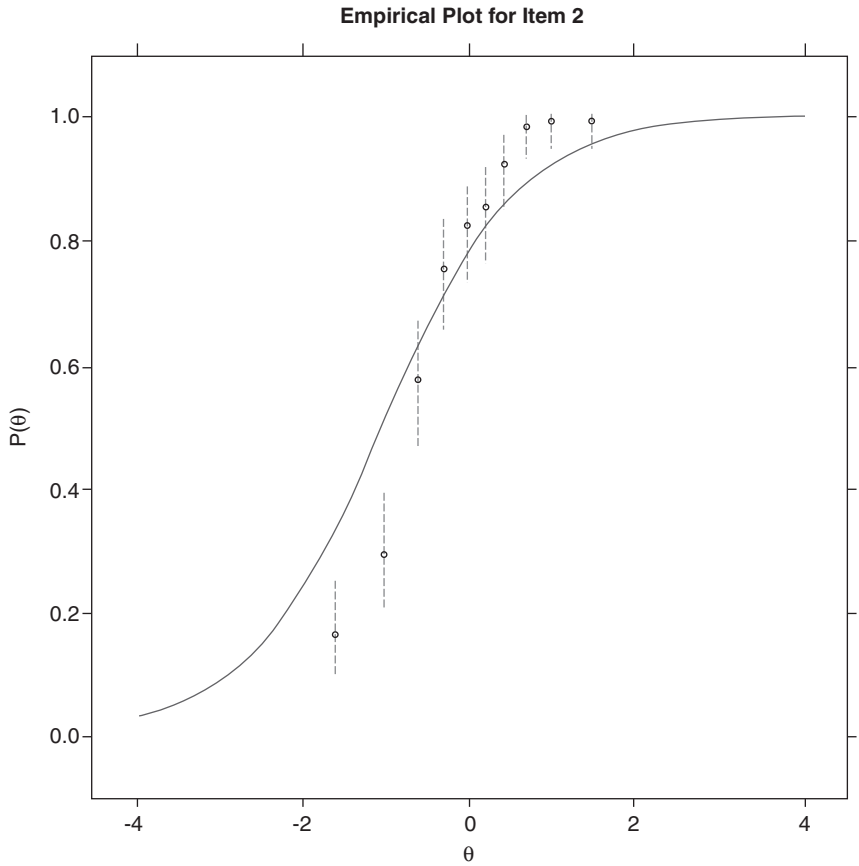
```
> M2(mod.1pl)
          M2   df  p      RMSEA     RMSEA_5    RMSEA_95     SRMSR       TLI       CFI
stats 563.49  189  0 0.04453551  0.04030528  0.04877253 0.0782851 0.9572492 0.9574742
```

#2.3_43. An item level fit statistic, S-X2, is calculated using the "`itemfit()`" function. The last column in the output, titled "`p.S_X2`," shows the *p*-values for each item's statistic. See #2.1_54.

```
> itemfit(mod.1pl)
   item    S_X2  df.S_X2  p.S_X2
1   It1  24.868       15   0.052
2   It2  40.329       15   0.000
3   It3  22.065       16   0.141
...
19 It19  19.863       15   0.177
20 It20  21.015       15   0.136
```

Eleven items are statistically flagged under the family-wise Type I error rate of 0.05, with the Bonferroni correction being poorly fitted items. Note that the data set "c2_2pl.dat" was also used for the 2PLM calibration with the "mirt" package. When the 2PLM was used for this data set, the *p*-value of the M2 test was 0.144, and the number of flagged item with the same 0.05 family-wise rate under the Bonferroni correction was 0. These comparisons indicate that the 2PLM may be a better model for the data than the 1PLM.

#2.3_44. The plot of the empirical ICC (circles with 95% CIs) and the expected model-based ICC for item 2 is plotted using the "`itemfit()`" function, specifying item 2 with the "`empirical.plot=2`" argument and the CI with "`empirical.CI=.95`." Item 2 had a near zero *p*-value for the item fit test, S-X2. The plot in Figure 2.3_44 shows the empirical ICC slope being steeper than the model-expected ICC's slope.



#2.3_45. Person fit statistics are requested with the "`personfit()`" function. The person fit measure l_z is shown under the column label "`Zh`." Small values of "`Zh`" (e.g., less than −2 or −3) may indicate potential aberrant response patterns. See #2.2_18.

Because the 1PLM was calibrated, as in the Rasch calibration case, the Rasch infit and outfit measures ("`infit`" and "`outfit`" in the output) and their standardized values ("`z.infit`" and "`z.outfit`") were also calculated.

```
> personfit(mod.1pl, method="EAP")
       outfit     z.outfit      infit       z.infit            Zh
1  1.01711767  0.153907421 1.0311384  0.221155714 -0.1369267309
2  0.96529065  0.099125947 0.9525552 -0.052732624  0.1169158448
3  0.61467981 -0.802402989 0.7150679 -1.081549757  1.0068407275
...
```

```
 998 1.11201916  0.447980726 1.0295839  0.207177295 -0.2386044664
 999 0.94712121 -0.127154379 0.9708041 -0.096527139  0.1669195311
1000 0.81949311 -0.162422098 0.9312988 -0.117993385  0.2396194407
```

To observe the output ordered by the fit statistic, use the following code.

```
p.fit <- cbind(1:1000,ddd, personfit(mod.1pl, method="EAP"))
p.fit[order(p.fit[,26]),]
```

#2.3_46 & 2.3_47. Descriptive measures of the local independence violation, LD-X2 and Q3, are calculated for every pair of items using the "`residuals()`" function. The output is a 20 by 20 matrix for each measure. Larger absolute values indicate stronger violations of local independence. See #2.1_57 and #2.1_58.

```
mod.2pl<-mirt(ddd, model=1, itemtype="2PL") #2.3_48
anova(mod.1pl, mod.2pl) #2.3_49
```

#2.3_48 and 2.3_49. We have observed that the 1PLM does not fit the data well. The 1PLM can be compared to the 2PLM using the log-likelihood ratio test because the 1PLM is nested with the 2PLM. For this, the 2PLM is calibrated to the same data using the "`mirt()`" function, with the arguments "`model=1`" and "`itemtype="2PL"`." Then the "`anova()`" command is used to conduct the likelihood ratio test and report the model fit indices, AIC, AICc, SABIC, and BIC.

```
> anova(mod.1pl, mod.2pl)
       AIC     AICc   SABIC       HQ      BIC    logLik       X2    df    p
1 22128.03 22128.98 22164.4 22167.21 22231.10 -11043.02      NaN   NaN  NaN
2 21787.23 21790.65 21856.5 21861.84 21983.54 -10853.61  378.807    19    0
```

All the information criteria such as the AIC, BIC, and SABIC support the 2PLM over the 1PLM by having smaller values. The *p*-value of the log-likelihood ratio test that compares the 1PLM (null, nested model) with the 2PLM (full, alternative model) is virtually 0 (in the last column, second row under the label "`p`"), rejecting the 1PLM in favor of the 2PLM.

2.4 3-Parameter logistic model (3PLM) application

In the 3PLM, the psychometric characteristics of an item are captured by three item parameters: item discrimination (or slope), item difficulty, and item

pseudo-guessing. The IRF of the 3PLM is given by Birnbaum (1968) and shown in Equation 2.4_1.

$$P\left(X_{ij}=1|\,\theta_j\right)=g_i+\left(1-g_i\right)\frac{\exp\left[a_i\left(\theta_j-b_i\right)\right]}{1+\exp\left[a_i\left(\theta_j-b_i\right)\right]}, \tag{2.4_1}$$

All the notations are the same as in the 2PLM, with the additional "g_i," which is the pseudo-guessing parameter of the *i*th item.

The "ltm" and "mirt" packages have the capability to perform 3PLM calibrations. The 3PLM in "ltm" is not equipped with the provision of the use of prior distributions for item parameters, which helps the estimation of a large model such as 3PLM. Specifically, the use of a prior distribution for the pseudo-guessing parameter is very important to ease the estimation challenges in the 3PLM calibration. For this reason, only the "mirt" 3PLM calibration is introduced. In a simulation study (Xu & Paek, 2016), the performance of the 3PLM using "mirt" was shown to be comparable to a commercial IRT program, flexMIRT (Cai, 2013).

3PLM calibration with the "mirt" package

```
install.packages("mirt") #2.4_1
library("mirt") #2.4_2
setwd("c:/mystudy/RIRT") #2.4_3
ddd<-read.fwf("c2_3pl.dat", widths=c(10,rep(1,30))) #2.4_4
dim(ddd) #2.4_5
ddd <- ddd[,-1] # 2.4_6
names(ddd) <- paste("MC",1:30, sep="") #2.4_7
```

#2.4._1. If the "mirt" package has not been installed on the computer, it must be done using the "`install.packages()`" function.

#2.4_2. With each new session of R, the "mirt" package needs to be loaded in the current R session using the "`library()`" function.

#2.4_3. If the users have not done so already, set the working directory and the folder where the data files exist using the "`setwd()`" function.

#2.4_4. The data file for use in the 3PLM calibrations is "c2_3pl.dat," and it should be stored in the working directory folder. It is a fixed-width file, where the columns 1–10 are for the person identification. The rest are item responses from 30 items. Because of this fixed format, "`read.fwf()`" was used where the first variable (person identification) takes ten columns and each of the next 30 columns, starting from the 11th column, are item response variables. (For more details on this file type, refer to #2.2_1.) The data are stored as a data frame named as "`ddd`." In "`ddd`," by default, "V1" through "V31" are given for the labels of the variables. This can be observed using the "`head(ddd)`" command.

#2.4_5. This "`dim()`" command reports the dimensions of the data. In this example, the number of rows, or the sample size, is 1500, and the number of columns, or the total number of variables is 31, including the person identification variable.

#2.4_6. "`ddd[,-1]`" removes the first variable (person identification variable) in the R data object "ddd" and overwrites it with the original object name, "ddd."

#2.4_7. The names of the variables in "ddd" are replaced with "MC1", "MC2", `...,` "`MC30`" (see #2.1_8 for details).

Item parameter estimation

In the 3PLM estimation, a prior distribution for the pseudo-guessing parameter (g_i) is used. It is common to use a prior distribution for the pseudo-guessing parameter in the 3PLM calibration to reduce problems of estimation convergence. The default prior distribution used and provided in the "mirt" package is the normal distribution for the logit transformation of g_i, that is, $\log[g_i/(1 - g_i)]$, which is assumed to follow a normal distribution. If the mean of the logit of g_i is chosen between −1.5 and −1.1, the expected values of g_i center between 0.18 and 0.25. The standard deviation in the prior normal distribution determines the degree of strength of the prior distribution, i.e., the extent of dispersion of g_i around its mean.

In the 1PLM calibration, a syntax was used to specify the constraint on the slope, or discrimination, parameter (see #2.3_28). In this 3PLM calibration, the prior distribution for the pseudo-guessing parameter is specified using a syntax.

```
spec<- ' F = 1-30
PRIOR = (1-30, g, norm, -1.1, 2)' #2.4_8
mirt(ddd, model=spec, itemtype="3PL", pars= "values") #2.4_9
```

#2. 4_8. The model specifications are defined using these two lines. The first line defines the single factor, or unidimensional model, with 30 items. "`F=1-30`" means that items 1 through 30 load on a single latent trait labeled "F." The second line specifies the prior distribution for the pseudo-guessing parameters, "g," of all 30 items. The normal prior distribution for the logit of g_i is specified to have a mean of −1.1 and a standard deviation of 2. A beta prior is also provided as an alternative for the pseudo-guessing parameter. The "mirt" package provides priors for the slope and the intercept parameters as well. For the slope parameter, a lognormal distribution is available, and, for the intercept parameter, a normal distribution is available. If a lognormal prior for the slope parameter is desired in addition to the prior for the pseudo-guessing parameter, the syntax for the model specification in "`spec`" is modified as shown.

```
spec<- ' F = 1-30
         PRIOR = (1-30, g, norm, -1.1, 2), (1-30, a1, lnorm, 0, 0.5)'
```

The parameters of the lognormal distribution are 0 for the mean and 0.5 for the standard deviation. The choice of prior distributions and their parameter value specifications need to be carefully made in order to minimize the "shrinkage" effect if one has no or little idea about what prior distribution is proper and the sample size is small. If readers are curious of the impact of the prior distribution with a specific sample size in a data set, they may modify the values of the prior distributions in the prior commands and compare the model parameter estimates in the output.

#2.4_9. Using the "`mirt()`" function with the "`pars="values"`" argument provides an opportunity to check whether the model was correctly specified, specifically to check if the prior(s) were correctly specified. The column under the label of "`name`" shows the labels used in the model for the slope ("`a1`"), intercept ("`d`"), pseudo-guessing ("`g`"), and slip ("`u`") parameters. Note that, in the prior specification of the pseudo-guessing parameter in the prior syntax ("`PRIOR = (1-30, g, norm, -1.1, 2)`"), the same notation for the "`g`" parameter was used. The last two columns, "`prior_1`" and "`prior_2`," indicate if the user-chosen prior(s) were correctly specified. The normal prior specification for the logit of g_i set the mean to −1.1 and the standard deviation to 2, which correspond to the last two column values for g_i. For the other item parameters, no prior was used; thus, "`NaN`" is printed. The column labeled as "`est`" indicates if the parameter is a free parameter to estimate (`"TRUE"`) or not (`"FALSE"`). The last two rows of the output hold the latent trait, or ability population distribution parameters (`"Mean"` and `"COV"`), which are fixed as 0 and 1 for the model identification.

```
> mirt(ddd, model=spec, itemtype="3PL", pars= "values")
   group item     class   name parnum      value lbound ubound   est prior.type prior_1 prior_2
1    all   MC1      dich     a1      1 0.85100000   -Inf         TRUE       none    NaN     NaN
2    all   MC1      dich      d      2 0.76434989   -Inf    Inf  TRUE       none    NaN     NaN
3    all   MC1      dich      g      3 0.24973989  0e+00      1  TRUE       norm   -1.1     0.5
4    all   MC1      dich      u      4 1.00000000  0e+00      1 FALSE       none    NaN     NaN
5    all   MC2      dich     a1      5 0.85100000   -Inf    Inf  TRUE       none    NaN     NaN
6    all   MC2      dich      d      6 0.39008000   -Inf    Inf  TRUE       none    NaN     NaN
7    all   MC2      dich      g      7 0.24973989  0e+00      1  TRUE       norm   -1.1     0.5
8    all   MC2      dich      u      8 1.00000000  0e+00      1 FALSE       none    NaN     NaN
...
117 all  MC30      dich     a1    117 0.85100000   -Inf    Inf  TRUE       none    NaN     NaN
118 all  MC30      dich      d    118 0.48773589   -Inf    Inf  TRUE       none    NaN     NaN
119 all  MC30      dich      g    119 0.24973989  0e+00      1  TRUE       norm   -1.1     0.5
120 all  MC30      dich      u    120 1.00000000  0e+00      1 FALSE       none    NaN     NaN
121 all GROUP GroupPars MEAN_1    121 0.00000000   -Inf    Inf FALSE       none    NaN     NaN
122 all GROUP GroupPars COV_11    122 1.00000000  1e-04    Inf FALSE       none    NaN     NaN
```

```
mod.3pl<-mirt(ddd, model=spec, itemtype="3PL", SE=T) #2.4_10
mod.3pl #2.4_11
extract.mirt(mod.3pl, what="converged") #2.4_12
extract.mirt(mod.3pl, what="secondordertest") #2.4_13
coef(mod.3pl, IRTpars=T, simplify=T) #2.4_14
coef(mod.3pl, IRTpars=T) #2.4_15
coef(mod.3pl, IRTpars=T, printSE=T) #2.4_16
round (sqrt (diag (extract.mirt(mod.3pl, what="vcov")) ), 3) #2.4_17
coef(mod.3pl, printSE=T) #2.4_18
```

#2.4_10. The 3PLM specified model is fit to the "ddd" data using the "mirt()" function as stored as the object "mod.3pl." In the "itemtype=" argument, "3PL" is specified. The previously created syntax "spec" is used in the "model=" option. "SE=T" requests the calculation of standard errors of the model parameter estimates.

#2.4_11. Convergence check results and other fit indices are reported when the stored object "mod.3pl" is executed. See also #2.1_41. For the 3PLM, deviance information criterion (DIC) is also calculated and shown. DIC is an analogue of AIC in the Bayesian model selection framework that evaluates relative model-data fit between different models. (See, for example, Gelman, Carlin, Stern, and Rubin (2004) for more details of DIC.) As in AIC and BIC, DIC takes into account both model complexity and the fit of the model. For model comparison, a smaller value corresponds to a better model-data fit. The DIC is calculated and reported when a prior distribution is specified for a parameter for the 3PLM estimation in "mirt." In this case the logit of the pseudo-guessing parameter, g_i took a normal prior.

```
> mod.3pl
Call:
mirt(data = ddd, model = spec, itemtype = "3PL", SE = T)

Full-information item factor analysis with 1 factor(s).
Converged within 1e-04 tolerance after 117 EM iterations.
mirt version: 1.28
M-step optimizer: BFGS
EM acceleration: Ramsay
Number of rectangular quadrature: 61
Latent density type: Gaussian

Information matrix estimated with method: Oakes
Condition number of information matrix = 1635.746
Second-order test: model is a possible local maximum

Log-posterior = -23480.33
Estimated parameters: 90
DIC = 47140.66
G2 (1073741733) = 25145.69, p = 1
RMSEA = 0, CFI = NaN, TLI = NaN>
```

Because a prior distribution is used, the objective function in the model estimation consists of the log-likelihood and the log of the prior. The "`Log-posterior`" is the sum of the log-likelihood and the log of the prior. The values of the log-likelihood and the log of the prior are −23430.3 (by typing "`extract.mirt(mod.3pl, what="logLik")`") and −50.03573 (by typing "`extract.mirt(mod.3pl, what="logPrior")`"), respectively. The sum of these values is equal to the reported "`Log-posterior = -23480.33`."

#2.4_12 and #2.4_13. Extracting the convergence check results using the "`extract.mirt()`" function prints "`TRUE`," indicating the estimation is finished within the stopping criteria used in the estimation, and that possible local maxima for the final solutions were obtained. See also #2.1_42 and #2.1_43.

#2.4_14. The item discrimination, or slope, difficulty, and pseudo-guessing parameter estimates are printed using the "`coef()`" function; here "`simplify=T`" is used to print the results in a simplified view. The last column under "`u`" is the slip, or upper asymptote parameter which is fixed to one in the 3PLM.

```
> coef(mod.3pl, IRTpars=T, simplify=T)
$'items'
          a       b      g  u
MC1   1.527   0.062  0.325  1
MC2   1.968  -0.135  0.082  1
MC3   1.412  -1.542  0.150  1
...
MC29  2.076   1.862  0.175  1
MC30  1.923   0.288  0.313  1

$means
F
0

$cov
  F
F 1
```

#2.4_15. Removing the "`simplify=T`" argument from the "`coef()`" function provides not only the parameter estimates for "`a`," "`b`," and "`g`," but also the 95% CIs. The slip parameters, "`u`," are fixed as one in the 3PLM, and "`NA`" is printed for the lower and the upper bounds of the 95% CIs.

```
> coef(mod.3pl, IRTpars=T)
$'MC1'
             a       b      g   u
par      1.527   0.062  0.325   1
CI_2.5   1.027  -0.294  0.198  NA
CI_97.5  2.026   0.418  0.452  NA

$MC2
             a       b      g   u
par      1.968  -0.135  0.082   1
CI_2.5   1.586  -0.295  0.004  NA
CI_97.5  2.349   0.025  0.161  NA
...

$MC30
             a      b      g   u
par      1.923  0.288  0.313   1
CI_2.5   1.286  0.044  0.219  NA
CI_97.5  2.560  0.533  0.407  NA

$GroupPars
     MEAN_1 COV_11
par       0      1
CI_2.5   NA     NA
CI_97.5  NA     NA
```

#2.4_16. Including "`printSE=T`" in the "`coef()`" command prints the standard errors of the item discrimination, difficulty, and pseudo-guessing parameter estimates. The "mirt" package and corresponding functions use the intercept (d_i) instead of difficulty (b_i) and the logit of g_i instead of g_i in the initial item parameter estimation, i.e., $a_i\theta + d_i$ instead of $a_i(\theta - b_i)$. For more information, see #2.2_32 and #2.2_34. The point estimates of b_i and g_i are obtained through a transformation of the intercept, d_i, and the logit of g_i, and the standard errors of the b_i and g_i estimates are calculated using the delta method.

```
> coef(mod.3pl, IRTpars=T, printSE=T)
$'MC1'
         a      b      g   u
par  1.527  0.062  0.325   1
SE   0.255  0.182  0.065  NA

$MC2
         a      b      g   u
par  1.968 -0.135  0.082   1
SE   0.195  0.082  0.040  NA
...
$MC30
         a      b      g   u
par  1.923  0.288  0.313   1
SE   0.325  0.125  0.048  NA

$GroupPars
 MEAN_1 COV_11
par   0      1
SE   NA     NA
```

#2.4_17. Extracting the error covariance matrix for the model parameters used in the estimation by "`mirt()`" is done using "`extract.mirt(mod.3pl, what="vcov")`." Again, the parameterization in "mirt" is $a_i\theta + d_i$, not $a_i(\theta - b_i)$. So, the extracted covariances are with regards to a_i, d_i, and the logit of g_i. Subsequently, the standard errors by this command are for each of these parameters. The "`extract.mirt()`" command can be replaced by the "`vcov()`" command, i.e., typing "`round (sqrt (diag (vcov(mod.3pl)) ), 3)`."

```
> round (sqrt (diag (extract.mirt(mod.3pl, what="vcov")) ), 3)
  a1.1   d.2   g.3  a1.5   d.6   g.7  a1.9  d.10  g.11
0.255 0.290 0.295 0.195 0.146 0.531 0.172 0.268 1.370
. . .

g.107  a1.109  d.110  g.111  a1.113  d.114  g.115  a1.117 d.118 g.119
0.182  0.426  0.608  0.111  0.496  0.778  0.113  0.325 0.314 0.223
```

"`a1.1`," "`d.2`," and "`g.3`" are the labels for the slope, intercept and the logit of g_i of item 1. Standard errors of each of those parameter estimates are shown under the label (e.g., the standard error of the slope for item 1 is 0.255). "`a1.5`," "`d.6`," and "`g.7`" are the labels for the slope, intercept and the logit of g_i of item 2. This pattern is repeated through the last item. The labels are designated by the "`mirt()`" function, not by the users. Note again that the label starting with "`g`" is not for the pseudo-guessing parameter g_i, but for the logit of g_i.

#2.4_18. As an alternative to the extraction method shown earlier, one may use the "`coef()`" function with the "`printSE=T`" argument and exclude the "`IRTpars=T`" argument (or use "`IRTpars=F`" by default). This prints the point estimates in the slope and intercept forms of the item parameters, i.e., a_i, d_i, and logit(g_i), with their standard errors.

```
> coef(mod.3pl, print=T)
$'MC1'
        a1       d  logit(g)  logit(u)
par  1.527  -0.095   -0.732       999
SE   0.255   0.290    0.295        NA

$MC2
        a1       d  logit(g)  logit(u)
par  1.968   0.265   -2.411       999
SE   0.195   0.146    0.531        NA
...

$MC30
        a1       d  logit(g)  logit(u)
par  1.923  -0.554   -0.786       999
SE   0.325   0.314    0.223        NA

$GroupPars
 MEAN_1 COV_11
par   0      1
SE   NA     NA
```

If readers compare the output of the estimated item parameters in the slope and intercept form ("IRTpars = F"; a_i = "`a1`," d_i = "`d`," *logit*(g_i) = "`logit(g)`") with those from the traditional IRT parameterization ("IRTpars = T"; a_i = "`a`," b_i = "`b`," g_i = "`g`"), the following relationships are observed for item 1.

`a1=a` → 1.527 = 1.527
`d = -a * b` → –0.095 = (−1)⋆ 1.527⋆0.062
`logit(g) = log(g/(1-g))` → –0.732 = log(0.325 / (1–0.325))

Similar relationships exist for all items.

Person latent trait, or ability score estimation

```
fscores(mod.3pl, method="EAP", full.scores=T, full.scores.
SE = T) #2.4_19
```

#2.4_19. The person ability parameters, estimated using the EAP method, is produced with the "`fscores()`" function and specifying "`method="EAP"`." The column titled "`F1`" shows the latent trait estimates for each individual (since "`full.scores=T`" is given), which are the means of posterior distributions of test-takers, and the column titled "`SE_F1`" shows the standard deviations of the test-takers' posterior distributions, which are treated as standard errors of the estimates. For more details, see #2.1_47.

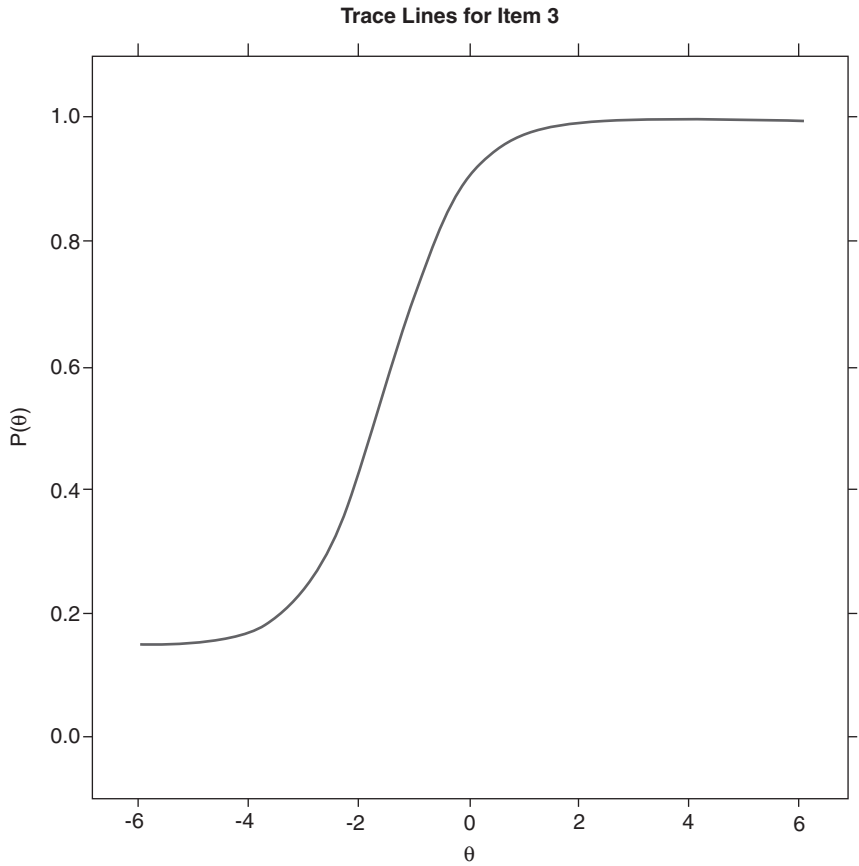
```
> fscores(mod.3pl, method="EAP", full.scores=T, full.
  scores.SE = T)
                F1      SE_F1
   [1,]  0.3893076  0.3208395
   [2,]  0.9479370  0.3533298
   [3,]  0.4160670  0.3051453
...
[1499,] -0.2082407  0.3730651
[1500,]  1.0362108  0.3585604
```

ICC, TCC, and information plots

```
itemplot(mod.3pl, 3) #2.4_20
plot(mod.3pl, type = "trace") #2.4_21
plot(mod.3pl) #2.4_22
plot(mod.3pl, type="info") #2.4_23
itemplot(mod.3pl, 3, type="info") #2.4_24
```

ICC, TCC, and information plots produced with the "mirt" package for the Rasch model were provided in #2.1_48 through #2.1_52. Plots of the 2PLM were provided in #2.2_37 through #2.2_41, and those of the 1PLM were provided in #2.3_37 through #2.3_41.

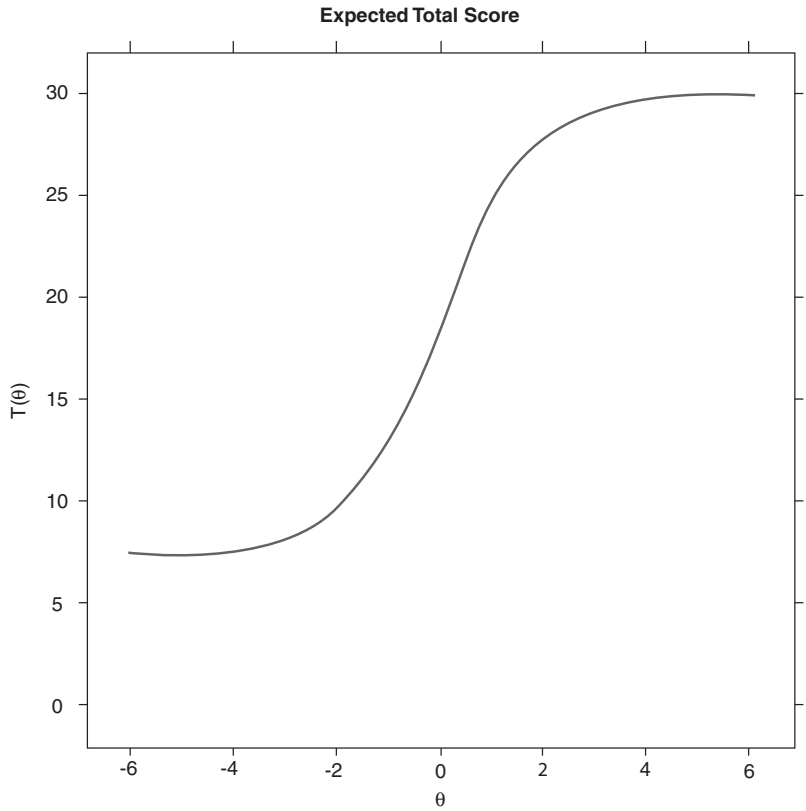
#2.4_20. The ICC for an individual item, item 3 in this example, is produced using the "`itemplot()`" function. Since the 3PLM is used in this section, the estimated lower asymptote, or pseudo-guessing parameter, is non-zero in the ICC. From the IRT parameter estimate output (#2.4_14), the $\hat{g}_3 = 0.150$, and the lower asymptote observed in the ICC in Figure 2.4_20 is at approximately $P(\theta) = 0.150$. Still, the upper asymptote of the ICC equals 1.0.



#2.4_21. ICCs for all items are shown using the "`plot()`" function and specifying "`type = "trace"`."

#2.4_22. The TCC is shown also using the "`plot()`" function with no additional arguments. Due to the non-zero lower asymptotes of the estimated IRFs (and ICCs) of the 3PLM, the tail of TCC is also non-zero. This indicates that even very low ability test-takers are expected to have a non-zero total sum score. When the θ latent trait, or ability, is equal to or less than –3, the expected total test score is close to about 7.5 in this application. The sum of the estimated $\hat{g}_i$ is equal to 7.452 (using the following code), which is the expected total sum score for $\theta = -\infty$.

```
sum(coef(mod.3pl, IRTpars=T, simplify=T)$items[,3])
```



#2.4_23. The test information curve is produced using the "`plot()`" function with "`type="info"`." In the 3PLM, although the item slope values still play a major role in determining the magnitude of the test information across the θ scale, test information is affected by all item parameters (item slope, difficulty, and the pseudo-guessing parameter values).

#2.4_24. Item information for item 3 is shown using the "`itemplot()`" function with the specified item, "3," and "`type="info"`." The location of the maximum item information on the theta scale is no longer at $\theta = b_i$, but near the item's difficulty due to the non-zero lower asymptote (Lord, 1980; Magis, 2013).

Model-data fit

```
M2(mod.3pl) #2.4_25
itemfit(mod.3pl) #2.4_26
itemfit(mod.3pl, empirical.plot=15, empirical.CI=.95) #2.4_27
personfit(mod.3pl, method="EAP") #2.4_28
residuals(mod.3pl, type="LD") #2.4_29
residuals(mod.3pl, type="Q3") #2.4_30
```

#2.4_25. The result of the absolute model-data fit test M2 is printed along with other fit statistics.

```
> M2(mod.3pl)
           M2  df     p RMSEA RMSEA_5  RMSEA_95       SRMSR       TLI  CFI
stats 352.222 375 0.7952366   0    0 0.006506372  0.02118802  1.001596    1
```

The M2 is considered a chi-square variate. The *p*-value of the M2 for this data set is 0.795, which corresponds to failing to reject the fitted model, 3PLM, for this data set. RMSEA is calculated using the M2 value. It should be noted that the conventionally known rules of thumb for the descriptive model-fit indices such as RMSE, TLI, and CFI are not applicable in this ordered response item factor analysis technique, which is the IRT model application. Unfortunately, there is no established cut-off for the categorical ordinal response data analysis with IRT or item factor analysis.

A study by Xu, Paek, and Xia (2017) provides some insight on the behavior of the RMSEA, which is based on the M2 chi-square value. Their results indicate that the value of the RMSEA for a good fit should be much closer to 0 than the conventionally known cut off of 0.05 for a good fit. Note that RMSEA can still be used for different model comparison purposes as well, i.e., a model with a smaller value could be considered a better fitting model to the data, although the degree of "better" is still not clearly known. See also #2.1_53.

#2.4_26. The results for item fit test S-X2 are reported with the "`itemfit()`" function. The *p*-value is shown in the last numeric column under "`p.S_X2`." See also #2.1_54.

```
> itemfit(mod.3pl)
   item     S_X2  df.S_X2  p.S_X2
1   MC1   40.436       20   0.004
2   MC2   20.593       18   0.300
3   MC3   17.797       18   0.469
...
29  MC29  22.488       20   0.315
30  MC30  13.196       20   0.869
```

With the Bonferroni correction under the family-wise Type I error rate of 0.05, there was no item flagged statistically.

#2.4_27. Including "`empirical.plot=15`" for item 15 and "`empirical.CI=.95`" to include the empirical ICC with 95% CIs in the "`itemfit()`" function produces a plot that compares the empirical ICC (circles) with the model-based ICC (solid line). The 95% CIs for the circles (empirical ICC) are also shown as dotted vertical bars. See also #2.1_55.

#2.4_28. The person fit statistic, l_z is calculated for each person under "`Zh`" using the "`personfit()`" function. A small value, such as −3 or −2, may be used for flagging potential aberrant response patterns. See #2.2_18.

```
> personfit(mod.3pl, method="EAP")
               Zh
1  -0.6839888
2   0.2376007
3  -0.1147056
...
1498   0.67679022
1499   0.54658384
1500  -0.34040031
```

To report the observed data and corresponding l_z statistics in increasing order of l_z, the following code can be used. There are two observations (numbers 1106 and 1325) with Zh < −3, which may indicate a problematic response string.

```
p.fit <- cbind(1:1500,ddd, personfit(mod.3pl, method="EAP"))
p.fit[order(p.fit[,32]),]
```

#2.4_29. LD-X2, which is an index for the local independence violation, is calculated and shown for every pair of items using the "`residuals()`" function and specifying "`type="LD"`." The results are shown in a 30 by 30 matrix (for this example with 30 items). A very large absolute value (e.g., greater than 10) implies a strong evidence for the violation of the local independence assumption. See #2.1_57.

#2.4_30. Q3 is another index for the local independence violation. Q3 is calculated and shown for every pair of items in a 30 by 30 matrix (for this example with 30 items) form using the "`residuals()`" function and specifying "`type="Q3"`." A large value of Q3 indicates strong evidence that the assumption of local independence is violated. See #2.1_58.

Traditional relative model comparisons (e.g., 2PLM vs. 3PLM or 1PLM vs. 3PLM) using a likelihood ratio test or information criteria, which use the log-likelihood (e.g., AIC and BIC), are not appropriate in this application due to the use of a prior specification for the pseudo-guessing parameter in the 3PLM estimation. Instead, the RMSEA from the M2 test application, for example, could be utilized to compare, say, the 2PLM with the 3PLM. A better fitted model shows a smaller value of RMSEA.

If the 3PLM is estimated successfully, without any convergence issues for a large sample size and without any item prior distributions in the MML approach, the log-likelihood ratio test and the popular information criteria of AIC and BIC are applicable. However, a convergence issue in the estimation of the 3PLM could

take place even with a large sample size (more often than in the estimation of the 2PLM). For this reason, a prior for the pseudo-guessing parameter is typically used in practice.

2.5 4-Parameter logistic model (4PLM) application

The 3PLM has the lower asymptote parameter to accommodate very low ability examinees' pseudo-guessing behaviors. The 4PLM adds one more feature to the 3PLM, which is the upper asymptote as a free parameter. The IRF of the 4PLM is given by:

$$P\left(X_{ij}=1|\,\theta_j\right)=g_i+(u_i-g_i)\frac{\exp\left[a_i\left(\theta_j-b_i\right)\right]}{1+\exp\left[a_i\left(\theta_j-b_i\right)\right]}, \qquad (2.1.5\_1)$$

All the notations are the same as they were in the 3PLM, with the addition of the parameter u_i. The slip parameter, or upper asymptote, u_i, is the ith item's upper bound, or maximum probability of a correct response along the θ scale. The slip parameter, or upper asymptote, was designed to accommodate high ability examinees' mistakes (incorrect answers) due to their carelessness or some other reasons. For more details of the 4PLM, see, for example, Barton and Lord (1981); Loken and Rulison (2010); and Cheng and Liu (2015). The "mirt" package is used to illustrate the 4PLM application.

4PLM calibration with the "mirt" package

```
install.packages("mirt") #2.5_1
library("mirt") #2.5_2
setwd("c:/mystudy/RIRT") #2.5_3
ddd<-read.table("c2_4pl.dat") #2.5_4
dim(ddd) #2.5_5
names(ddd) <- paste("Item",1:35, sep="") #2.5_6
```

#2.5_1. If the "mirt" package is not installed, use the "`install.packages()`" command to download the "mirt" package from the selected CRAN mirror. See #2.1_1 as well.

#2.5_2. Once the "mirt" package is downloaded, is must be loaded with each new R session using the "`library()`" function.

#2.5_3. Before starting any new project or section, set up the working directory where data files and files that are exported from the R are stored using the "`setwd()`" function. This obviates typing a directory specification for reading and exporting files.

#2.5_4. The data file "c2_4pl.dat" is read and stored as an R object "`ddd`" using "`read.table()`." The default variable (item) label is "V1" through "V35."

#2.5_5. The "`dim()`" command provides the number of rows, or test-takers, and the number of columns, or variables. For this data, The number of test-takers is 2000 and the number of items (or variables) is 35.

#2.5_6. The variable names are changed from "V1," "V2," . . . , "V35" to "Item1," "Item2," . . . , "Item35."

Item parameter estimation

As in the estimation of the 1PLM and 3PLM, the 4PLM specification is established prior to fitting the model to the data. In the "`spec`" object, the model dimensionality with the item loading structure and the prior distributions for item parameters are specified.

```
spec<- 'F = 1-35
 PRIOR = (1-35, g, norm, -1.4, 2),(1-35, u, norm, 2.2, 2))' #2.5_7
mirt(ddd, model=spec, itemtype="4PL", pars="values") #2.5_8
```

#2.5_7. In the "`spec`" object, the model dimensionality with the item loading structure and the prior distributions for item parameters are specified in the two lines enclosed with single quotations. Unidimensionality with all 35 items is specified by "`F = 1-35`." The "`PRIOR`" command is used for the pseudo-guessing parameter "`g`" and the slip parameter "`u`."

The logit of g_i is assumed to follow a normal distribution with a mean of −1.4 and standard deviation of 2. This specification establishes the pseudo-guessing parameter values to be around 0.2. The logit of u_i (i.e., the logit of the slip parameter) is also a normal distribution with a mean of 2.2 and standard deviation of 2. The specification sets the slip parameters to be around 0.9. For the slope and the intercept parameters, no prior is specified. The use of the priors for the pseudo-guessing and the slip parameters are to reduce the chance of estimation problems. The pseudo-guessing and the slip parameters are not easy to estimate; these parameters pertain mostly to the outer ends of the θ scale. Because a relatively small amount of data at both tails of the person ability distribution is provided from the responses in the data, the estimation of these parameters is challenging. The use of priors for these two parameters are recommended in the estimation of the 4PLM.

#2.5_8. When the data are fit to the model specified, and the "`pars="values"`" is used, the output provides the model details. Here, users can check if the prior specification(s) and other model specifications are done as intended by the user.

```
> mirt(ddd, model=spec, itemtype="4PL", pars= "values")
 group item class name parnum value lbound ubound est prior.type prior_1
  prior_2
1   all Item1  dich    a1   1 0.851000000   -Inf   Inf  TRUE none  NaN NaN
2   all Item1  dich     d   2 0.709734865   -Inf   Inf  TRUE none  NaN NaN
3   all Item1  dich     g   3 0.197816111  0e+00     1  TRUE norm -1.4   2
4   all Item1  dich     u   4 0.900249511  0e+00     1  TRUE norm  2.2   2
5   all Item2  dich    a1   5 0.851000000   -Inf   Inf  TRUE none  NaN NaN
6   all Item2  dich     d   6 0.864561484   -Inf   Inf  TRUE none  NaN NaN
7   all Item2  dich     g   7 0.197816111  0e+00     1  TRUE norm -1.4   2
8   all Item2  dich     u   8 0.900249511  0e+00     1  TRUE norm  2.2   2
...
137 all Item35 dich    a1 137 0.851000000   -Inf   Inf  TRUE none  NaN NaN
138 all Item35 dich     d 138 0.432040938   -Inf   Inf  TRUE none  NaN NaN
139 all Item35 dich     g 139 0.197816111  0e+00     1  TRUE norm -1.4   2
140 all Item35 dich     u 140 0.900249511  0e+00     1  TRUE norm  2.2   2
141 all GROUP GroupPars MEAN_1 141    0.000000000   -Inf    Inf  FALSE  none   NaN  NaN
142 all GROUP GroupPars COV_11 142    1.000000000   1e-04    Inf  FALSE  none   NaN  NaN
```

The rows indexed by 1 through 4 are the first item's parameters. The slope "`a1`" and the intercept "`d`" do not have priors, so the "`prior.type`" column reports "`none`," and the last two columns ("`prior_1`" and "`prior_2`") show "`NaN`." For the (logit of) "`g`" and "`u`" parameters, normal priors were used. On these rows, the "`prior.type`" show "`norm`" and the "`prior_1`" and "`prior_2`" indicate the normal distribution parameters specified by the user.

```
mod.4pl<-mirt(ddd, model=spec, itemtype="4PL", SE=T,
  technical=list(NCYCLES=5000)) #2.5_9
mod.4pl #2.5_10
extract.mirt(mod.4pl, what="converged") #2.5_11
extract.mirt(mod.4pl, what="secondordertest") #2.5_12
coef(mod.4pl, IRTpars=T, simplify=T) #2.5_13
coef(mod.4pl, IRTpars=T) #2.5_14
coef(mod.4pl, IRTpars=T, printSE=T) #2.5_15
round (sqrt (diag (extract.mirt(mod.4pl, what="vcov")) ),
  3) #2.5_16
coef(mod.4pl, print=T) #2.5_17
```

#2.5_9. The 4PLM estimation is requested using the "`itemtype=4PL`" argument in the "`mirt()`" function. Again, the default standard error estimation method is Oakes when "`SE=T`" is used. The estimation of the 4PLM is challenging without having priors for g_i and u_i. Also, it is not rare to see convergence issues in the 4PLM estimation even with the priors for the pseudo-guessing and the slip parameters. It is recommended to have a large sample size with the use of item prior distributions, specifically the priors for the pseudo-guessing and the slip parameters. If convergence issues remain, adding a prior for the slope parameter and increasing the permitted number of estimation iterations may help to achieve convergence.

In this example, the sample size is 2000. Priors for the pseudo-guessing and slip parameters were used while the maximum number of estimation iteration was increased to 5000 (by using "`technical=list(NCYCLES=5000)`"). The default of the maximum number of iteration is 500 for the EM estimation, which appears to be not large enough in the author's experience to reach convergence in the estimation of the 4PLM. Depending on an application, the number of iterations necessary for convergence could be several thousands.

#2.1.5_10. "`mod.4pl`" prints the convergence results and some fit statistics.

```
> mod.4pl
Call:
mirt(data = ddd4, model = spec, itemtype = "4PL", SE = T,
  technical = list(NCYCLES = 5000))

Full-information item factor analysis with 1 factor(s).
Converged within 1e-04 tolerance after 764 EM iterations.
mirt version: 1.28
M-step optimizer: BFGS
EM acceleration: Ramsay
Number of rectangular quadrature: 61
Latent density type: Gaussian

Information matrix estimated with method: Oakes
Condition number of information matrix = 2517.042
Second-order test: model is a possible local maximum

Log-posterior = -41607.98
Estimated parameters: 140
DIC = 83495.96
G2 (1e+10) = 52580.8, p = 1
RMSEA = 0, CFI = NaN, TLI = NaN
```

The output shows 764 (EM) estimation iterations were required for convergence in this application, which is greater than the default value of 500. As in the 3PLM case, the log-posterior is the sum of the log-likelihood and the log prior parts, which are due to the use of the prior distributions.

Though the 3PLM is nested within the 4PLM, conventional likelihood test and information criteria such as AIC and BIC are not applicable for the model comparison because of the use of the prior for the pseudo-guessing and slip parameters and because the value of the slip parameter in the 3PLM is on the boundary of the parameter space. See also #2.4_11.

#2.5_11 & 2.5_12. The convergence checks printed from the "`extract.mirt()`" function with "`what="converged"`" or "`what="secondordertest"`" print "`TRUE`," meaning that the convergence checks did not detect particular issues. See also #2.1_42 and #2.1_43.

#2.5_13. The point estimates of the item parameters are printed in a simplified version using "`coef()`" and the "`simplify=T`" argument. "`a`," "`b`," "`g`," and

"u" represent the discrimination (or slope), difficulty, pseudo-guessing (or lower asymptote), and slip (or upper asymptote) parameters, respectively. The latent trait, or ability, population parameters of the normal distribution are fixed with a mean of 0 and a standard deviation of 1 for resolving the model identification.

```
> coef(mod.4pl, IRTpars=T, simplify=T)
$'items'
            a       b      g      u
Item1   1.289   1.895  0.150  0.895
Item2   2.269  -1.169  0.427  0.911
...
Item34  1.658  -0.438  0.230  0.883
Item35  3.016   0.656  0.252  0.840

$means
F
0

$cov
  F
F 1
```

#2.5_14. Here, the "coef()" function prints the point estimates of item parameters and their 95% CIs.

```
> coef(mod.4pl, IRTpars=T)
$'Item1'
             a      b      g      u
par      1.289  1.895  0.150  0.895
CI_2.5   0.410  1.033  0.092  0.524
CI_97.5  2.168  2.758  0.208  1.266

$Item2
             a       b       g      u
par      2.269  -1.169   0.427  0.911
CI_2.5  -0.471  -2.283  -0.024  0.877
CI_97.5  5.009  -0.056   0.878  0.946
. . .

$Item35
             a      b      g      u
par      3.016  0.656  0.252  0.840
CI_2.5   0.381  0.348  0.202  0.678
CI_97.5  5.651  0.964  0.301  1.002

$GroupPars
     MEAN_1  COV_11
par       0       1
CI_2.5   NA      NA
CI_97.5  NA      NA
```

#2.5_15. If the standard error output is desired, the "`printSE=T`" argument can be supplied into "`coef()`."Then the command line prints the item parameter estimates and their standard errors. The latent trait or ability population parameters are fixed with a mean of 0 and a standard deviation of 1; thus, "`NA`" is printed for their SEs.

```
> coef(mod.4pl, IRTpars=T, printSE=T)
$'Item1'
         a      b     g      u
par  1.289  1.895  0.15  0.895
SE   0.448  0.440  0.03  0.189

$Item2
         a       b      g      u
par  2.269  -1.169  0.427  0.911
SE   1.398   0.568  0.230  0.018
...

$Item35
         a      b      g      u
par  3.016  0.656  0.252  0.840
SE   1.344  0.157  0.025  0.083

$GroupPars
   MEAN_1  COV_11
par     0       1
SE     NA      NA
```

#2.5_16. If the error covariance matrix for the model parameters is desired, this command, which utilizes "`extract.mirt()`." extracts it. However, the error covariance is based on the "mirt" model slope and the intercept parameterization form, $a_i\theta + d_i$, rather than the IRT parameterization, $a_i(\theta - b_i)$. (For more on that, refer to #2.2_32 and #2.2_34.) Furthermore, "`mirt()`" uses the logit of g_i and the logit of u_i in the estimation. The intercept (d_i), logit of g_i, and logit of u_i are transformed to produce the traditional IRT parameter values of b_i, g_i, and u_i, while their standard errors are calculated using the delta method.

The error covariance matrix extracted with this command shows the variances of the a_i, d_i, logit of g_i, and logit of u_i. Though the labels in the output show "`g.3`" and "`u.4`," for example, the values are the variances for the logit of g_i and logit of u_i in the extracted error covariance matrix. The current command takes the square root of the extracted variances that produces the standard errors, rounded to three decimals.

```
> round (sqrt (diag (extract.mirt(mod.4pl, what="vcov"))
  ), 3)
 a1.1    d.2    g.3    u.4   a1.5    d.6    g.7   u.8
0.448  0.614  0.233  2.016  1.398  0.678  0.941 0.218 ...

 a1.137  d.138  g.139  u.140
  1.344  0.630  0.134  0.614
```

The standard errors of the first item parameters are shown under "`a1.1`," "`d.2`," "`g.3`," and "`u.4`." The standard errors of the last item are shown under "`a1.137`," "`d.138`," "`g.139`," and "`u.140`."

#2.5_17. If the standard errors for the model following the "`mirt()`" parameterization, where the slope and intercept form are used with the logit of g_i, and logit of u_i, is desired, "`coef()`" with the "`printSE=T`" argument prints the standard errors of the point estimates of the slope (`a1`), intercept (`d`), logit of `g`, and logit of `u`.

```
> coef(mod.4pl, printSE=T)
$'Item1'
        a1       d   logit(g)  logit(u)
par  1.289  -2.444    -1.735      2.145
SE   0.448   0.614     0.233      2.016

$Item2
        a1      d  logit(g) logit(u)
par  2.269  2.653   -0.294     2.330
SE   1.398  0.678    0.941     0.218
...

$Item35
        a1       d  logit(g)  logit(u)
par  3.016  -1.978   -1.090      1.656
SE   1.344   0.630    0.134      0.614

$GroupPars
  MEAN_1  COV_11
par    0       1
SE    NA      NA
```

Person latent trait, or ability score estimation

```
fscores(mod.4pl, method="EAP", full.scores=T, full.scores.
SE = T) #2.5_18
```

#2.5_18. EAP scores are calculated and shown for each person using the "`fscores()`" function, including the appropriate arguments. The EAP ability estimates are printed under the "`F1`" column, and their standard errors

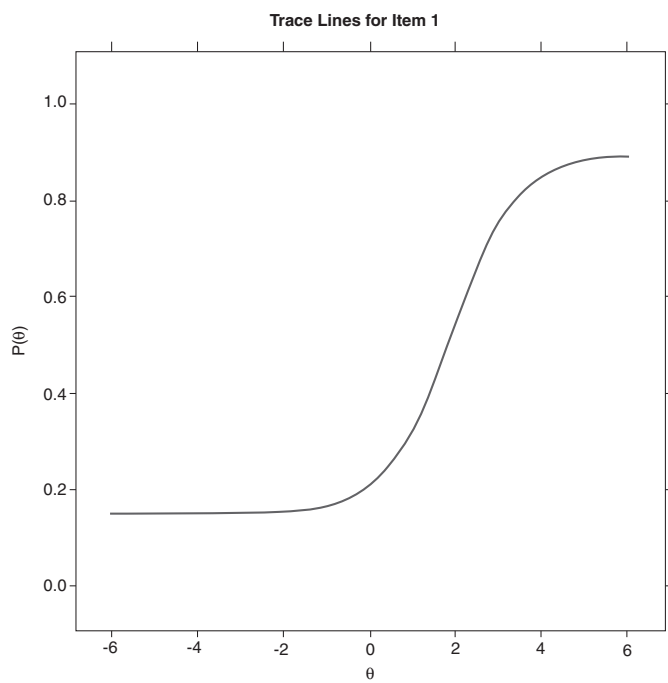
are under the "SE_F1" column, which are the standard deviations of the person posterior latent trait, or ability, distributions.

```
> fscores(mod.4pl, method="EAP", full.scores=T, full.
  scores.SE = T)
                      F1        SE_F1
   [1,]   0.4725321144   0.4023065
   [2,]   1.2968462395   0.5127255
   [3,]   0.3111692674   0.3788728
...
[1999,]  -0.3474311857   0.4280389
[2000,]   0.7919822977   0.3784682
```

ICC, TCC, and information plots

```
itemplot(mod.4pl, 1) #2.5_19
plot(mod.4pl, type = "trace") #2.5_20
plot(mod.4pl) #2.5_21
plot(mod.4pl, type="info") #2.5_22
itemplot(mod.4pl, 3, type="info") #2.5_23
```

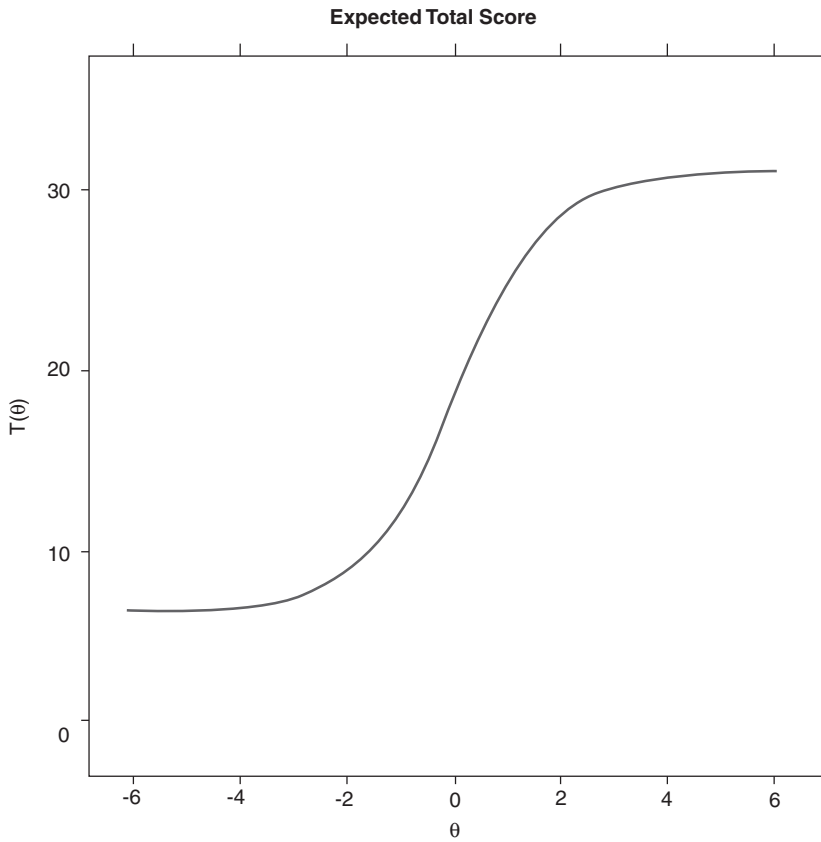
#2.5_19. The "`itemplot()`" function is used to print the ICC for item 1. The lower asymptote (guessing parameter) and the upper asymptote (slip parameter) are not 0. They are 0.15 and 0.995 for the item 1. See the 3PLM ICC in Figure 2.4_20 for comparison, which has only a lower asymptote estimate and the upper, asymptote is fixed to 1.0.



#2.5_20. The ICCs of all items are drawn on a single plot using the "`plot()`" function with "`type= "trace"`."

#2.5_21. The TCC is drawn using the "`plot()`" function without additional arguments. Compared to the 3PLM TCC (Figure 2.4_22), the 4PLM TCC has a lifted left tail at the low ability range and a flattened right tail at the high ability range due to the freely estimated pseudo-guessing and slip parameters. The sum of the estimated g_i is equal to the value of the lower asymptote of the TCC, and the sum of the estimated u_i equals the value of the upper asymptote of the TCC in Figure 2.5_21. These can be verified using the following code to sum the parameters across all items.

```
sum(coef(mod.4pl, IRTpars=T, simplify=T)$items[,3])
sum(coef(mod.4pl, IRTpars=T, simplify=T)$items[,4])
```



#2.5_22. The test information plot is produced using the "`plot()`" function with "`type= "info"`." The highest information is achieved near zero on the ability, θ, scale.

#2.5_23. Item information for item 1 is shown using the "`itemplot()`" function, specifying the item number, and including "`type="info"`." The highest information is around the item's difficulty estimate, which is 1.895, though all item parameter values affect the shape of the item information curve.

Model-data fit

```
M2(mod.4pl) #2.5_24
itemfit(mod.4pl) #2.5_25
itemfit(mod.4pl, empirical.plot=15, empirical.CI=.95)
 #2.5_26
round (personfit(mod.4pl, method="EAP"), 4) #2.5_27
residuals(mod.4pl, type="LD") #2.5_28
residuals(mod.4pl, type="Q3") #2.5_29
```

#2.5_24. The limited information overall model-data fit test, M2, is calculated using the "`M2()`" function. Its chi-square value is 446.3219 with the *p*-value of 0.92, failing to reject the fitted model. The RMSEA, which is based on the chi-square value of the M2, is virtually 0. See #2.1_53 and #2.4_25.

```
> M2(mod.4pl)
           M2  df         p RMSEA RMSEA_5    RMSEA_95     SRMSR      TLI CFI
stats 446.3219 490 0.9218373     0       0 0.002749359 0.0184665 1.003433   1
```

#2.5_25. The results of the S-X2 item fit test are printed using the "`itemfit()`" function. The last column under the label "`p.S_X2`" shows the *p*-value for each item. See #2.1_54.

```
> itemfit(mod.4pl)
     item     S_X2  df.S_X2  p.S_X2
1   Item1   25.284       23   0.336
2   Item2   20.247       24   0.683
3   Item3   16.185       23   0.847
...
34  Item34  14.044       23   0.926
35  Item35  16.245       23   0.845
```

With the family-wise Type I error rate of 0.05 using the Bonferroni correction, no item is flagged statistically as a poor fitting item in this example.

#2.5_26. A graphical check of item fit is done within the "`itemfit()`" function. A model-based ICC is compared with the empirical ICC in a plot. The empirical ICC is represented by the circles and their dotted vertical bars are the 95% CIs. See also #2.1_55.

#2.5_27. A person fit measure, l_z, is computed for each person using the "`personfit()`" function. In the output, "`Zh`" is the l_z. A small value (e.g., less than, say, −3 or −2) may indicate a potential aberrant response pattern. The "`round()`" command was used to print the l_z values up to the four decimals. See #2.2_18.

```
> round(personfit(mod.4pl, method="EAP"),4)
           Zh
1     -1.0066
2     -0.1672
3      0.2365
...
1999  -1.9873
2000   0.8268
```

#2.5_28 and #2.5_29. LD-X2 and Q3, which are the indices for the violation of the local independence assumption, are calculated for every pair of items using the "`residuals()`" function. The results are shown in a matrix form. In this application, the result for each index is a 35 by 35 matrix, since the data contain 35 items. A large absolute value indicates a strong possibility of the violation of the local independence. An absolute of 10 may be used for the value of the standardized LD-X2 for flagging an item pair showing clear sign of local independence. An absolute value of 0.2 was suggested as a rule of thumb for Q3 when the number of items is equal or greater than 17 (Yen, 1993). See #2.1_57 and #2.1_58 for more details.

The 4PLM is the largest model, where the 3PLM, the 2PLM, and the 1PLM models are nested. In spite of this hierarchical relationship among these models, the conventional likelihood ratio test or the use of information criteria of AIC and BIC is not applicable for the comparison of the 4PLM with the rest of the model. The 4PLM has specified priors for the pseudo-guessing and slip parameters, whereas the 3PLM set priors for the pseudo-guessing parameter and the slip parameter was fixed; in the 2PLM the upper and lower asymptotes are fixed with the values, which are on the boundary of the parameter space. This makes the likelihood-based model comparisons method no longer valid between the 4PLM with specified priors and all nested models, or between the 3PLM with specified priors and all nested models, whenever the prior for the pseudo-guessing parameter and/or the prior for the slip parameter are used in the estimation of the 3PLM and the 4PLM. If the 4PLM and the 3PLM are estimated without any item priors, the use of the likelihood ratio test and corresponding information criteria is valid; however, the estimation of the 3PLM and 4PLM are challenging without the use of the priors for the pseudo-guessing and the slip parameters.

References

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

Andersen, E. B. (1973). A goodness of fit test for the Rasch model. *Psychometrika*, *38*, 123–140.

Andrich, D. (2004). Controversy and the Rash model: A characteristic of incompatible paradigm? *Medical Care*, *42*, I-7–1–16.

Baker, F. B., & Kim, S-H. (2004). *Item response theory parameter estimation technique* (2nd edition). New York, NY: Marcel Dekker, Inc.

Barton, M. A., & Lord, F. M. (1981). *An upper asymptote for the three-parameter logistic item-response model* (No. 150–453). Princeton, NJ: Educational Testing Service.

Birnbaum, A. (1957). *Efficient design and use of tests of a mental ability for various decision making problems* (Series Report No. 58-16). Randolph Air Force Base, TX: USAF School of Aviation Medicine.

Birnbaum, A. (1968). Some latent trait models and their use in inferring an examinee's ability. In F. M. Lord & M. R. Novick (Eds.), *Statistical theories of mental test scores* (pp. 397–479). Reading, MA: Addison-Wesley.

Cai, L. (2008). SEM of another flavour: Two new applications of the supplemented EM algorithm. *British Journal of Mathematical and Statistical Psychology*, *61*, 309–329.

Cai, L. (2013). *flexMIRT version 2: Flexible multilevel multidimensional item analysis and test scoring* (Computer software). Chapel Hill, NC: Vector Psychometric Group.

Cai, L., du Toit, S. H. C., & Thissen, D. (2011). *IRTPRO: User guide*. Lincolnwood, IL: Scientific Software International.

Chalmers, R. P. (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software*, *48*, 1–29. doi:10.18637/jss.v048.i06

Chalmers, R. P. (2018). Numerical approximation of the observed information matrix with Oakes' identity. *British Journal of Mathematical and Statistical Psychology*, *71*, 415–436.

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

Cheng, Y., & Liu, C. (2015). The effect of upper and lower asymptotes of IRT models on computerized adaptive testing. *Applied Psychological Measurement*, *39*, 551–565.

Christensen, K. B., Bjørner, J. B., Kreiner, S., & Petersen, J. H. (2002). Testing unidimensionality in polytomous Rasch models. *Psychometrika*, *67*(4), 563–574.

Christensen, K. B., & Kreiner, S. (2007). A monte carlo approach to unidimensionality testing in polytomous Rasch models. *Applied Psychological Measurement*, *31*, 20–30.

de Ayala, R. J. (2009). *The theory and practice of item response theory*. New York, NY: Guilford Press.

Drasgow, F., Levine, M.V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology*, *38*, 67–86.

Fischer, G. H., & Molenaar, I. (1995). *Rasch models – Foundations, recent developments, and applications*. New York, NY: Springer.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis* (2nd edition). Boca Raton, FL: CRC Press.

Levine, M.V., & Rubin, D. B. (1979). Measuring the appropriateness of multiple-choice test scores. *Journal of Educational Statistics*, *4*, 269–290.

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

Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Hillsdale, NJ: Lawrence Erlbaum.

Magis, D. (2013). A note on the item information function of the four-parameter logistic model. *Applied Psychological Measurement*, *37*, 304–315.

Mair, P., & Hatzinger, R. (2007a). CML based estimation of extended Rasch models with the *eRm* package in R. *Psychology Science*, *49*, 26–43.

Mair, P., & Hatzinger, R. (2007b). Extended Rasch modeling: The eRm package for the application of IRT models in R. *Journal of Statistical Software*, *20*, 1–20. Retrieved from www.jstatsoft.org/v20/i09

Mair, P., Hatzinger, R., & Maier, M. J. (2018). *eRm: Extended Rasch Modeling. 0.16–2*. Retrieved from http://erm.r-forge.r-project.org/

Maydeu-Olivares, A., & Garcia-Forero, C. (2010). Goodness-of-fit testing. *International Encyclopedia of Education*, 7, 190–196.

Maydeu-Olivares, A., & Joe, H. (2005). Limited- and full-information estimation and goodness-of-fit testing in 2n contingency tables: A unified framework. *Journal of the American Statistical Association*, *100*, 1009–1020.

Maydeu-Olivares, A., & Joe, H. (2006). Limited information goodness-of-fit testing in multidimensional contingency tables. *Psychometrika*, *71*, 713–732.

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

Orlando, M., & Thissen, D. (2003). Further investigation of the performance of S-X^2: An item fit index for use with dichotomous item response theory models. *Applied Psychological Measurement*, *27*, 289–298.

Paek, I., & Cai, L. (2014). A comparison of item parameter standard error estimation procedures for unidimensional and multidimensional item response theory modeling. *Educational and Psychological Measurement*, *74*, 58–76.

R Core Team. (2018). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from www.R-project.org/

Rasch, G. (1960, 1980). *Probabilitistic models for some intelligence and attainment tests*. Chicago, IL: University of Chicago Press.

Rizopoulos, D. (2006). ltm: An R package for latent variable modelling and item response theory analyses. *Journal of Statistical Software*, *17*, 1–25. Retrieved from www.jstatsoft.org/v17/i05/

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

Sclov, S. (1987). Application of model-selection criteria to some problems in multivariate analysis. *Psychometrika*, *52*(3), 333–343.

Smith Jr., E.V., & Smith, R. M. (2004). *Introduction to Rasch measurement*. Maple Grove, MN: JAM Press.

Snijders, T. A. B. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika*, *66*, 331–342.

Sugiura, N. (1978). Further analysis of the data by Akaike's information criterion and the finite corrections. *Communications in Statistics, Theory and Methods*, *A7*, 13–26.

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*, 412–439.

Tollenaar, N., & Mooijart, A. (2003). Type I errors and power of the parametric goodness-of-fit test: Full and limited information. *British Journal of Mathematical and Statistical Psychology*, *56*, 271–288.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, *54*, 427–450.

Wilson, M. (2011). Some notes on the term: 'Wright Map'. *Rasch Measurement Transactions*, *25*(3), 1331.

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

Wright, B. D., & Masters, G. N. (1990). Computation of OUTFIT and INFIT statistics. *Rasch Measurement Transactions*, *3*(4), 84–85.

Xu, J., & Paek, I. (2016). *The 3PL model parameter recovery with different item prior distributions when using flexMIRT and the mirt package in R*. Paper presented at the 61st annual meeting of Florida Educational Research Association, Florida Educational Research Association, Lakeland, FL.

Xu, J., Paek, I., & Xia, Y. (2017). Investigating the behaviors of M2 and RMSEA2 in fitting a unidimensional model to multidimensional data. *Applied Psychological Measurement*, *41*, 632–644.

Yen, W. M. (1981). Using simulation results to choose a latent trait model. *Applied Psychological Measurement*, *5*, 245–262.

Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three parameter logistic model. *Applied Psychological Measurement*, *8*, 125–145.

Yen, W. M. (1993). Scaling performance assessments: Strategies for managing local item dependence. *Journal of Educational Measurement*, *30*, 187–213.

3

UNIDIMENSIONAL IRT WITH POLYTOMOUS ITEM RESPONSES

Item responses could be more than dichotomous outcomes. Responses for constructed-response partial credit items and Likert style responses are scored polytomously. Numbers assigned for different categorically ordered responses are typically 0, 1, 2, and 3 (or 1, 2, 3 and 4) for a four-option or four-level set of categories in an item. This chapter introduces the applications of the IRT models that handle polytomously scored item responses.

In a model for dichotomously scored item responses, the item response function (IRF) is the essential unit of modeling the item responses. In polytomous response modeling, the basic unit of modeling is at the category level, which is the category response function (CRF), or category probability function. Just as a dichotomous IRF can be graphically displayed using an item characteristic curve (ICC), a polytomous CRF can be graphically displayed using a category characteristic curve (CCC), or category probability curve. (There is one exception to this; the graded response model does utilize the cumulative category response probabilities. This is further explained in section 3.4.)

Four different polytomous IRT models will be discussed in this chapter: partial credit model (PCM), rating scale model (RSM), generalized partial credit model (GPCM), and graded response model (GRM). The nominal response model (NRM) is briefly presented at the end. The IRT packages used within the R software in this chapter are the "eRm," "ltm," and "mirt." For the PCM, "eRm" and "mirt" were used. For the RSM, "mirt" was used. Both the "ltm" and "mirt" were used for the GPCM and the GRM. The NRM is discussed using "mirt."

"eRm" has the capability to fit the RSM, but it was not considered here because the authors have experienced model estimation problems with warning messages when multiple data sets were tried. "ltm" has an option for specifying CCC of PCM, but it appears to fix the variance of the population ability distribution as one, which makes a more reduced model than traditional PCM in the marginal maximum likelihood (MML) estimation context which "ltm" employs.

3.1 Partial credit model (PCM) application

The partial credit model (PCM; Masters, 1982) is a Rasch family item response model that can handle item responses with more than two categories. The PCM is a direct extension of the dichotomous simple Rasch model, and it is classified as a divided-by-total model (Thissen & Steinberg, 1986). The CRF, and CCC, for category $k \in \{0,1,2, \ldots, m_i\}$, i.e., responding in category k of an item with $m_i + 1$ possible category options, starting with 0, of the ith item has the following form.

$$P\left(X_{ikj} = k \middle| \theta_j\right) = \frac{\exp\sum_{h=0}^{k}\left(\theta_j - b_{ih}\right)}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left(\theta - b_{ih}\right)}, \tag{3.1_1}$$

where X_{ikj} is the jth person's response in category k to the ith item. $P(X_{ikj} = k \mid \theta_j)$ is sometimes written as $P_{ikj}(\theta)$ for short. θ_j is the jth person's latent trait or ability score. The item parameter is b_{ih}; this is the threshold parameter for the ith item in category k (where $h = 0, \ldots k$), hence it is sometimes simply denoted b_{ik}. Function b_{ih} is a point on the θ scale, where the CCC for the response in category k, $P_{ik}(\theta)$, intersects with the CCC for the response in category $k - 1$, $P_{i(k-1)}(\theta)$. Therefore, it is sometimes referred to as a category-intersection parameter to provide an easy way of remembering what the parameter represents. For the category $k = 0$, $b_{i(k=0)} = 0$ and $\theta_j - b_{i(k=0)}$ is defined as 0. As the earlier formula shows, the number of item parameters is m_i when an item has $m_i + 1$ number of categories. For instance, a four-option item has three threshold parameters: $b_{i(k=1)}$, $b_{i(k=2)}$, and $b_{i(k=3)}$.

Equation 3.1_1 is under the "threshold" parameterization. Another parameterization, called the "location plus deviation" parameterization, is sometimes used. The threshold parameter, or category-intersection parameter, b_{ik}, is sometimes decomposed into two components: an overall location parameter for an item plus the deviation parameter for a specific category, as presented in Equation 3.1_2.

$$b_{ik} = b_i + d_{ik}, \tag{3.1_2}$$

where b_i is the location parameter for item i, which is a measure of the overall difficulty of the item, and d_{ik} is the deviation of category k from the overall location of b_{ik} in reference to b_i, (i.e., $d_{ik} = b_{ik} - b_i$). Under the location plus deviation parameterization, $\Sigma_k\, b_{ik} = 0$ is used typically as part of the model identification. An alternative model constraint for the PCM is to set the first threshold of the first item to 0, i.e., $b_{1(k=1)} = 0$. When the number of categories is two, then the PCM is reduced to the dichotomous simple Rasch model.

We want to caution readers that the terms to describe b_{ik} and d_{ik} vary. Some researchers call b_{ik} a "step" parameter and d_{ik} a "threshold" parameter (see Wright & Masters, 1982). We call d_{ik} the deviation parameter to convey clearly that d_{ik} is the difference between b_{ik} and b_i.

PCM calibration with the "eRm" package

```
install.packages("eRm") #3.1_1
library(eRm) #3.1_2
setwd("c:/mystudy/RIRT/") #3.1_3
ddd<-read.fwf("c3_pcm.dat", width=c(rep(1,5))) #3.1_4
names(ddd)<-paste("I", 1:5, sep="") #3.1_5
dim(ddd) #3.1_6
head(ddd) #3.1_7
summary(ddd) #3.1_8
apply(ddd, 2, table) #3.1_9
```

#3.1_1. The "eRm" package is installed onto the computer using "`install.packages()`." This only needs to be done once on a computer system. See #2.1_1.

#3.1_2. The "eRm" package must be loaded into the current R session using the "`library()`" function. See #2.1_2.

#3.1_3. The working directory where data and R output files are stored is established using "`setwd()`." For the PCM, the "`c3_pcm.dat`" file should be saved into the folder established as the working directory. See #2.1_3.

#3.1_4. Read the item response data file "`c3_pcm.dat`" into the R session using "`read.fwf()`" since the item responses are arranged with a fixed format style having no space between columns. The data are saved it as an R object called "`ddd`." Each of the five columns represents an item and each row represents a person. See #2.2_1.

#3.1_5. Replace the default variable, or item, names by new names ("I1" through "I5"). See #2.1_8.

#3.1_6. The dimensions of "`ddd`" are providing using the "`dim()`" function. The number of rows and the number of columns are shown; in this application, 900 test-takers and 5 items. See #2.1_5.

#3.1_7. The first six rows of the data are displayed using "`head()`." These data are polytomous with four category response options: 0, 1, 2, or 3.

#3.1_8. The "`summary()`" command produces descriptive statistics (the first quartile, second quartile [median], third quartile, minimum, maximum, and mean) for each item's responses. The value of the mean for each item is the item easiness, or the item difficulty in classical test theory. For example, in a Likert style psychological test with response options, 0 = *Strongly Disagree*, 1 = *Disagree*, 2 = *Agree*, and 3 = *Strongly Agree*, the higher the item mean, or item easiness, the easier it was for test-takers to respond in a higher category to endorse, or agree with, the item statement.

```
> summary(ddd)
       I1              I2              I3              I4              I5
Min.   :0.000 Min.   :0.000 Min.   :0.000 Min.   :0.000 Min.   :0.000
1st Qu.:0.000 1st Qu.:1.000 1st Qu.:1.000 1st Qu.:0.000 1st Qu.:0.000
Median :1.000 Median :1.000 Median :2.000 Median :1.000 Median :2.000
Mean   :1.024 Mean   :1.484 Mean   :1.943 Mean   :1.397 Mean   :1.529
3rd Qu.:2.000 3rd Qu.:2.000 3rd Qu.:3.000 3rd Qu.:2.000 3rd Qu.:3.000
Max.   :3.000 Max.   :3.000 Max.   :3.000 Max.   :3.000 Max.   :3.000
```

Unfortunately, the "`summary()`" function in R does not include the standard deviation of the item responses. If the means and the standard deviations of items are desired, the "`apply()`" function can be useful for applying a specific function across rows or columns of a data set. The third argument is the operation, "`mean`" or "`sd`," for the mean or standard deviation of the data, respectively. The second argument indicates if the operation is applied across the rows, "`1`," or across the columns "`2`."

```
apply(ddd, 2, mean) # For the mean calculation across columns
apply(ddd, 2, sd) # For the standard deviation calculation across columns
```

#3.1_9. To check that all categories were utilized for each item, i.e., checking items for any unused category (null category), the "`apply()`" command can be used. The first argument is the data set, the second argument is "`2`" to request the operation across columns, and the third argument applies the "`table`" function to each column of the data. If a table for a single item is desired, "`table(ddd$I1)`" can be used, here to request the table by the item name, I1.

```
> apply(ddd, 2, table)
    V1   V2   V3   V4   V5
0  353  181   92  242  236
1  274  281  171  224  209
2  171  259  333  269  198
3  102  179  304  165  257
```

If any item's output does not display a frequency for all four categories, for this four-category example, the item and category may be problematic in the IRT analysis. All five items have no zero-frequency category in this application.

Before estimating an IRT model in R, we recommend that users rescore, or downcode, the data if null categories are found by removing the null category from the sequence of categories. Take, for example, an item that theoretically has five response categories: 0, 1, 2, 3, and 4. If categories 0 and 2 are never used, i.e., they are null categories, then the non-null categories of 1, 3, and 4 are rescored as 0, 1, and 2. Although this is a straightforward treatment of the null categories, this may not always be the most satisfactory solution. See Wilson and Masters (1993) for their proposal on handling null categories using a modified item parameterization for the PCM. Currently, no universally accepted approach for all polytomous response IRT models exists for the null categories. It is recommended that readers pay full attention in the early developments of a test to the item development, the optimal number of response options, or categories, and category descriptions for an item to mitigate the chance to observe this "null" category issue. When the null category is observed, readers may also be prompted to more closely evaluate the item and its categories for a better understanding of why there were no responses within a category.

Item parameter estimation

```
mod.pcm<-PCM(ddd, sum0 = F) #3.1_10
mod.pcm$conv #3.1_11
thresholds(mod.pcm)$threshtable$'1' #3.1_12
cbind(thresholds(mod.pcm)$threshpar, thresholds(mod.
  pcm)$se.thresh) #3.1_13
```

#3.1_10. The PCM is fit to the "ddd" data, and item parameters are estimated using the "`PCM()`" function. The first threshold of the first item ($b_{1(k=1)}$) is constrained to be 0 for the model identification constraint by specifying the option "`sum0 = F`." The estimation result is saved as an object called "`mod.pcm`."

The "eRm" uses the conditional maximum likelihood (CML) estimation. A popular model constraint of the Rash family model when the CML estimation method is to impose a constraint on the item parameters. As in the simple Rasch model, either the average of item parameters or the first item parameter of the first item is set to equal 0. For a straightforward understanding of the model identification imposed to the item parameters and the output, the authors prefer to use the constraint of the first threshold equal to 0 in the PCM application when using the "eRm" package. This allows the rest of the item parameters and person ability parameters to be estimated in reference to $b_{1(k=1)}$.

#3.1_11. The PCM convergence check result is displayed through "`mod.pcm$conv`." An output value of "`1`" indicates successful convergence, and no specific issue was detected under the default convergence criteria.

#3.1_12. The "`thresholds()`" command produces the parameter estimates of b_{ik}, which are the threshold parameter estimates and the overall item location, or difficulty. Specifying "`$threshtable$'1'`" provides the output as a matrix, rather than a list or other object type.

```
> thresholds(mod.pcm)$threshtable$'1'
       Location Threshold 1  Threshold 2  Threshold 3
I1   0.757580792   0.0000000   0.84016632    1.4325761
I2   0.062194359  -1.0091013   0.16114421    1.0345402
I3  -0.643925513  -1.5291272  -0.89319037    0.4905410
I4   0.241977890  -0.4118716  -0.05097397    1.1887793
I5   0.009683789  -0.4169371   0.10777690    0.3382115
```

The first numeric column is the overall "`Location`," or overall item difficulty; the remaining "`Threshold`" columns are the threshold estimates (e.g., $\hat{b}_{1(k=1)}$, $\hat{b}_{1(k=2)}$, and $\hat{b}_{1(k=3)}$ for item 1). The first threshold of the first item is fixed as 0 for the model identification. The overall item difficulty of an item is defined as the average of all threshold parameters. The first item overall difficulty, or "`Location`," is 0.7576, which is the average of the three threshold estimates (0.0000, 0.8402, and 1.4326).

If the first threshold of the first item happens to be the highest, then, because it is set to be 0, the other item thresholds will be estimated with a negative value. From the model-identification point of view, this is not a problem because the origin of the θ scale is arbitrary. However, some may find the negative item parameter estimates in all (or a majority of) items but the first item peculiar. In this case, the parameters could be rescaled by subtracting a constant from all item and person parameter estimates, which is the average of the overall location parameter estimates of all items. This sets the mean of the overall difficulty of items, or, simply speaking, the overall difficulty of the test, to 0.

For example, suppose that there is a three-item test and each item has four categories (0, 1, 2, and 3). Also, suppose that item threshold parameter estimates using the constraint of $b_{i(k=1)} = 0$ are 0, 0.4, 0.8 for the first item, −3.3, −2.3, −1.3 for the second item, and −2.2, −1.1, −0.9 for the third item. Then the overall item location, or difficulty parameter estimates, which are equal to the average of all threshold parameters, are 0.4, −2.3, and −1.4 for the first, second, and third items, respectively. To conduct the rescaling, the grand mean of the three overall item difficulty parameter estimates, −1.1, is subtracted from $\hat{b}_{ik}$. (Also, to place person ability on the same scale, $\hat{\boldsymbol{\theta}}$, −1.1 is subtracted from all estimates.) These steps are illustrated here. This simple rescaling that sets the average of the overall item difficulties to be 0 may appeal more in some applications.

Typing "`thresholds(mod.pcm)`" or "`thresholds(mod.pcm)$threshtable`" will also provide the same output as #3.1_12, which contains the item location and deviation parameters (b_i and d_{ik}, respectively). However, the current command in #3.1_12 produces the results as a matrix, which allows further operations when necessary, while these shorter commands do not. For example, suppose the current output is saved as an R object labeled as "`rrr`" as shown.

```
rrr<-thresholds(mod.pcm)$threshtable$'1'
mean(rrr[,1])
rrr-mean(rrr[,1])
```

If the average difficulty of a test is sought, the location parameter values are extracted ("`rrr[,1]`") and the mean of them is calculated ("`mean(rrr[,1])`"). In this example, the mean is 0.0855. The aforementioned rescaling process that sets the overall difficulty of the test to 0 can be conducted with ease by implementing the third line as well, which subtracts the mean of all overall location parameters from the estimated item location and threshold parameters. The rescaled item parameter estimates are shown here.

```
> rrr-mean(rrr[,1])
       Location  Threshold 1  Threshold 2  Threshold 3
I1   0.67207853  -0.08550226   0.75466405    1.3470738
I2  -0.02330790  -1.09460355   0.07564195    0.9490379
I3  -0.72942778  -1.61462948  -0.97869263    0.4050388
I4   0.15647563  -0.49737391  -0.13647623    1.1032770
I5  -0.07581847  -0.50243933   0.02227463    0.2527093
```

#3.1_13. The output for this command shows both item threshold parameter estimates (in the first numeric column) and their standard errors (in the second numeric column).

```
> cbind(thresholds(mod.pcm)$threshpar, thresholds(mod.
  pcm)$se.thresh)
                          [,1]       [,2]
thresh beta I1.c1  0.00000000 0.0000000
thresh beta I1.c2  0.84016632 0.1576391
thresh beta I1.c3  1.43257606 0.1671750
thresh beta I2.c1 -1.00910129 0.1351391
thresh beta I2.c2  0.16114421 0.1264066
thresh beta I2.c3  1.03454016 0.1412999
...
thresh beta I5.c1 -0.41693707 0.1335477
thresh beta I5.c2  0.10777690 0.1356705
thresh beta I5.c3  0.33821154 0.1374057
```

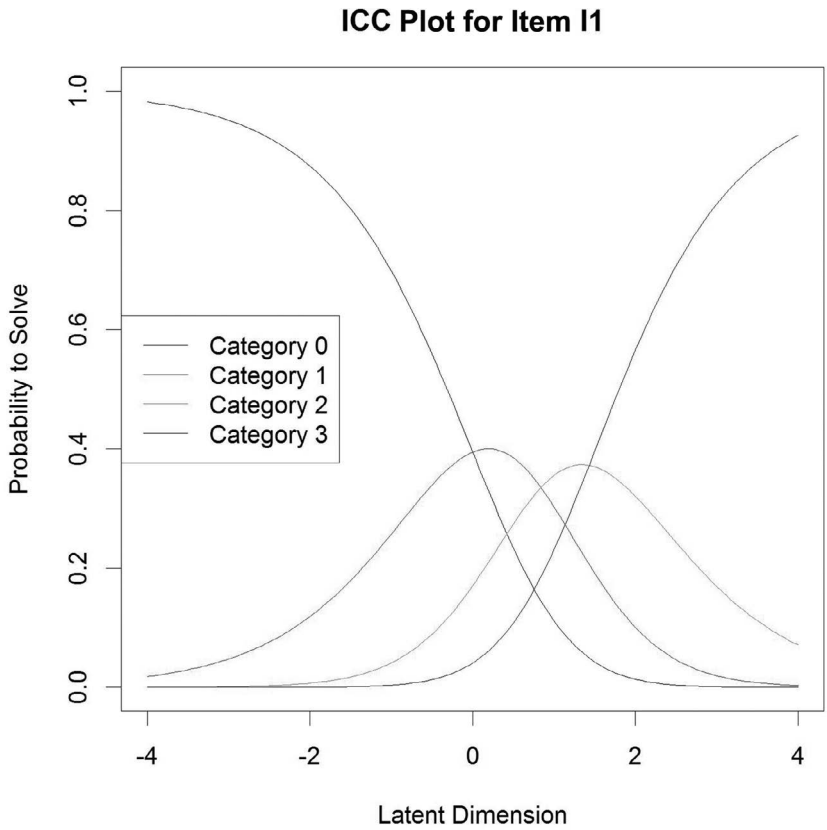
Typing "`summary(mod.pcm)`" also produces the estimation results; however, they are for the original model parameterization used in the estimation in the "eRm" package and do not follow the usual PCM parameterization.

Category characteristic curve (CCC) plot

In the dichotomous response modeling, the IRF was displayed graphically with an ICC. In the polytomous response modeling, the response function for a category is plotted using a category characteristic curve (CCC). Sometimes a CCC is also called a "trace line," although no universally accepted nomenclature exists.

```
plotICC(mod.pcm, 1) #3.1_14
```

#3.1_14. The CCC of a specified item can be requested using "`plotICC()`" with the desired item in the second argument. "`plotICC(mod.pcm)`" will provide a graphical user interface for all item CCCs, where users must press the return key to advance through the displays. A subset of items can be also selected by the command "`plotICC(mod.pcm, 1:3)`," for example, which provides the CCCs for items 1, 2, and 3. Unlike the simple Rasch case in the "eRm" package, there is no provision of empirical category characteristic curves compared with the model-based CCCs in the PCM application (shown in #2.1_18).



Person latent trait, or ability score estimation

```
p.pcm<-person.parameter(mod.pcm) #3.1_15
p.pcm #3.1_16
coef(p.pcm) #3.1_17
round(cbind(p.pcm$thetapar$NAgroup1, p.pcm$se.
  theta$NAgroup1), 3) #3.1_18
plot(p.pcm) #3.1_19
```

#3.1_15 and #3.1_16. The "`person.parameter()`" command provides the maximum likelihood (ML) estimates for person latent trait or ability scores. There were five items and each item response ranged from 0 and 3. The minimum observed sum score is 0 and the maximum observed sum score is 15 (5 × 3). For the perfect score and all zero score, no finite ML estimate is possible. An interpolation method is used to provide finite ability estimates for those zero and perfect scores, and "NA" is reported for the "`Std. Error.`" In the Rasch family model application, the observed

score and the ability estimate have the one-to-one relationship, i.e., the same observed score has only the same ability estimate.

```
> p.pcm
Person Parameters:
Raw Score     Estimate  Std.Error
        0  -3.21643297         NA
        1  -2.41023367  1.0181894
        2  -1.67059744  0.7445929
        3  -1.20498122  0.6312095
          ...
       14   2.56074144  1.0170702
       15   3.36097380         NA
```

#3.1_17. The use of "`coef()`" prints person ability estimates for all test-takers, including those who have zero and perfect raw sum scores.

```
> coef(p.pcm)
        P1           P2          P3
1.82791632  -0.27991927  0.72989374 ...

...

      P898         P899        P900
3.36097380  -2.41023367  1.02386414
```

#3.1_18. The individual person latent trait or ability score estimates and their standard errors can be extracted from the "`p.pcm`" object using "`$thetapar$NAgroup1`" and "`$se.theta$NAgroup1`," respectively. Combining these extractions column-wise, using "`cbind()`" prints the person ability estimates in the first numeric column and their standard errors in the second numeric column. Using the "`round()`" command, estimates are shown rounded to three decimals.

```
> round(cbind(p.pcm$thetapar$NAgroup1, p.pcm$se.
  theta$NAgroup1), 3)
       [,1]    [,2]
P1    1.828   0.738
P2   -0.280   0.507
P3    0.730   0.526
...
P899 -2.410   1.018
P900  1.024   0.562
```

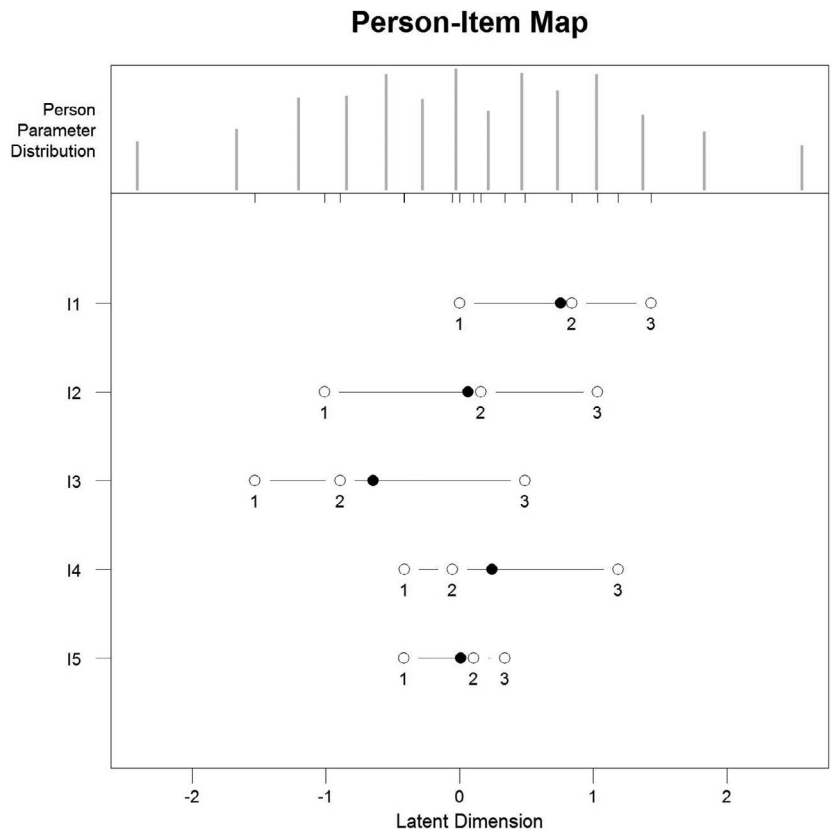
#3.1_19. A plot of the person parameters is constructed using the "`plot()`" function, where the x-axis is the observed sum score scale and the y-axis is the person ability estimate scale.

Person-item map

```
round (thresholds(mod.pcm)$threshtable$'1', 2) #3.1_20
plotPImap(mod.pcm) #3.1_21
```

#3.1_20 and #3.1_21. The overall difficulty and the threshold parameter estimates are shown by #3.1_20, which is the same command as #3.1_12 with the addition of "`round()`" for requesting the estimates rounded to two decimals for simplicity. This was repeated to facilitate the understanding of the person-item map produced by "`plotPImap()`."

Figure 3.1_21 depicts a "Wright Map." The top pane displays the distribution of person latent trait, or ability, score estimates, $\hat{\theta}$. The bottom pane displays threshold estimates by a white circle and the overall item difficulty, or location estimates, by a black circle.



This plot is a graphical summary of the item and the person estimates.

Model-data fit

```
alrt<-LRtest(mod.pcm); alrt #3.1_22
Waldtest(mod.pcm) #3.1_23
plotGOF(alrt, conf=list()) #3.1_24
itemfit(p.pcm) #3.1_25
personfit(p.pcm) #3.1_26
```

#3.1_22. An overall model-data fit is conducted using the "`LRtest()`" function. Its result is saved as "alrt" and shown. Including "`;`" between commands allows a catenation of multiple commands. This "`LRtest()`" splits the test-takers into two groups based on the median of observed score, by default. Then the estimated parameters are compared for the two groups. The *p*-value of the test in this application is 0.245, and the conclusion is to fail to reject that the fitted model is invariant over the subgroups. See #2.1_25 for a similar test with dichotomous items fit to the simple Rasch model.

#3.1_23. An additional model-data fit check at the item level is the Wald test. The test-takers are split based on the median of the observed score, and the Wald test is conducted for checking parameter invariance of each item across groups. "eRm" conducts the test with the original parameterization used in the estimation, not with the threshold parameterization. See #2.1_28 for a similar test with dichotomous items fit to the simple Rasch model.

```
> Waldtest(mod.pcm)
Wald test on item level (z-values):
             z-statistic  p-value
beta  I1.c1        0.525    0.599
beta  I1.c2       -0.140    0.889
beta  I1.c3       -1.387    0.165
...
beta  I5.c1       -0.858    0.391
beta  I5.c2       -2.255    0.024
beta  I5.c3       -1.249    0.212
```

No item parameters were flagged under the family-wise Type I error rate of 0.05 using the Bonferroni correction.

#3.1_24. The invariance of items across the two groups created by the observed score median split may be investigated graphically using the "`plotGOF()`" function. The 95% confidence ellipses are drawn for each item's estimate within the two groups. If invariance holds for the item, the identity line passes through the confidence ellipses. In this example, the ellipse of categories 1, 2, and 3 for item 3 (labeled "`3.c1`," "`3.c2`," and "`3.c3`") and category 2 from item 5 (labeled "`5.c2`") do not fall on the identity line; these item categories also have a *p*-value less than 0.05 according to the Wald test without the Bonferroni correction. See #2.1_29 for a similar graph with dichotomous items fit to the simple Rasch model.

#3.1_25. Rasch fit measures of infit and outfit mean square (MSQ) values and the standardized t values are calculated using the "`itemfit()`" function. Infit and outfit MSQ values equal or close to one indicate a good fitting item. Large values of the "`infit t`" or "`outfit t`" (e.g., greater than 3 in absolute value) might be used for the diagnosis of potential misfit items. The "`Chisq`" statistic in the output is a χ^2 approach, which is based on the squared residuals, and it assumes the normality of the residuals. (One should be cautious of interpreting the result of the "`Chisq`" approach because the behavior of the residuals is not best approximated by a normal distribution.)

In this example, "`infit t`" and "`outfit t`" values are all greater than 3 in absolute value except item 3 ("`I3`"), while the "`Chisq`" *p*-values are 1 or near 1. These results are in disagreement, which could make researchers wonder about the degree of the consistency between the "`Chisq`" approach and the infit and outfit measures in the current "eRm" PCM application. These item fit calculation results appear to suggest further investigations of the item fit measures calculated for the PCM in the current "eRm." Users should exercise caution in using the "`itemfit()`" of the PCM application in the current "eRm." For more on the infit and outfit measures from the dichotomous Rasch model, see #2.1_30.

```
> itemfit(p.pcm)
Itemfit Statistics:
     Chisq  df p-value Outfit MSQ Infit MSQ Outfit t Infit t
I1 704.298 863   1.000     0.815    0.820   -3.46  -4.11
I2 715.217 863   1.000     0.828    0.836   -3.84  -3.93
I3 774.380 863   0.986     0.896    0.885   -2.08  -2.55
I4 713.493 863   1.000     0.826    0.852   -3.61  -3.50
I5 747.194 863   0.998     0.865    0.856   -2.41  -3.32
```

#3.1_26. The "eRm" package applies the Rasch infit and outfit measures to evaluate person fit as well, using the "`personfit()`" function. The response patterns following the fitted model should have the infit and outfit mean square ("`MSQ`") values of 1 or close to 1. The standardized version of the infit and outfit MSQ is the t index. A t value of greater than three in absolute value might be considered to flag potential cases of aberrant response patterns for further investigations.

```
> personfit(p.pcm)
Personfit Statistics:
     Chisq df p-value Outfit MSQ Infit MSQ Outfit t Infit t
P1  13.209  4   0.010     2.642    2.134    1.61   1.32
P2  14.210  4   0.007     2.842    2.430    2.49   2.14
...
P900 4.929  4   0.295     0.986    0.949    0.19   0.12
```

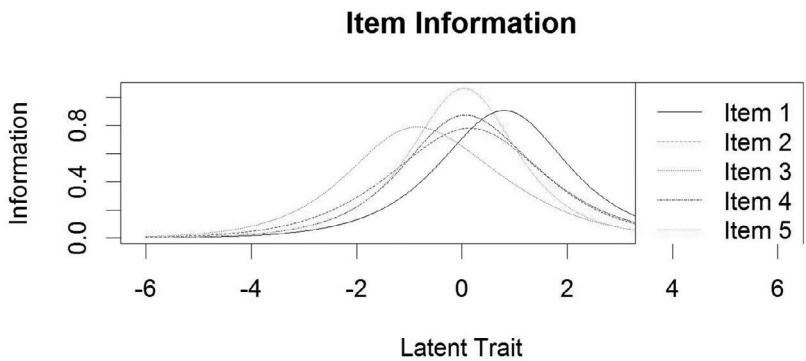
Readers may use any of the following commands, which call specific attributes of the person fit statistics to view each infit or outfit value in increasing order.

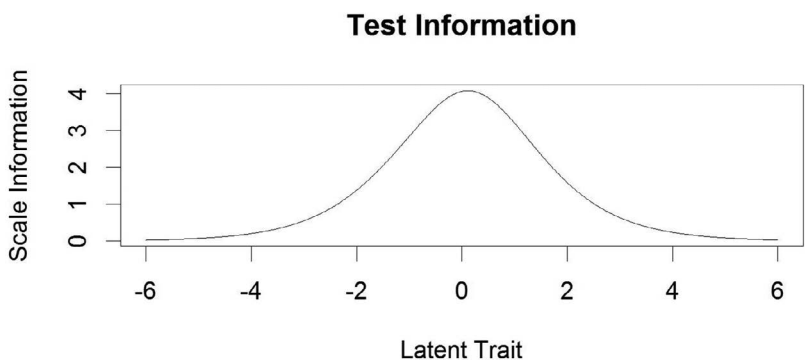
```
sort(personfit(p.pcm)$p.infitMSQ)
sort(personfit(p.pcm)$p.outfitMSQ)
sort(personfit(p.pcm)$p.infitZ)
sort(personfit(p.pcm)$p.outfitZ)
```

Information plot

```
plotINFO(mod.pcm) #3.1_27
plotINFO(mod.pcm, type="item") #3.1_28
```

#3.1_27. The item and the test information curves from the PCM calibration are provided using "`plotINFO()`." Recall that the simple Rasch model item information curves had the same shape, and the only difference across different item information curves was the location of the curve where the maximum item information is achieved. The item information in the PCM does not have the same shapes across items because of the different category threshold values across items.





#3.1_28. If only the test information plot is desired, users can specify "`item`" in the "`type=`" option when using "`plotINFO()`."

PCM calibration with the "mirt" package

While "eRm" estimates the PCM with the CML estimation, the "mirt" package provides the MML estimation for the PCM. Again, by default, the assumed population distribution in "mirt" is a normal distribution, with its mean and standard deviation equal to 0 and 1, respectively, for model identification purposes.

```
install.packages("mirt") #3.1_29
library(mirt) #3.1_30
setwd("c:/mystudy/RIRT/") #3.1_31
ddd<-read.fwf("c3_pcm2.dat", width=c(10, rep(1,10)))
  #3.1_32
ddd<-ddd[,-1] #3.1_33
names(ddd)<-paste("It", 1:10, sep="") #3.1_34
dim(ddd) #3.1_35
summary(ddd) #3.1_36
apply(ddd, 2, table) #3.1_37
```

#3.1_29. The "mirt" package is installed onto the computer using "`install.packages()`." This only needs to be done once on a computer system. See #2.1_1.

#3.1_30. The "mirt" package is loaded onto the current R session using the "`library()`" function. See #2.1_2.

#3.1_31. The working directory where data and R output files are stored is established using "`setwd()`." For the PCM using the "mirt" package, the "`c3_pcm2.dat`" should be saved into the folder established in the working directory. The data contain item responses to ten polytomous items, with responses of 0, 1, 2, and 3 for each item. See #2.1_3.

#3.1_32. The data file "`c3_pcm2.dat`" is read into R using "`read.fwf()`" and stored as an R object named as "`ddd`." The structure of this is different than the "`c3_pcm.dat`" used for the PCM application in the "eRm" package. The first ten columns are for person ID, and each of the remaining columns represents an item response in the data file. Reading this fixed format data file is done by using the "`read.fwf()`." See #2.2_1 for details.

#3.1_33. The first variable read from the data file is the person ID. Because it is not necessary to have the person ID in this application, it is removed from "`ddd`" and only the remaining item responses are stored in the same name "`ddd`." This is similar to #2.2_3.

#3.1_34. Replacing the variable or item names with "`It"1, "It2", ..., "It10`." See #2.1_8.

#3.1_35. The dimensions of the R object "`ddd`" are produced using the "dim()" function. The output shows two values: 750 rows as the first value,

corresponding to the number of test-takers, and ten columns as the second value, corresponding to each item.

#3.1_36. The "`summary()`" command shows the descriptive statistics of the minimum and maximum, the three quartiles, and the mean of each column of item response data. If only the mean or the standard deviation is desired, then the commands "`apply(ddd, 2, mean)`" and "`apply(ddd, 2, sd)`" can be used. See details following #3.1_8.

#3.1_37. The "`apply()`" command produces the frequency table of each response option for all items by requesting a "`table`." See #3.1_9.

Item parameter estimation

Similar to the 3PLM and 4PLM calibration (see #2.4_8 and #2.5_7, respectively), an efficient way to estimate the PCM with "mirt" is to use user-defined syntax to specify the PCM. Without a model specification, the "mirt" package estimates the PCM using a more general model, the generalized partial credit model (GPCM; Muraki, 1992). The necessary constraint for the PCM is to fix all item slope parameters of the GPCM to 1.0 and free the variance of the population ability distribution as a model parameter to estimate.

```
spec<-'F = 1-10
START = (1-10, a1, 1.0)
FIXED = (1-10, a1)
FREE = (GROUP, COV_11)' #3.1_38
mod.pcm<-mirt(ddd, model=spec, itemtype="gpcm", SE=T)
  #3.1_39
mod.pcm #3.1_40
extract.mirt(mod.pcm, what="converged") #3.1_41
extract.mirt(mod.pcm, what="secondordertest") #3.1_42
coef(mod.pcm, simplify=T, IRTpars=T) #3.1_43
coef(mod.pcm, IRTpars=T, printSE=T) #3.1_44
coef(mod.pcm, IRTpars=T) #3.1_45
```

#3.1_38. Four lines of syntax enclosed in single quotes is used to define the PCM. First, "`F=1-10`" specifies a unidimensional model for items 1 through 10. The "`START`" and "`FIXED`" commands set and fix the item slope parameters as 1. Lastly, the "`FREE`" command is used to free the variance of the ability population distribution for estimation.

#3.1.39. The syntax created in the previous step is read inside the estimation command, "`mirt()`." For the "`itemtype`," GPCM is specified. Again, the standard error calculation is requested by "`SE=T.`"

#3.1_40. The convergence results and the values of log-likelihood, AIC, and BIC are printed by calling the object "`mod.pcm`."

#3.1_41 and #2.1_42. Separate extraction of the convergence check results are printed using the "`extract.mirt()`" command. "`TRUE`" is printed if no issue was detected by the default convergence criteria and the second-order test.

#3.1_43. The model parameter estimation results are shown using the "`coef()`" function. Because GPCM is the model employed for the estimation of the

PCM, the value of the common slope, which is "1," is shown for all items under "a." The last three numeric columns under "b1," "b2," and "b3" are the threshold parameter estimates, $\hat{b}_{ik}$. There are three threshold parameters reported for the four-category response data. Notice that "b1" for the first item is not 0.0, as it was in the "eRm" package estimation. This is because the "eRm" and the "mirt" packages use different model identification. Here, the mean of the population ability distribution is constrained to 0, and the variance is a free parameter, and its estimate is "0.889" which is shown under "$cov."

```
> coef(mod.pcm, simplify=T, IRTpars=T)
$'items'
      a      b1      b2      b3
It1   1  -1.001  -0.293   0.331
It2   1  -1.502  -0.786   1.753
It3   1  -1.524  -1.194  -0.427
...
It9   1  -1.263  -0.080  -0.293
It10  1  -0.870  -0.275   0.359

$means
F
0

$cov
      F
F 0.889
```

These item parameter estimates are in the threshold parameterization (b_{ik}, Equation 3.1_1). They can be transformed to the location plus deviation parameterization, i.e., the overall location or difficulty, plus the deviations (from the overall location), as shown in Equation 3.1_2. Recall, the overall location parameter (b_i) estimate is the average of the threshold estimates. The deviation parameter (d_{ik}) estimate is obtained by the threshold minus the overall location parameter estimate. For this data, the location or difficulty parameter estimate, which is equal to the average of the thresholds across categories for each item, can be obtained using the following commands.

```
thresh<-coef(mod.pcm, simplify=T, IRTpars=T)$'items'
overall.diff<-apply(thresh[,-1],1, mean)
overall.diff
```

The "overall.diff" object holds the overall location or difficulty parameter estimates (b_i) for each item. The deviation parameter estimates (d_{ik}) can be obtained using the following commands.

```
deviat<-data.frame(thresh[,-1]-overall.diff)
names(deviat)<-c("dev1", "dev2", "dev3")
cbind(overall.diff, deviat)
```

The "deviat" object is the value of the threshold parameter estimates ("thresh") minus the overall location ("overall.diff"). Then "data.frame()" was used to convert the result to a data frame. The labels of the columns ("b1," "b2," "b"3) in "deviat" are replaced by "dev1," "dev2," "dev3" to avoid confusion, because the "deviat" object originated from the "mirt()" output and keeps the original labels for columns. The last line utilizes the "cbind()" function, which combines the results of the overall location column and the deviation columns.

```
> cbind(overall.diff, deviat)
    overall.diff       dev1        dev2      dev3
It1  -0.32099102 -0.6795753  0.02796377 0.6516116
It2  -0.17838866 -1.3232784 -0.60797322 1.9312516
It3  -1.04870443 -0.4757522 -0.14553360 0.6212858
...
It9  -0.54506373 -0.7175275  0.46540659 0.2521209
It10 -0.26188861 -0.6081567 -0.01278368 0.6209404
```

In this application of polytomous data, let a response of 0 = *Strongly Disagree*, 1 = *Disagree*, 2 = *Agree*, and 3 = *Strongly Agree*. The lowest overall difficulty is found for item 3, "It3," whose estimates is −1.0487, meaning that even those with lower amounts of the latent trait tend to exhibit higher degrees of agreement. Item 4, "It4," has the highest overall difficulty estimate of 0.2220, meaning that even those with a moderate or higher amount of the latent trait tend to show lower degrees of agreement, i.e., disagreement.

#3.1_44. The "coef()" function with "IRTpars=T" and "printSE=T" prints the threshold parameter estimates (b_{ik}) and their standard errors.

```
> coef(mod.pcm, IRTpars=T, printSE=T)
$'It1'
      a      b1      b2     b3
par   1  -1.001  -0.293  0.331
SE   NA   0.173   0.111  0.107

$It2
     a      b1      b2     b3
par  1  -1.502  -0.786  1.753
SE  NA   0.198   0.101  0.122
...

$It10
     a      b1      b2     b3
par  1  -0.870  -0.275  0.359
SE  NA   0.171   0.112  0.108

$GroupPars
  MEAN_1  COV_11
par    0   0.889
SE    NA   0.065
```

#3.1_45. The "coef()" function with "IRTpars=T" prints the 95% CIs for the threshold estimates. For the "a" parameter, "NA" is printed because it is fixed as one. The variance estimate of the population ability distribution is "0.889" and its 95% CI is shown for the lower and the upper bounds under "$GroupPar."

```
> coef(mod.pcm, IRTpars=T)
$'It1'
          a      b1      b2     b3
par       1  -1.001  -0.293  0.331
CI_2.5   NA  -1.339  -0.511  0.121
CI_97.5  NA  -0.662  -0.075  0.541

$It2
          a      b1      b2     b3
par       1  -1.502  -0.786  1.753
CI_2.5   NA  -1.891  -0.984  1.514
CI_97.5  NA  -1.113  -0.589  1.992
...

$It10
          a      b1      b2     b3
par       1  -0.870  -0.275  0.359
CI_2.5   NA  -1.205  -0.494  0.147
CI_97.5  NA  -0.535  -0.055  0.571

$GroupPars
     MEAN_1  COV_11
par       0   0.889
CI_2.5   NA   0.762
CI_97.5  NA   1.017
```

Person latent trait, or ability score estimation

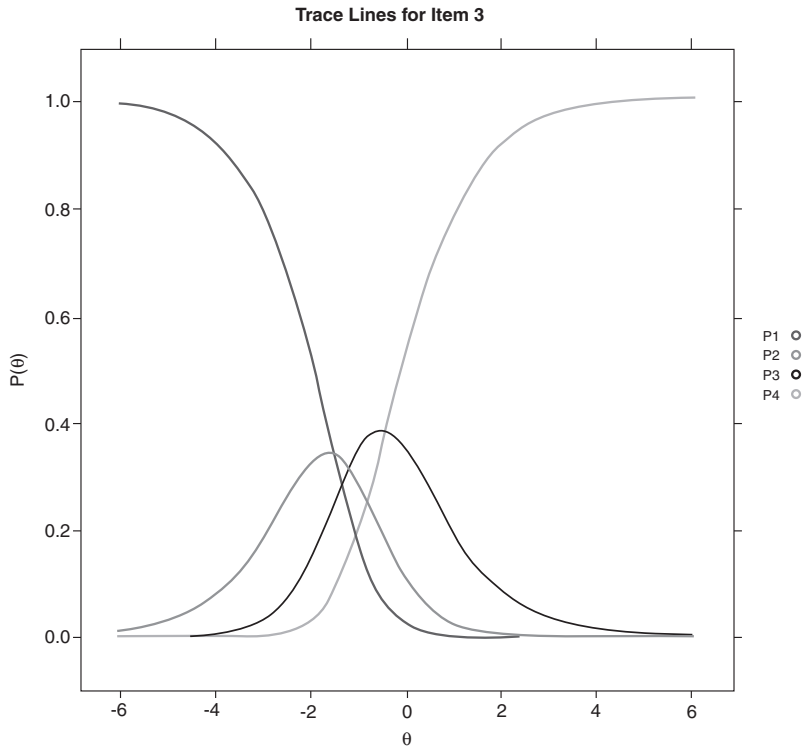
```
fscores(mod.pcm, method="EAP", full.scores=T, full.
scores.SE = T) #3.1_46
```

#2.1_46. Person latent trait or ability scores are calculated with the EAP estimator using the "fscores()" function and including the "method="EAP"" argument. "mirt" provides MAP and WLE estimator options in addition to EAP. The output column under "F1" is the person EAP score estimate, and its standard error is under the last column titled "SE_F1." See also #2.2_36.

ICC, TCC, and information plots

```
itemplot(mod.pcm, 3) #3.1_47
plot(mod.pcm, type = "trace") #3.1_48
plot(mod.pcm) #3.1_49
plot(mod.pcm, type="info") #3.1_50
itemplot(mod.pcm, 3, type="info") #3.1_51
```

#3.1_47. The CCC for item 3 is printed using "`itemplot()`" and specifying the item as the second argument.



#3.1_48. All CCCs for all items are shown in one output using "`plot()`" with "`type = "trace"`."

#3.1_49. The TCC is drawn using "`plot()`" without any additional argument.

#3.1_50. The test information curve is produced using "`plot()`" with "`type = "info"`."

#3.1_51. The item information curves are plotted using "`itemplot()`" with "`type = "info"`."

Model-data fit

```
M2(mod.pcm, type="M2*") #3.1_52
itemfit(mod.pcm) #3.1_53
itemfit(mod.pcm, empirical.plot=3) #3.1_54
personfit(mod.pcm, method="EAP") #3.1_55
residuals(mod.pcm, type="LD") #3.1_56
residuals(mod.pcm, type="Q3") #3.1_57
```

#3.1_52. The limited information overall model fit test, M2⋆, is requested using the "`M2()`" function. This M2⋆ test (Cai & Hansen, 2013) is an extension of

the dichotomous M2 (Maydeu-Olivares & Joe, 2006) for polytomous item response data. M2* is reduced to the original M2 when item responses are dichotomous. The M2* test has some restrictions regarding the number of categories and the number of items in its use. The number of items should be equal to or greater than 6 if M2* is applied to dichotomous data. For polytomous response data, the number of items should be equal to or greater than 12 if M2* is applied to the five-category item response data (Cai & Hansen, 2013).

To overcome these restrictions, another limited information test C2 has been proposed (Cai & Monro, 2014). C2 is equivalent to M2* for dichotomous response data, but more flexible in terms of the scope of the number of items and the number of categories. The C2 test can also be applied to mixed format response data (e.g., a test that has dichotomously scored responses and polytomously scored item responses). The default in "mirt," when using the "`M2()`" command, is the M2* test, so the "`M2()`" command can be used without specifying "`type="M2*"`." Computation-wise, as long as the degrees of freedom for M2* is positive, M2* is computed. Here, the option of the limited information test was specified to show explicitly that the default is M2* and the limited information type can be changed.

If C2 is desired, "`type="C2"`" is specified in the function. The results of the C2 test include the chi-square value, degrees of freedom, and the *p*-value, which are different from the results of M2*. These differences are due to the different ways employed in the two tests to collapse the lower order marginals. In addition to the M2* test results, descriptive overall model-fit measures such as RMSEA, are computed.

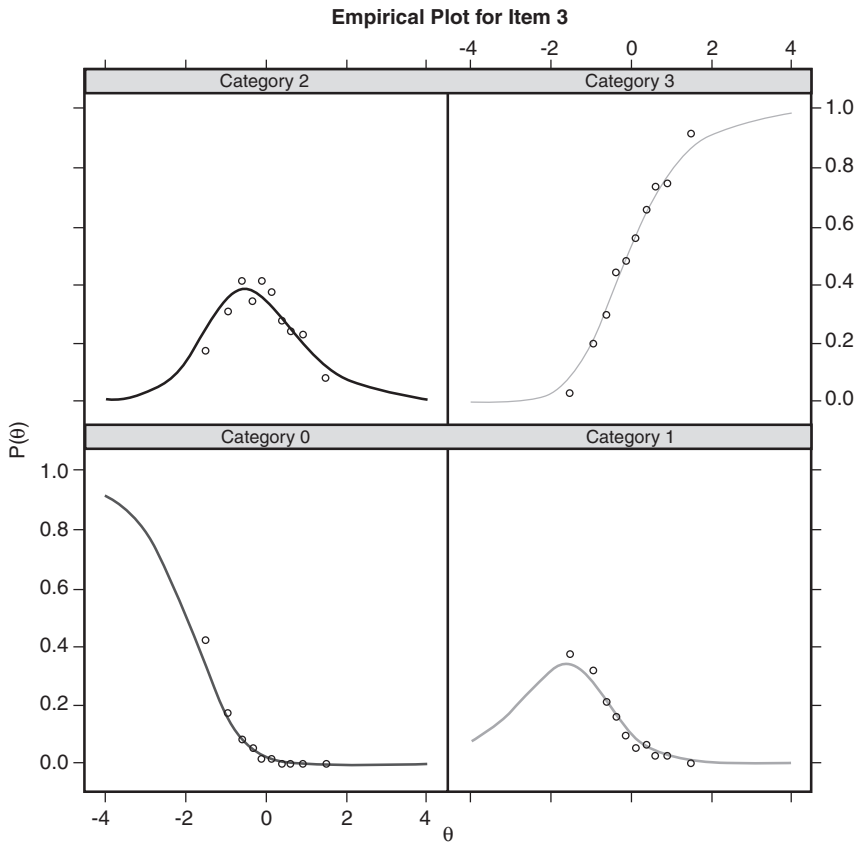
```
> M2(mod.pcm, type="M2*")

           M2 df         p      RMSEA RMSEA_5   RMSEA_95      SRMSR       TLI       CFI
stats 29.92818 24 0.1871434 0.01815994       0 0.03655109 0.02846619 0.9908137 0.9911812
```

The M2* results have a *p*-value of 0.187 and the fitted model is not rejected when using a level of significance of $\alpha = 0.05$.

#3.1_53. The item fit test, S-X2, is calculated for every item using the "`itemfit()`" function. The last column displays the *p*-value for every item test. With and without applying the family-wise Type I error rate of 0.05 using the Bonferroni correction, no item was flagged in this application.

```
> itemfit(mod.pcm)
    item    S_X2  df.S_X2  p.S_X2
1    It1  45.894       54   0.776
2    It2  60.380       52   0.199
...
9    It9  55.940       53   0.365
10  It10  36.160       54   0.970
```

#3.1_54. Empirical and model-based CCCs are printed for each category for the specified item using the "`itemfit()`" function and including the "`empirical.plot=3`" argument for item 3. This is a graphical item misfit check. Because an item in this application has four categories, four plots are produced, one for each category. The empirical CCC is represented by circles, and the model-based CCC is represented by a line or curve.



The empirical and the model-based CCCs of item 3 are close to each other in Figure 3.1_54.

#3.1_55. Because the PCM is a Rasch family model, the "mirt" package produces the standardized likelihood fit statistic l_z ("`Zh`") and the Rasch infit and outfit measures. When the fit is reasonable, the infit and outfit MSQ values ("`outfit`" and "`infit`") are one or close to one. A large departure from the value of one indicates a potential misfitting response pattern. The standardized versions of the infit and outfit values are the "`z.infit`" and "`z.outfit`." A large absolute value, e.g., greater than 3, may indicate potential

aberrant responses for a person. For the "Zh" standardized likelihood fit statistic, a smaller value, e.g., less than –3.0, indicates a more likely aberrant, or unusual, response pattern.

```
> personfit(mod.pcm, method="EAP")
       outfit      z.outfit      infit        z.infit            Zh
1   0.6775192 -0.674084763 0.7344235 -0.5597583216  1.0819229680
2   0.7015099 -0.811094173 0.7504293 -0.6557838967  0.5929651598
3   0.8400861 -0.285837981 0.9033658 -0.1351778883  0.2855441441
...
749 0.9814898  0.096444002 1.0861489  0.3431099540 -0.2096254552
750 0.7997592 -0.467537544 0.8226796 -0.4232339506 -0.0350766699
```

The following can be used to sort the output to better identify potential aberrant patterns. In the 3rd line, the value "16" specifies the sorting according to the "Zh" statistic column. To sort according to the "outfit," use the value 12, for "z.outfit," use the value 13, for "infit," use the 14, and for "z.infit," use 15.

```
p.fit <- cbind(rep(1:750),ddd, personfit(mod.pcm,
  method="EAP"))
head(p.fit)
p.fit[order(p.fit[,16]),]
```

#3.1_56. The "residuals()" function is used to check for local item dependence using various indices. LD-X2 is calculated for every item pair when "type="LD"." The output presents the LD-X2 value in the lower triangle of the resulting matrix. The elements in the upper triangle are the signed Cramer V statistic, which is based on the LD-X2 value and range from −1 to 1. Large absolute values in the LD-X2 and the Cramer V values indicate local dependence. See #2.1_57.

#3.1_47. Another local dependence index is the Q3 value. When "type="Q3"," the Q3 index is calculated for every item pair and presented in a matrix form. When the number of items is equal to or greater than 17, an absolute value of 0.2 was suggested as a cutoff for local dependence by Yen (1993). Another suggestion from de Ayala (2009) is $|Q3| \geq .2236$. See #2.1_58 for more information on Q3 and a set of code to print the results in an ordered manner for easily identifying potentially dependent item pairs.

3.2 Rating scale model (RSM) application

The rating scale model (RSM; Andrich, 1978) is another Rasch family IRT model that can handle polytomously scored item response data. The RSM requires that all items have the same number of options, or categories, and it assumes that adjacent threshold parameters are equally spaced, i.e., are equidistant, across all items. For

a four-category item scored 0, 1, 2, and 3, for instance, the distance between the first and the second threshold is the same across all items, the distance between the second and the third threshold is the same across all items, etc. Note, this is not a constraint that the distance between the first and second categories be equal to the distance between the second and third categories, etc., for a single item.

According to the threshold parameterization in Equation 3.1_1, the constraint may be defined as follows: $b_{i2} - b_{i1}$ is equal for all items; $b_{i3} - b_{i2}$ is equal for all items, etc. When using the location plus deviation parameterization in Equation 3.1_2, the overall difficulty (b_i) is allowed to vary while the deviation parameters are constrained to be same across all items, i.e., $d_{ik} = d_k$. Using the location plus deviation parameterization, the CRFs for the PCM and RSM for modeling the probability of responding in category $k \in \{0,1,2,\ldots,m_i\}$ to the ith item are expressed here. The RSM is a submodel of PCM, where the RSM requires the same number of options, or categories, for all items and equal spacing of adjacent thresholds, or $d_{ih} = d_h$.

$$\text{PCM}: P\left(X_{ikj} = k|\, \theta_j\right) = \frac{\exp\sum_{h=0}^{k}\left(\theta_j - b_i - d_{ih}\right)}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left(\theta - b_i - d_{ih}\right)}. \tag{3.2_1}$$

$$\text{RSM}: P\left(X_{ikj} = k|\, \theta_j\right) = \frac{\exp\sum_{h=0}^{k}\left(\theta_j - b_i - d_h\right)}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left(\theta - b_i - d_h\right)}. \tag{3.2_2}$$

All the terms in the previous equations are defined as before in Equations 3.1_1 and 3.1_2. Notice the exponential term of the PCM is the same as Equation 3.1_1, where

$$\theta_j - b_{ih} = \theta_j - (b_i + d_{ih}) = \theta - b_i - d_{ih}.$$

The RSM calibration with the "mirt" package is explained in this section. The CRF and CCC of the RSM in "mirt" does not exactly follow Equation 3.2_2. Rather, it estimates a common set of threshold parameters for each category across all items ("`b1`," "`b2`," and "`b3`") and an overall location parameter for each item ("`c`"). These are not equal to the parameterization shown in Equation 3.2_2, for which a conversion is provided in a later section.

RSM calibration with the "mirt" package

The "mirt" package and corresponding function uses the MML estimation for the IRT calibrations, where the population ability distribution is assumed to follow a normal distribution. In the estimation of the Rasch family model with the MML method, the variance of the population ability distribution is a free parameter.

```
install.packages("mirt") #3.2_1
library(mirt) #3.2_2
setwd("c:/mystudy/RIRT/") #3.2_3
ddd<-read.table("c3_rsm.dat") #3.2_4
dim(ddd) #3.2_5
```

#3.2_1. Install the "mirt" package using "`install.packages()`." This is only required if it has not been previously installed on the computer.

#3.2_2. Upload the "mirt" package onto the current R session using the "`library()`" function.

#3.2_3. Set up the working directory where the folder holding the data files is stored established using "`setwd()`." Readers should save the data file in the working directory.

#3.2_4. The data file "`c3_rsm.dat`" is read into R and stored in an R objected called "`ddd`." This data has responses to five items, each of which is scored with 0, 1, 2, or 3. The data have a space-delimited format that can be read by the "`read.table()`" command with ease. The default variable or item labels are "V1," "V2," . . ., "V5." This can be observed using the "`head()`" function.

#3.2_5. The "`dim()`" function reports that the number of rows or test-takers is 550, and the number of columns or items is five.

Item parameter estimation

```
mod.rsm<-mirt(ddd, model=1, itemtype="rsm", SE=T) #3.2_6
mod.rsm #3.2_7
coef(mod.rsm, simplify=T, IRTpars=T) #3.2_8
```

#3.2_6. The "`mirt()`" function is used to fit the RSM by specifying the "`itemtype=rsm`." Standard errors of the estimations are also calculated when "`SE=T`."

#3.2_7. Overall convergence results are printed. "`Converged within 1e-04 tolerance after 21 EM iterations`" and "`Second-order test: model is a possible local maximum`" indicate the estimation ran without any issues detected under the default convergence criteria. Similar to #2.1_42 and #2.1_43, typing "`extract.mirt(mod.rsm, what="converged")`" and "`extract.mirt(mod.rsm, what="secondordertest")`" also provide the convergence check results; output of "`TRUE`" for these indicate that the models converged without error.

#3.2_8. The estimated item thresholds and overall location are produced using the "`coef()`" function. The RSM parameterization in "mirt" does not exactly follow the CRF shown in Equation 3.2_2. It estimates a common set of threshold parameters for each category across all items ("`b1`," "`b2`," and "`b3`") and an overall location parameter for each item ("`c`").

```
> coef(mod.rsm, simplify=T, IRTpars=T)
$'items'
   a1     b1     b2     b3       c
V1  1  -0.02  0.496  1.268   0.000
V2  1  -0.02  0.496  1.268   1.497
V3  1  -0.02  0.496  1.268  -0.493
V4  1  -0.02  0.496  1.268  -0.278
V5  1  -0.02  0.496  1.268   0.963

$means
F1
0

$cov
      F1
F1 0.989
```

The thresholds are constrained to be the same across all items, while allowing the overall locations of items to vary. The overall location parameter ("c") in the RSM in "mirt" represents an overall item easiness, not the overall item difficulty. And, the overall item easiness or location parameter ("c") is not defined as the average of the category threshold parameters as was in the PCM.

For the model identification, the first item location or easiness parameter is set to 0 and the population ability mean is constrained to 0. As usual, the variance of the population ability distribution is freely estimated, which is 0.989 in this application.

The current way to resolve the model identification using the first item location parameter is not always the most useful or appropriate because it could sometimes provide incorrect interpretations of item difficulty parameters in reference to the average population ability. For instance, in this application, suppose that the population ability mean is truly 0, while the true overall item difficulty of the first item is 0.5 (more than average or medium difficulty). The first item overall easiness is set to 0 and equal to the average of the examinees' abilities in the population. Accordingly, treating the first item as a medium difficulty relative to the examinee population is not correct.

For the relationship between the RSM parameters produced by the "mirt" package with the traditional item parameters presented in the PCM, the transformation is presented here. Converting the RSM results by "mirt" to the threshold parameterization (similar to Equation 3.1_1 for the PCM with the constraint of equidistant thresholds) can be done in the following way.

```
mrp<-coef(mod.rsm, simplify=T, IRTpars=T)$'items'[,-1]
  #3.2_9
rsm.thresh< - mrp[,4]+mrp[,-4] #3.2_10
rsm.thresh #3.2_11
```

#3.2_9. The "`coef()`" function is used to extract the item thresholds, "`b1`," "`b2`," and "`b3`," and the item location, "`c`," and exclude the item "`a`" parameter and population ability distribution parameters. These are saved as the object "`mrp`."

#3.2_10 and #.3.2_11. The estimated "mirt" RSM parameters are transformed to the threshold parameterization, presented in Equation 3.1_1 for the PCM, with the constraint of equidistant thresholds for the RSM. The full expression of the command sums the negative of the overall item easiness (i.e., the overall item difficulty, "`-c`": "`-mrp[,4]`") and the values of "`b1`," "`b2`," and "`b3`" ("`mrp[,-4]`"), which yield the conventional threshold values (b_{ik} in Equation 3.1_1).

```
> rsm.thresh
            b1          b2         b3
V1 -0.02041868  0.4962116  1.2682571
V2 -1.51790861 -1.0012783 -0.2292328
V3  0.47307824  0.9897086  1.7617541
V4  0.25770108  0.7743314  1.5463769
V5 -0.98303509 -0.4664048  0.3056407
```

Because the RSM imposes equal spacing of the adjacent thresholds, the difference between the second and the first threshold is constant across all items $(\hat{b}_{i(k=2)} - \hat{b}_{i(k=1)} = 0.517)$, and the difference between the third and the second threshold is constant across all items:

$$(\hat{b}_{i(k=3)} - \hat{b}_{i(k=2)} = 0.772).$$

To convert these thresholds into the overall difficulty (b_i) plus deviation (d_k) parameters as defined in Equation 3.1_2 (and 3.2_2), the steps shown previously in #3.1_43 for the PCM could be applied. The steps are provided here again for the RSM.

```
overall.diff<-apply(rsm.thresh, 1, mean)
deviat<-data.frame(rsm.thresh-overall.diff)
names(deviat)<-c("dev1", "dev2", "dev3")
cbind(overall.diff, deviat) #3.2_12
```

#3.2_12. The overall item location or difficulty parameters (b_i), "`overall.diff,`" are calculated as the average of the item thresholds, "`rsm.thresh.`" The category deviations, d_k, are calculated by subtracting the overall location or difficulty from each threshold, i.e., "`rs rsm.thresh-overall.diff`." The resulting category deviations are renamed, "`dev1`," "`dev2`," and "`dev3`." These are equivalent to the parameters in Equation 3.2_2.

```
> cbind(overall.diff, deviat)
  overall.diff        dev1        dev2       dev3
V1    0.5813500  -0.6017687  -0.0851384  0.6869071
V2   -0.9161399  -0.6017687  -0.0851384  0.6869071
V3    1.0748470  -0.6017687  -0.0851384  0.6869071
V4    0.8594698  -0.6017687  -0.0851384  0.6869071
V5   -0.3812664  -0.6017687  -0.0851384  0.6869071
```

The standard errors and the 95% CIs for the estimated parameters from the RSM in the "mirt" package are provided by "`coef(mod.rsm, IRTpars=T, printSE=T)`" and "`coef(mod.rsm, IRTpars=T)`." However, they are not available for the transformed threshold or location plus deviation parameterizations such as Equation 3.2_2.

Person latent trait, or ability score estimation

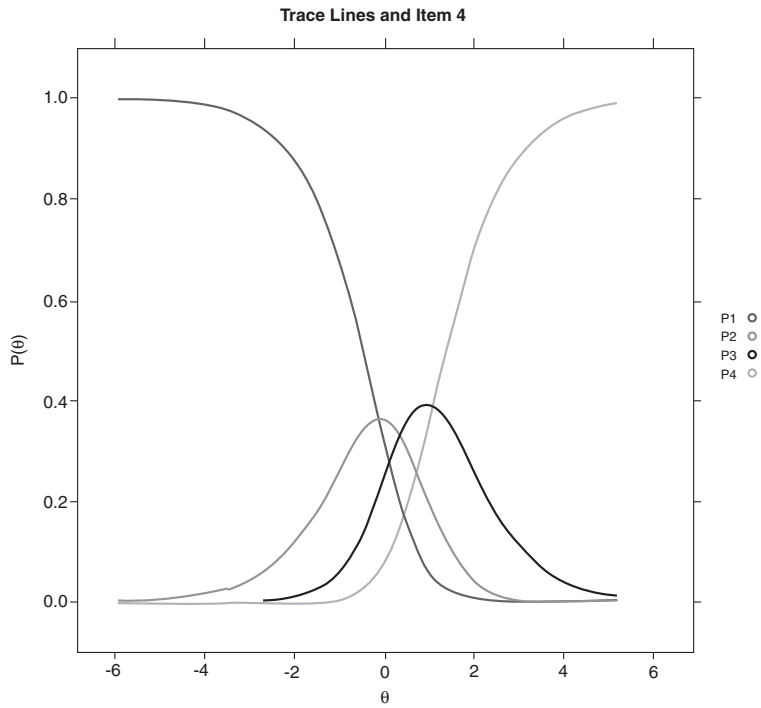
```
fscores(mod.rsm, method="EAP", full.scores=T, full.
  scores.SE = T) #3.2_13
```

#3.2_13. The EAP estimates for person ability scores are obtained using the "`fscores()`" function. The values under "`F1`" in the output represent EAP estimates, and the values under "`SE_F1`" are the standard deviations of the posterior distribution, which are used as standard errors. See #3.1_46 and #2.1_47 for details.

CCC, TCC, and information plots

```
itemplot(mod.rsm, 4) #3.2_14
plot(mod.rsm, type = "trace") #3.2_15
plot(mod.rsm) #3.2_16
plot(mod.rsm, type="info") #3.2_17
```

#3.2_14. CCCs are drawn for item 4 using the "`itemplot()`" function, with the second argument in the command specifying the item number. The values on the θ metric at the intersection of the adjacent category curves are the threshold estimates shown from "`rsm.thresh`" in #3.2_11. They correspond to 0.258, 0.774, and 1.546 for item 4. The RSM imposes a restriction on the CCC such that these CCCs are different only in their overall locations without changing shapes or slopes.



#3.2_15. The CCCs are drawn for all items using the "`plot()`" function with "`type="trace"`."

#3.2_16. The TCC is constructed and shown using the "`plot()`" function without any additional arguments.

#3.2_17 and #3.2_18. The test information curve is displayed using the "`itemplot()`" function with "`type="info"`." For the construction of item information curve, "`itmplot()`" function with "`type="info"`" is used. For example, "`itemplot(mod.rsm, 1, type="info")`" produces an item information curve plot for item 1.

Model-data fit and model comparison

```
M2(mod.rsm, type="C2") #3.2_18
itemfit(mod.rsm) #3.2_19
itemfit(mod.rsm, empirical.plot=4) #3.2_20
personfit(mod.rsm, method="EAP") #3.2_21
residuals(mod.rsm, type="LD") #3.2_22
residuals(mod.rsm, type="Q3") #3.2_23
```

#3.2_18. The limited information C2 test is reported using the "`M2()`" function with "`type="C2"`." Specifying "`type=M2*`," which is the default for the type command, results in "`Error: M2() statistic cannot be calculated due to too few degrees of freedom.`" M2* is

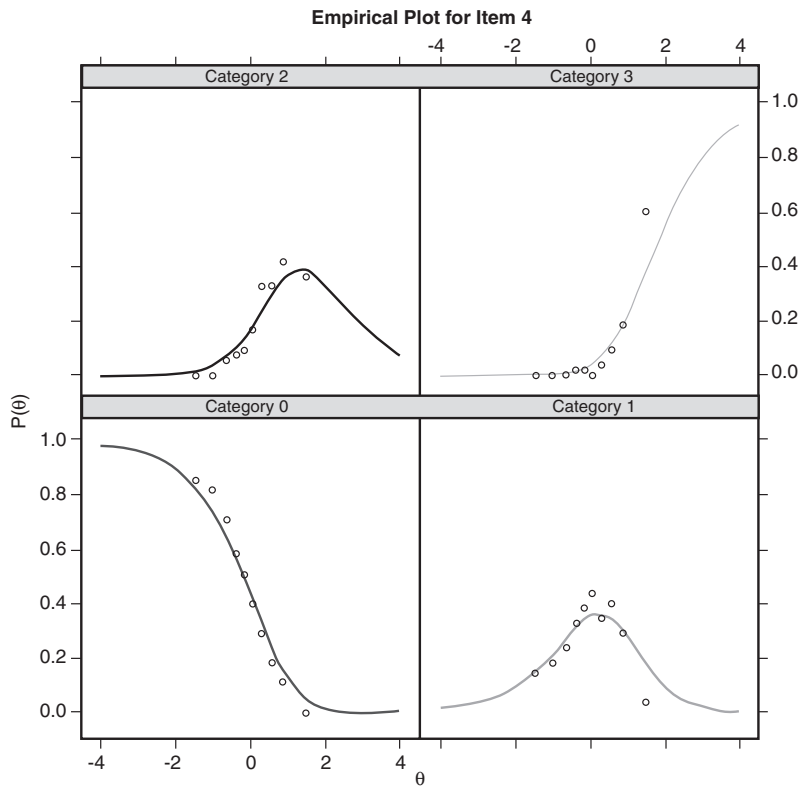
developed for polytomous item response data but doesn't hold up in some cases. For a short test length with a large number of categories, the M2⋆ cannot be calculated because its degrees of freedom is not possible to calculate. The C2 statistic was proposed to relax such restrictions in M2⋆, which makes it suitable for mixed format data sets as well.

```
> M2(mod.rsm, type="C2")
             M2 df p RMSEA RMSEA_5 RMSEA_95      TLI CFI
stats 0.5868575 17 1     0       0        0 1.832146   1
```

The results from the C2 test in this application are shown. The *p*-value of the C2 test is virtually 1, and we fail to reject the fitted model, RSM, with this data set. See #3.1_52 for more descriptions of the M2⋆ and the C2 test.

#3.2_19. The "`itemfit()`" function provides item level fit tests using the S-X2 statistic. The last column under "`p.S_X2`" shows the *p*-values of the test for each item. When applying the Bonferroni correction under the family-wise Type I error rate of 0.05, all items but the first one were not statistically flagged.

#3.2_20. The comparison of empirical CCC and model-based CCC is shown for each category for item 4 using "`itemfit(mod.rsm, empirical.plot=4)`." This is a graphical check for model-data fit at the item level.



According to the plots for each category curve, the empirical dots and the model-based curve align for a majority of the θ scale.

#3.2_21. The Rasch fit indices for the person response pattern evaluation are provided using the "`personfit()`" function. The results include the infit and outfit MSQ values ("`infit`" and "`outfit`"), their standardized values ("`z.infit`" and "`z.outfit`"), and the likelihood-based person fit measure l_z statistic ("`Zh`"). When data fits the model, the infit and outfit MSQs are close to one. Large absolute values, e.g., greater than three, of "z.infit" and "z.outfit" may indicate potential aberrant response patterns. A very small value of "`Zh`" indicate a potential response pattern (e.g., less than negative three). The following code can be used to produce output in increasing order of the "`Zh`" statistic. Users can modify the column reference in the third line (here "`11`") to order the data based on another statistic by referencing the statistic's column.

```
p.fit <- cbind(rep(1:550),ddd, personfit(mod.rsm,
method="EAP"))
head(p.fit)
p.fit[order(p.fit[,11]),]
```

#3.2_22. "`residuals()`" produces the LD-X2 statistic (elements in the lower diagonal) for the examination of local dependence for every pair of items when "`type="LD"`." The upper diagonal elements are the Cramer's V calculated using the LD-X2 values. A large absolute value of LD-X2 or Cramer's V is a strong indicator of the violation of the local independence. See #2.1_57.

#3.2_23. The residual correlation matrix, Q3, for the examination of local dependence, is calculated for every pair of items and shown in a matrix when "`type="Q3""` in "`residuals().`" The larger the absolute value of Q3, the stronger the indication of local dependence. See #2.1_58 for more details about Q3.

```
spec<-'F = 1-5
START = (1-5, a1, 1.0)
FIXED = (1-5, a1)
FREE = (GROUP, COV_11)' #3.2_24
mod.pcm<-mirt(ddd, model=spec, itemtype="gpcm", SE=T)
  #3.2_25
anova(mod.rsm, mod.pcm) #3.2_26
```

Because the RSM is nested within the PCM, a model comparison using the likelihood ratio test can be conducted. Because a different data set was used in the PCM and RSM sections, the PCM must be fit to this "`c3_rsm.dat`" data.

#3.2_24 and #3.2_25. The PCM is defined (similar to #3.1_38) and fit to the same data set (similar to #3.1_39).

#3.2_26 The output for the "`anova()`" test for nested models shows the model comparison test results with descriptive model fit indices such as AIC, BIC, AICc, and SABIC. Smaller values of the fit indices indicate a better model-data fit.

```
> anova(mod.rsm, mod.pcm)
Model 1: mirt(data = ddd, model = 1, itemtype = "rsm", SE = T)
Model 2: mirt(data = ddd, model = spec, itemtype = "gpcm", SE = T)
 AIC AICc SABIC HQ BIC logLik X2 df p
1 6457.337 6457.603 6466.420 6470.810 6491.816 -3220.668   NaN NaN   NaN
2 6468.601 6469.622 6486.769 6495.549 6537.560 -3218.301 4.735   8 0.785
```

The *p*-value of the likelihood ratio test is 0.785, failing to reject RSM. This support for the RSM is consistent with the model fit indices, AIC, AICc, BIC, and SABIC. For more on model comparisons of nested models, see #2.2_20 as well.

3.3 Generalized partial credit model (GPCM) application

The generalized partial credit model (GPCM; Muraki, 1992) is an extension of PCM (and therefore an extension of the RSM), where the items' slope or discrimination parameters are freely estimated. The threshold parameterization of the CRF and CCC of the GPCM to determine the probability of a response in category $k \in \{0,1,2,\ldots,m_i\}$ of the ith item is:

$$P\left(X_{ikj} = k|\,\theta_j\right) = \frac{\exp\sum_{h=0}^{k}\left[a_i\left(\theta_j - b_{ih}\right)\right]}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left[a_i\left(\theta - b_{ih}\right)\right]}, \tag{3.3_1}$$

where X_{ikj} is the jth person's response in category k to the ith item, a_i is the ith item slope or discrimination parameter, θ_j is the jth person's latent trait or ability score, and b_{ih} is the threshold parameter for the kth category of the ith item when $h = k$. Both b_{ih} and $a_i(\theta - b_{ih})$ for $k = 0$ are defined as 0, i.e., $b_{i0} = 0$ and $a_i(\theta - b_{i0}) = 0$. In the MML estimation of the GPCM, $\theta \sim N(0,1)$ is typically used for the model identification. The threshold parameter, b_{ih}, when $h = k$, is a point on the θ scale where the CCC for the response in category k, $P_{ik}(\theta)$, intersects with the CCC for the response in category $k - 1$, $P_{i(k-1)}(\theta)$ (similar to the PCM). If the parameterization of the location plus deviation form is used,

$$P\left(X_{ikj} = k|\,\theta_j\right) = \frac{\exp\sum_{h=0}^{k}\left[a_i\left(\theta_j - b_i + d_{ih}\right)\right]}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left[a_i\left(\theta - b_i + d_{ih}\right)\right]}, \tag{3.3_2}$$

where b_i is the overall item location, or difficulty, and d_{ih} is the deviation parameter defined as $b_i - b_{ih}$. For the model identification of the model in Equation 3.3_2, when MML is used for estimation, $\theta \sim N(0,1)$ and $\sum_{k=0}^{m_i} d_{ik} = 0$ are typically used. Recall that the deviation parameter, d_{ih}, in PCM (and RSM) is equal to $b_{ih} - b_i$, i.e., the threshold minus the overall difficulty. The deviation parameter, d_{ik}, for the kth category in GPCM is the negative of the PCM (and RSM) deviation parameter definition. This is the way the author of GPCM (Muraki, 1992) defined the deviation parameter. This means that,

in the PCM (and RSM), d_{ih} is the relative position of the kth category threshold to the overall location, or difficulty, parameter, while d_{ih} in the GPCM is the relative position of the overall location, or difficulty, in reference to the kth category threshold.

For example, if $d_{i(k=1)} = -1.0$ in the PCM, the threshold $b_{i(k=1)}$ is far from the overall difficulty by a value of 1.0 in the negative direction, while a $d_{i(k=1)} = 1$ in the GPCM means that the overall difficulty is far apart from the first category threshold by 1.0 in the positive direction.

Another expression of the CCC sometimes used in the literature and some IRT software, including the "mirt" package in R is as follows.

$$P\left(X_{ikj} = k|\,\theta_j\right) = \frac{\exp\left[a_i\left[k(\theta_i - b_i) + \sum_{h=0}^{k} d_{ih}\right]\right]}{\sum_{c=0}^{m_i}\exp\left[a_i\left[c(\theta_i - b_i) + \sum_{h=0}^{c} d_{ih}\right]\right]}. \qquad (3.3\_3)$$

$k(\theta - b_i)$ and d_{ih} are defined as 0 when $k = 0$. All the model parameters are defined the same as before.

The GPCM calibration is shown using the "ltm" and "mirt" packages. Both use the MML estimation, which assumes a normal population ability distribution with a mean and standard deviation of 0 and 1, respectively, for the model identification.

GPCM calibration with the "ltm" package

The "ltm" package uses the Gauss-Hermit quadrature method (by default) for the numerical integration used in the process of item parameter estimation.

```
install.packages("ltm") #3.3_1
library(ltm) #3.3_2
setwd("c:/mystudy/RIRT") #3.3_3
ddd<-read.table("c3_gpcm.dat") #3.3_4
dim(ddd) #3.3_5
head(ddd) #3.3_6
apply(ddd, 2, table) #3.3_7
```

#3.3_1. The "ltm" package is installed using "`install.packages()`"; this is only required if the "ltm" is not already installed on the computer.

#3.3_2. The "ltm" package must be loaded into the current R session using the "`library()`" function.

#3.3._3. The working directory and folder in which data files are stored is set using "`setwd().`" Readers should store the data file in their working directory folder.

#3.3_4. The raw data is read into R using the "`read.table()`" function and saved as "`ddd`." The original data has space-delimited format, which can be easily read by the "`read.table()`" command. The default variable, or item, names are "V1," "V2," . . . , "V12" for the 12 item columns.

#3.3_5. The "`dim()`" function shows the dimensions of the data set. The number of rows, or test-takers, is 750, and the number of columns, or items, is 12.

#3.3_6. The "`head()`" function shows the first six rows of the data.

#3.3_7. "`apply()`" is used to check the frequencies of responses across the category options for the items by requesting a "`table`" by columns. The item responses are 0, 1, 2, and 3. See #3.1_9 for more on null categories and recoding of the item responses if necessary. One may also use "`summary(ddd)`," which shows the minimum, the maximum, median, mean, etc., for each item.

Item parameter estimation

```
mod.gpcm <- gpcm(ddd, IRT.param=T) #3.3_8
mod.gpcm$conv #3.3_9
mod.gpcm #3.3_10
coef(mod.gpcm) #3.3_11
summary(mod.gpcm) #3.3_12
```

#3.3_8. The GPCM is estimated using the "`gpcm()`" function. To make sure the model parameterization follows the conventional form as in Equation 3.3_1, "`IRT.param=T`" was used explicitly, which is the default. If "`IRT.param=F`," then the slope and intercept parameterization (i.e., $a_i\theta + d_{ik}$) is used.

#3.3_9. Observing "`mod.gpcm$conv`" checks for the model convergence. "`0`" means no particular issue was detected under the default convergence criteria.

#3.3_10 and #3.3_11. "`mod.gpcm`" and "`coef(mod.gpcm)`" report the point estimates of the item parameters (slope or discrimination, a_i and threshold, b_{ik}). #3.3_10 includes the final log-likelihood value once the convergence is achieved, but #3.3_11 does not.

```
> mod.gpcm
Call:
gpcm(data = ddd, IRT.param = T)
Coefficients:
    Catgr.1  Catgr.2  Catgr.3  Dscrmn
V1   -0.207    0.674    0.726   1.823
V2   -1.365   -0.795   -0.761   0.811
...
V11  -0.007    0.534    1.695   1.398
V12   0.200    0.823    1.478   1.525

Log.Lik: -9150.037
```

Values under "`Catgr.1`," "`Catgr.2`," and "`Catgr.3`" are the threshold estimates, $b_{i(k=1)}$, $b_{i(k=2)}$, and $b_{i(k=3)}$, respectively. "`Dscrmn`" is the item slope or discrimination a_i.

#3.3_12. The "`summary()`" function reports the same point estimates of the item parameters and their standard errors. Additionally, the log-likelihood value and information criteria, i.e., AIC and BIC, are printed for the model.

```
> summary(mod.gpcm)
Call:
gpcm(data = ddd, IRT.param = T)

Model Summary:
  log.Lik      AIC      BIC
-9150.037 18396.07 18617.84

Coefficients:
$'V1'
          value  std.err  z.value
Catgr.1  -0.207    0.073   -2.833
Catgr.2   0.674    0.089    7.612
Catgr.3   0.726    0.096    7.565
Dscrmn    1.823    0.150   12.185

$V2
          value  std.err  z.value
Catgr.1  -1.365    0.187   -7.306
Catgr.2  -0.795    0.157   -5.064
Catgr.3  -0.761    0.151   -5.035
Dscrmn    0.811    0.070   11.498
...
$V12
          value  std.err  z.value
Catgr.1   0.200    0.086    2.320
Catgr.2   0.823    0.098    8.403
Catgr.3   1.478    0.125   11.818
Dscrmn    1.525    0.127   11.970

Integration:
method: Gauss-Hermite
quadrature points: 21

Optimization:
Convergence: 0
max(|grad|): 0.018
optimizer: nlminb
```

Person latent trait, or ability score estimation

```
factor.scores(mod.gpcm, method="EAP") #3.3_13
```

#3.3_13. EAP ability estimates are calculated for each observed response pattern using the "`factor.scores()`" function. Note that the original data have 0, 1, 2, and 3 item scores. In this output, those are represented as 1, 2, 3, and 4.

```
> factor.scores(mod.gpcm, method="EAP")
Call:
gpcm(data = ddd, IRT.param = T)
Scoring Method: Expected A Posteriori
Factor-Scores for observed response patterns:
   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 Obs   Exp     z1 se.z1
1   1  1  1  1  1  1  1  1  1   1   1   1   1 1.571 -2.718 0.498
2   1  1  1  1  1  1  1  1  1   2   1   1   2 0.389 -2.498 0.453
...
716 4  4  4  4  4  4  4  4  4   4   4   3   3 3.051  1.894 0.459
717 4  4  4  4  4  4  4  4  4   4   4   4  10 7.531  2.264 0.531
```

"`Obs`" represents the observed number of test-takers for the given response pattern, and "`Exp`" represents the model-predicted or expected number of test-takers for that item response pattern. The column labeled "`z1`" is the EAP latent trait or ability score estimate, and the column labeled "`se.z1`" is the standard error. The results are ordered from the smallest estimated latent score to the largest. The observed response patterns are displayed under "`V1`" through "`V12`." The last row indicates that there are 717 unique response patterns in the data.

CCC and information plots

All curves from the GPCM in the "ltm" package are created using the "`plot()`" function.

```
plot(mod.gpcm) #3.3_14
plot(mod.gpcm, type="IIC", item=0) #3.3_15
plot(mod.gpcm, type="IIC") #3.3_16
plot(mod.gpcm, type="IIC", item=c(2,3,5), legend=T) #3.3_17
```

#3.3_14. CCCs are constructed for each item using the "`plot()`" function with no additional arguments. When this command is executed, a separate graphical window appears. In the top of the window, instructions are given to "Click or hit ENTER for next page." Users must click within the window or pressing "Enter" on the keyboard to display the CCCs, one by one. If a single item's CCC is desired, the following can be used, where the number specified in "`item=`" is the requested item, i.e., 2 for this example.

```
plot(mod.gpcm, type="ICC", item=2, legend=T)
```

#3.3_15. The test information curve is drawn when "`type="IIC", item=0`" is included.

#3.3_16. Omitting "`item=0`" in the prior command, only including "`type="IIC"`" displays all item information curves in one plot.

#3.3_17. This command with "`type="IIC", item=c(2,3,5)`" displays the item information curves for selected items, which are items 2, 3, and 5 in one plot. Rather than the curves be labeled directly with the item, "`legend=T`" prints the legend on the graph.

Model-data fit and model comparison

"ltm" does not provide item-level fit or person fit measures for the polytomous response models, but it provides an overall model-data fit procedure, which is a parametric bootstrap method using a Pearson chi-square test when the GPCM is fit.

```
GoF.gpcm(mod.gpcm, B = 99) #3.3_18
```

"`B`" is the number of bootstrap samples requested. The last row in the output from this command shows the *p*-value based on the bootstrap; for this data "`p-value: 0.52`" thus, failing to rejected fitted model under $\alpha = 0.05$. Note, this command may take a longer amount of time to complete than many other executions.

A relative model evaluation using the likelihood ratio test may be conducted by comparing the fit of the GPCM with a reduced model, or a nested model fir with the same data. When the dichotomous models were presented, it was mentioned that the 1PLM was mathematically equivalent to the Rasch model in terms of the model-data fit. In the same vein, when GPCM is fitted with the equality constraint imposed on item slopes, then the common-slope GPCM is equivalent to PCM. Thus, the common-slope GPCM, or PCM, is nested within the unconstrained GPCM, so the comparison of these can be done through a log-likelihood ratio test.

```
mod.gpcm.ca<-gpcm(ddd, constraint="1PL") #3.3_19
mod.gpcm.ca$conv #3.3_20
mod.gpcm.ca #3.3_21
anova(mod.gpcm.ca, mod.gpcm) #3.3_22
```

#3.3_19. The common-slope GPCM is fitted. In the constraint option, "`1PL`" is specified. This restricts all item slope, or discrimination parameters, a_i, to be equal. In addition to the "`1PL`" constraint option, there is a "`rasch`" constraint, which fixes the item slopes to equal 1.0. However, "ltm" uses MML with $\theta \tilde{} N(0,1)$ and it does not appear that the population variance is treated as a free parameter in the case of the "`rasch`" option. A model with the option "`rasch`" is equivalent to fitting a more constrained model than PCM.

#3.3_20. A quick convergence check of the constrained GPCM, i.e., the PCM, is made. A result of "0" means no issues were detected in the estimation process.

#3.3_21. Model parameter estimates are shown. The common slope or discrimination is estimated to be 1.391.

#3.3_22. The "anova()" function is used to conduct the likelihood ratio test to comparing the common-slope GPCM with the unconstrained GPCM. The *p*-value is less than 0.001; thus, rejecting the common-slope GPCM. The output includes the AIC, BIC, and log-likelihood values. Both the AIC and BIC prefer the GPCM as well.

```
> anova(mod.gpcm.ca, mod.gpcm)
Likelihood Ratio Table
                 AIC      BIC  log.Lik    LRT df  p. value
mod.gpcm.ca 18548.26 18719.21 -9237.13        37
mod.gpcm    18396.07 18617.84 -9150.04 174.19 48    <0.001
```

GPCM calibration with the "mirt" package

```
install.packages("mirt") #3.3_23
library(mirt) #3.3_24
setwd("c:/mystudy/RIRT/") #3.3_25
ddd<-read.fwf("c3_gpcm2.dat", width=c(rep(1,15))) #3.3_26
dim(ddd) #3.3_27
summary(ddd) #3.3_28
apply(ddd, 2, table) #3.3_29
```

#3.3_23. The "mirt" package is installed using "install.packages()" if it has not been previously installed on the computer. See #2.1_1.

#3.3_24. "mirt" is loaded onto the current R session using the "library()" function. See #2.1_2.

#3.3_25. The working directory where data files and R output files are exported is specified using "setwd()." The "c3_gpcm2.dat" file should be stored in the specified working directory folder. See #2.1_3.

#3.3_26. "read.fwf()" is used to read the fixed format data. Columns 1 through 15 are item responses. The data are stored in the object "ddd," and it has the following default variable or item names: "V1," "V2," . . . , "V15." Each item has five categories scored 0, 1, 2, 3, and 4.

#3.3_27. The output of the "dim()" function shows the number of rows, or test-takers, is 1250, and the number of columns, or variables or items, is 15.

#3.3_28. Descriptive statistics such as mean and median are shown for every item using the "summary()" command. If standard deviations of the items are desired, the command "apply(ddd,2,sd)" can be used. See also #3.1_8.

#3.3_29. The "`apply()`" command requesting a "`table`," shows frequencies of the categories for the items. This can be used to ensure that all categories were utilized for each item. See #3.1_9 for more details.

Item parameter estimation

```
mod.gpcm<-mirt(ddd, model=1, itemtype="gpcm", SE=T)
  #3.3_30
mod.gpcm #3.3_31
extract.mirt(mod.gpcm, what="converged") #3.3_32
extract.mirt(mod.gpcm, what="secondordertest") #3.3_33
coef(mod.gpcm, simplify=T, IRTpars=T) #3.3_34
coef(mod.gpcm, IRTpars=T, printSE=T) #3.3_35
coef(mod.gpcm, IRTpars=T) #3.3_36
```

#3.3_30. The GPCM is fit to the data using the "`mirt()`" function, specifying a unidimensional model ("`model=1`"), the GPCM CRF ("`itemtype="gpcm"`"), and the inclusion of the standard errors of the estimates ("`SE=T`").

#3.3_31. Convergence check results including the final log-likelihood, AIC, and BIC are printed from the fitted object, "`mod.gpcm`."

#3.3_32 and #3.3_33. Separate extractions of convergence check results are produced using the "`extract.mirt()`" function. A result of "`TRUE`" implies no particular issues detected in the estimation process.

#3.3_34. Item threshold parameter estimates are printed from "`coef()`."

```
> coef(mod.gpcm, simplify=T, IRTpars=T)
$'items'
         a     b1     b2     b3    b4
V1   2.010 -1.208 -0.580 -0.068 0.265
V2   1.547 -0.145  0.645  1.020 1.632
...
V14  0.973 -1.648 -1.252  0.648 1.526
V15  1.745 -1.278 -0.400  0.375 1.239
```

The column under "`a`" shows the item slope or discrimination estimates. The threshold parameter estimates $\left(\hat{b}_{ik}\right)$ are shown under "`b1`," "`b2`," "`b3`," and "`b4`." There are four threshold parameters because the data have five categories. Each $\hat{b}_{ik}$ corresponds to the threshold between responding in adjacent categories.

#3.3_35. The standard errors of the item parameter estimates are shown when "`printSE=T`" is included in the "`coef()`" function. The population latent trait or ability distribution parameters are set to 0 and 1 for the model identification; thus, "NA" is printed for their standard errors under "`$GroupPars`."

```
> coef(mod.gpcm, IRTpars=T, printSE=T)
$'V1'
        a     b1     b2     b3     b4
par 2.010 -1.208 -0.580 -0.068  0.265
SE  0.119  0.148  0.061  0.058  0.056

$V2
        a     b1     b2     b3     b4
par 1.547 -0.145  0.645  1.020  1.632
SE  0.090  0.178  0.071  0.085  0.108
...

$V15
        a     b1     b2     b3     b4
par 1.745 -1.278 -0.400  0.375  1.239
SE  0.097  0.163  0.059  0.059  0.073

$GroupPars
    MEAN_1  COV_11
par      0       1
SE      NA      NA
```

#3.3_36. 95% CIs for the item parameter estimates are shown when no additional arguments are supplied to the "`coef()`" function. Again, for the population parameters, "`NA`" is printed since these are fixed.

```
> coef(mod.gpcm, IRTpars=T)
$'V1'
            a      b1     b2     b3     b4
par     2.010  -1.208  -0.58 -0.068  0.265
CI_2.5  1.777  -1.498  -0.70 -0.181  0.154
CI_97.5 2.242  -0.919  -0.46  0.046  0.375

$V2
            a      b1     b2     b3     b4
par     1.547  -0.145  0.645  1.020  1.632
CI_2.5  1.370  -0.493  0.506  0.853  1.421
CI_97.5 1.724   0.203  0.784  1.186  1.843
...
$V15
            a      b1     b2     b3     b4
par     1.745  -1.278 -0.400  0.375  1.239
CI_2.5  1.555  -1.598 -0.515  0.261  1.095
CI_97.5 1.935  -0.959 -0.284  0.490  1.383

$GroupPars
        MEAN_1  COV_11
par          0       1
CI_2.5      NA      NA
CI_97.5     NA      NA
```

The "mirt" package offers two forms of the GPCM. Previously, the "`itemtype="gpcm"`" was used. An additional option is to use "`itemtype="gpcmIRT"`."

```
mod.gpcmIRT<-mirt(ddd, model=1, itemtype="gpcmIRT", SE=T)
mod.gpcmIRT
coef(mod.gpcmIRT, simplify=T, IRTpars=T)
```

The threshold estimates from this option are not the same as those from the "`gpcm`" option, although the same degree of model-data fit is expected. The same, or nearly-the-same, (final) log-likelihood values are obtained from both options. For this data, the log-likelihood values of the final model are −22720.56 and −22720.55 for the "`gpcm`" and "`gpcmIRT`," respectively. The item parameter estimates from both option approaches are also close to each other, but not exactly the same. The "`coef()`" output of "`gpcmIRT`" contains an additional column named as "`c`" which is fixed as 0. This "`c`" becomes part of a model parameter for each item in the "mirt" RSM parameterization.

The results of the original GPCM fit in the "mirt" package, #3.3_30 and "`itemtype="gpcm",`" follow the CRF presented in Equation 3.3_1, which is the threshold parameterization of the item parameters. If the location plus deviation parameterization of the GPCM, presented in Equation 3.3_2, is desired, the commands shown here produce the overall location or difficulty, and the deviation parameter values.

```
gpcm.thresh<-coef(mod.gpcm, simplify=T, IRTpars=T)
$'items'[,-1]
overall.diff<-apply(gpcm.thresh, 1, mean)
deviat<-data.frame(-gpcm.thresh+overall.diff)
names(deviat)<-c("dev1", "dev2", "dev3", "dev4")
cbind(overall.diff, deviat) #3.3_37
```

The threshold parameters of the GPCM fit, b_{ik}, are averaged across categories for each item and stored as "`overall.diff`," which is the overall location parameter, b_i. Then the negative of the threshold parameters are added to the overall location parameters, which are the deviation values for each category; these are stored as "`deviat`."

#3.3_37. The estimates for the overall location or difficulty, and the deviation parameters, are printed.

```
> cbind(overall.diff, deviat)
    overall.diff      dev1       dev2        dev3       dev4
V1   -0.39776736 0.8107264 0.18210700 -0.33023412 -0.6625993
V2    0.78791735 0.9329003 0.14288232 -0.23164572 -0.8441369
...
V14  -0.18144231 1.4669335 1.07063523 -0.82970773 -1.7078610
V15  -0.01596123 1.2622306 0.38374569 -0.39129110 -1.2546851
```

While the RSM restricts the deviation parameters to be the same across all items, the GPCM allows freely estimated item slope or discrimination, and threshold

parameters. If the Rasch family model's parameterization for the overall location or difficulty, plus deviation ($b_{ik} = b_i + d_{ij}$) is desired, one can simply the change the signs of the estimates from the last command, i.e., by tying "`-cbind(overall.diff, deviat).`"

Person latent trait, or ability score estimation

```
fscores(mod.gpcm, method="EAP", full.scores=T, full.scores.SE = T)
#3.3_38
```

#3.3_38. Person ability scores are estimated with the EAP estimator using the "`fscores()`" function. In addition to EAP, "mirt" provides MAP and WLE estimator options. The score estimates are provided in the first column, labeled "`F1`," and the standard errors are in the second column, labeled "`SE_F1`."

CCC, TCC, and information plots

```
itemplot(mod.gpcm, 1) #3.3_39
plot(mod.gpcm, type = "trace") #3.3_40
plot(mod.gpcm) #3.3_41
plot(mod.gpcm, type="info") #3.3_42
itemplot(mod.gpcm, 1, type="info") #3.3_43
```

Similar functions for the CCC, TCC, and information plots using the "mirt" package for the PCM are provided in #3.1_47 through #3.1_51; for examples for the RSM, see #3.2_14 through #3.2_17.

#3.3_39. CCCs for individual items are produced using the "`itemplot()`" function, supplying the requested item as the second argument. Here, "`1`" prints the CCCs for item 1.

#3.3_40. CCCs for all items are drawn using the "`plot()`" function with "`type= "trace"`."

#3.3_41. The TCC, which is a plot of the estimated latent trait scores on the x-axis and the total sum score on the y-axis, is drawn using the "`plot()`" function with no additional arguments.

#3.3_42. The test information curve is produced using the "`plot()`" function with "`type="info"`."

#3.3_43. Each item's information curve is plotted using the "`itemplot()`" function with "`type="info"`." The second argument indicates the requested item; here "`1`" requests the curve for item 1.

Model-data fit and model comparison

```
M2(mod.gpcm, type="M2*") #3.3_44
itemfit(mod.gpcm) #3.3_45
itemfit(mod.gpcm, empirical.plot=1) #3.3_46
personfit(mod.gpcm, method="EAP") #3.3_47
residuals(mod.gpcm, type="LD") #3.3_48
residuals(mod.gpcm, type="Q3") #3.3_49
```

Model-data fit statistics for the PCM using the "mirt" package are provided in #3.1_52 through #3.1_57. Those for the RSM are provided in #3.2_18 through #3.2_23.

#3.3_44. M2* which is an overall model-data fit test is requested with the "`M2()`" function, with "`type="M2*"`." The output includes the test results of M2* and other descriptive model fit indices such as RMSEA. The p-value of M2* for this example is 0.746. Under $\alpha = 0.05$, we fail to reject the fitted GPCM. See #3.1_52 for more descriptions of the M2* and the C2 test for limited information test approaches for polytomous item response data.

```
> M2(mod.gpcm, type="M2*")
            M2 df         p RMSEA RMSEA_5   RMSEA_95      SRMSR      TLI CFI
stats 38.39453 45 0.7461281     0       0 0.01400288 0.01102601 1.002489   1
```

#3.3_45. The item fit test, S-X2, is applied to every item using "`itemfit()`." The p-values are shown at the last column. With and without the Bonferroni correction for the family-wise Type I error rate of 0.05, no item is statistically flagged.

```
> itemfit(mod.gpcm)
   item     S_X2  df.S_X2  p.S_X2
1    V1   90.633       99   0.714
2    V2  100.116      107   0.668
...
14  V14  139.006      132   0.321
15  V15  102.545      109   0.656
```

#3.3_46. A graphical check of model-based CCCs (displayed as smooth curves) and empirical CCCs (plotted with circles) is provided using the "`itemfit()`" function with "`empirical.plot=1`" for item 1.

#3.3_47. The person fit measure, l_z, statistic ("`Zh`" in the output) is calculated for every person. If the value of l_z is small, e.g., less than negative three, one may investigate the case for a potential aberrant response pattern. The following code can be used to print the person ID, item response data, and l_z statistic, sorted by the value of the statistic.

```
p.gpcm.fit <- cbind(1:nrow(ddd), ddd, personfit(mod.gpcm,
method="EAP"))
head(p.gpcm.fit)
p.gpcm.fit[order(p.gpcm.fit[,17]),]
```

#3.3_48. The index of local dependence, LD-X2 is calculated for every pair of items and shown in a matrix form (the diagonal elements) using the "`residuals()`" function and specifying "`type="LD"`." The upper diagonal elements are Cramer's V coefficients based on the LD-X2 values. A large absolute value of LD-X2 or Cramer's V coefficient indicate a potential case of local dependence. See #2.1_57.

#3.3_49. Another measure of local dependency, Q3, is presented for every item pair in a matrix form using the "`residuals()`" function and specifying "`type="Q3"`." A large absolute value of Q3 indicates a potential violation of local independence. See #2.1_58 for more descriptions of Q3 and a set of code to print the results in an ordered manner for easily identifying potentially dependent item pairs.

A relative model comparison between the unconstrained GPCM and common-slope GPCM, where all item slopes are constrained to be equal, can be tested using the likelihood ratio test because the common-slope GPCM is nested within the unconstrained GPCM.

```
spec<-'F = 1-15
CONSTRAIN = (1-15, a1)' #3.3_50
mod.gpcm.ca<-mirt(ddd, model=spec, itemtype="gpcm", SE=T)
#3.3_51
anova(mod.gpcm.ca, mod.gpcm) #3.3_52
```

#3.3_50. A syntax for GPCM with the equal slope constraint on all 15 items is specified.

#3.3_51. Estimation of the common-slope GPCM with the syntax made from the previous step is done using the "`mirt()`" function by including "`model=spec`." Typing "`coef(mod.gpcm.ca, simplify=T, IRTpars=T)`" shows that the estimated common slope is 1.4.

#3.3_52. "`anova()`" is used for the model comparison test, with the first argument being the reduced model (`mod.gpcm.ca`) and the second argument being the full model (`mod.gpcm`). The *p*-value, shown at the last column under "p," of the log-likelihood test is near zero; thus, rejecting the reduced model, i.e., the common-slope GPCM, in favor of the unconstrained GPCM.

```
> anova(mod.gpcm.ca, mod.gpcm)
Model 1: mirt(data = ddd, model = spec, itemtype = "gpcm", SE = T)
Model 2: mirt(data = ddd, model = 1, itemtype = "gpcm", SE = T)
       AIC     AICc    SABIC      HQ      BIC     logLik      X2   df    p
1 45867.94 45874.31 45987.17 45985.61 46180.93 -22872.97     NaN  NaN  NaN
2 45591.11 45600.82 45737.70 45735.78 45975.93 -22720.56 304.832   14    0
```

The values of the information criteria, AIC, AICc, BIC, and SABIC are smaller for unconstrained GPCM than those for the common-slope GPCM, preferring the larger model (i.e., the full GPCM) to the smaller model (i.e., the common-slope GPCM).

3.4 Graded response model (GRM) application

The widely known graded response model (GRM; Samejima, 1969) is originally introduced as a homogenous case of the graded response model in the sense that the shape of the cumulative category response functions are the same, and they never cross each other. Previously, the PCM, RSM, and GPCM modeled the probability of responding in category k of the ith item. The GRM utilizes a cumulative response to model the probability of responding in category k or higher. The cumulative response probability function of the ith item for responding in category k or higher (i.e., $k \in \{0,1,2,\ldots,m_i\}$) is

$$P\left(X_{ij} \geq k|\, \theta_j\right) = \frac{\exp\left[a_i\left(\theta_j - b_{ik}\right)\right]}{1+\exp\left[a_i\left(\theta_j - b_{ik}\right)\right]}, \tag{3.4\_1}$$

where X_{ikj} is the jth person's response in category k or higher to the ith item. a_i is the ith item slope or discrimination parameter, θ_j is the jth person's latent trait or ability score, and b_{ik} is the category boundary parameter for the ith item's category k or higher.

The GRM for responding in category k or higher may be related to the dichotomous 2PLM IRF, where the category response options are dichotomized. The first category boundary parameter, b_{i1}, is the boundary between category 0 and category 1 or higher; the second category boundary parameter, b_{i2}, is the boundary between categories less than category 2 and category 2 or higher. The last category parameter, b_{im}, is the boundary between responding in categories 0 through $m_i - 1$ and category m_i. Similar to the 2PLM item difficulty parameter, the cumulative GPCM's b_{ik} is a point on the θ scale where the cumulative response probability for k or higher is 0.50.

The probability of responding specifically in category k, $X_{ikj} = k$ is obtained by subtracting adjacent cumulative CRFs:

$$(X_{ikj} = k|\, \theta_j) = P(X_{ikj} \geq k|\, \theta_j) - P(X_{i(k+1)j} \geq k+1|\, \theta_j). \tag{3.4\_2}$$

Because the probability of responding in category k is calculated as the difference between two cumulative response probabilities, it is called a "difference model" (Thissen & Steinberg, 1986). The category boundary parameter, b_{ik}, can be re-parameterized into the overall location (b_i) plus deviation (d_{ik}) form, following either the Rasch family decomposition ($b_{ik} = b_i + d_{ik}$) or the GPCM decomposition ($b_{ik} = b_i - d_{ik}$). Compared with the threshold parameters from the GPCM (or PCM), the category boundary parameters from the GRM are always ordered, i.e., $b_{ik} \geq b_{i(k-1)}$. When the number of categories is two the GRM reduces to the 2PLM. Applications of GRM using "ltm" and "mirt" are introduced here.

GRM application with the "ltm" package

```
install.packages("ltm") #3.4_1
library(ltm) #3.4_2
setwd("c:/mystudy/RIRT/") #3.4_3
ddd<-read.table("c3_grm.dat", header=T) #3.4_4
dim(ddd) #3.4_5
summary(ddd) #3.4_6
names(ddd) #3.4_7
apply(ddd, 2, table) #3.4_8
```

#3.4_1. If the "ltm" package is not installed, use the "`install.packages()`" function to install the "ltm" package.

#3.4_2. After downloading the "ltm" package, the package should be uploaded into the current R session using the "`library()`" command.

#3.4._3. Set up the working directory where data files exist and R output files will be stored using "`setwd()`." The "`c3_grm.dat`" file should be stored in the specified working directory folder.

#3.4_4. Read the raw data file into R using the "`read.table()`" function, and save it as "`ddd`." The raw data file has the space-delimited format. The first row of the data file includes the variable or item names. To read the variable names, the "`header = T`" argument is included in the "`read.table()`" function.

#3.4_5. Dimensions of the data set are provided using "`dim()`." The number of rows or test-takers is 880, and the number of columns or items is 14.

#3.4_6. Descriptive statistics of the item responses for every item are printed using "`summary()`." Each item has five responses of 0, 1, 2, 3, and 4. The means represents the average level of endorsement; a mean of two corresponds to the middle or most neutral response.

#3.4_7. The "`name()`" command show the variable, item, names read from the raw data file. They are "It.1," "It.2," . . . , "It.14."

#3.4_8. Frequencies of responses within each category for the items are shown using the "`apply()`" command and requesting a "`table`." See #3.1_9.

Item parameter estimation

```
mod.grm <- grm(ddd, IRT.param=T, Hessian=T) #3.4_9
mod.grm$conv #3.4_10
mod.grm #3.4_11
coef(mod.grm) #3.4_12
summary(mod.grm) #3.4_13
```

#3.4_9. Item parameters from the GRM are estimated using the "`grm()`" function with the request of the standard error calculation using the Hessian matrix, "`Hessian=T`." By default, no standard error is calculated. Requesting the standard error does take a longer model run time. Including "`IRT.param=T`" estimates item parameters according to Equation 3.4_1, with the item slope or discrimination and category boundary. If "`IRT.param=F`," the item parameters are estimated under the slope and intercept form.

#3.4_10. A quick check on convergence is done with "`mod.grm$conv`." "`0`" in the response means no particular issues were detected by the default convergence criteria in the process of estimation.

#3.4_11 and #3.4_12. The point estimates of the boundary category parameters ($b_{i(k=1)}$, $b_{i(k=2)}$, $b_{i(k=3)}$, and $b_{i(k=4)}$) are shown under the names "`Extrmt1`," "`Extrmt2`," "`Extrmt3`," and "`Extrmt4`." The slope or discrimination parameters (a_i) are printed under "`Dscrmn`.' #3.4.11 "`mod.grm`" prints the item parameter estimates and one additional output of the final log-likelihood value.

```
> coef(mod.grm)
      Extrmt1 Extrmt2 Extrmt3 Extrmt4 Dscrmn
It.1    0.775   0.934   1.259   1.740  1.132
It.2   -1.006  -0.709  -0.282   0.288  2.156
...
It.13  -0.572  -0.266  -0.001   0.861  2.280
It.14  -1.287  -1.054  -0.345   0.755  2.216
```

If the location minus deviation parameterization is desired, as in the GPCM form (i.e., $b_{ik} = b_i - d_{ik}$), the category boundary parameters can be decomposed using the following commands.

```
grm.cbp<-coef(mod.grm)[,-5]
overall.diff<-apply(grm.cbp, 1, mean)
deviat<-data.frame(-grm.cbp+overall.diff)
names(deviat)<-c("dev1", "dev2", "dev3", "dev4")
cbind(overall.diff, deviat)
```

First, the boundary, or "`Extrmt`," values are extracted from the coefficients and stored in "`grm.cbp`." Then these category boundary values are averaged across categories for each item, which are the item locations; these are saved as "`overall.diff`." The "deviat" values are calculated as the difference between the negative of the category boundaries and the overall location, or "`overall.diff`." The deviations are renamed for clarity. The last command, "`cbind()`," combines and prints the overall location, or difficulty, (b_i) and the deviation (d_{ik}) parameter estimates.

```
> cbind(overall.diff, deviat)
     overall.diff    dev1    dev2     dev3     dev4
It.1      1.17700 0.40200 0.24300 -0.08200 -0.56300
It.2     -0.42725 0.57875 0.28175 -0.14525 -0.71525
...
It.13     0.00550 0.57750 0.27150  0.00650 -0.85550
It.14    -0.48275 0.80425 0.57125 -0.13775 -1.23775
```

If the PCM or RSM Rasch family's overall location plus deviation parameterization (i.e., $b_{ik} = b_i + d_{ik}$) is sought for the category boundary parameters; simply changing the sign of the final estimates for the deviation parameter estimates is sufficient. Thus, at the last command of "`cbind`," use the negative of "`deviat,`" i.e., "`cbind(overall.diff, -deviat)`."

#3.4_13. The standard errors, information criteria (AIC and BIC), and the convergence check results are printed together with the item parameter estimates using the "`summary()`" function.

```
> summary(mod.grm)
Call:
grm(data = ddd, IRT.param = T, Hessian = T)
Model Summary:
log.Lik      AIC      BIC
 -15368 30876.01 31210.6
Coefficients:
$'It.1'
        value std.err z.vals
Extrmt1 0.775   0.089  8.701
Extrmt2 0.934   0.477  1.959
Extrmt3 1.259   0.669  1.882
Extrmt4 1.740   0.981  1.773
Dscrmn  1.132   0.102 11.056
...
$It.14
         value std.err   z.vals
Extrmt1 -1.287   0.075  -17.053
Extrmt2 -1.054   0.103  -10.226
Extrmt3 -0.345   0.072   -4.776
Extrmt4  0.755   0.125    6.061
Dscrmn   2.216   0.132   16.815

Integration:
method: Gauss-Hermite
quadrature points: 21

Optimization:
Convergence: 0
max(|grad|): 1.1
quasi-Newton: BFGS
```

Person latent trait, or ability score estimation

```
factor.scores(mod.grm, method="EAP") #3.4_14
```

#3.4_14. For each unique observed response pattern, a latent trait or ability score is estimated using the EAP estimator with the "`factor.scores()`" function. The column titled "`z1`" is the latent trait estimate, and the column titled "`se.z1`" is the standard error. See #3.3_13 as well.

CCC and information plots

```
plot(mod.grm) #3.4_15
plot(mod.grm, type="IIC", item=0) #3.4_16
plot(mod.grm, type="IIC") #3.4_17
plot(mod.grm, type="IIC", item=c(1, 2, 5), legend=T)
#3.4_18
```

#3.4_15. CCCs are shown for every item using the "`plot()`" function. Users must click the graphical window or press "Enter" to progress through each item's plot. The GRM in Equation 3.4_1 models the cumulative response probabilities, not the CCCs. The values of the CCCs are obtained by Equation 3.4_2, after GRM parameters are estimated. Note again that the category boundary parameter, b_{ik}, for the GRM is the value on the θ scale that corresponds to the probability of 0.50 for responding in category k or higher, not the category-intersection parameter, which is the threshold parameter in the RSM, PCM, and GPCM.

#3.4_16. The test information plot is shown using "`type="IIC", item=0`" in the "`plot()`" function.

#3.4_17. All item information curves are displayed in one plot using "`type="IIC"`" in the "`plot()`" function.

#3.4_18. Item information curves for specific items, here for items 1, 2, and 5, are requested and drawn in a single plot with the legend when "`type="IIC", item=c(1, 2, 5), legend=T`" is included in the "`plot()`" function.

Model-data fit and model comparison

For the GRM cali.bration, "ltm" is not equipped with an overall model-data fit measure or the item- and person-level fit statistics. However, the usual model comparison test using the likelihood ratio test is possible when a reduced model (e.g., a common-slope GRM) is compared with the unconstrained GRM.

```
mod.grm.ca <- grm(ddd, constrained=T, IRT.param=T)
  #3.4_19
mod.grm.ca$conv #3.4_20
mod.grm.ca #3.4_21
anova(mod.grm.ca, mod.grm) #3.4_22
```

#3.4_19. Including "`constrained = T`" in the "`grm()`" function imposes the equality constraint on the item slope parameters. This constrains all slope parameters to be equal in the GRM (but not necessarily equal to 1.0).

#3.4_20. A quick convergence check using "`mod.grm.ca$conv`" is done to ensure no particular issues are detected in the estimation process; this is confirmed with the output "`0`."

#3.4_21. The common-slope GRM parameter estimates are shown from the object "`mod.grm.ca`." The common item slope is estimated to be 1.541.

#3.4_22. The *p*-value of the log-likelihood ratio test result is shown in the last column from the "`anova()`" test, which is less than 0.001. The common-slope GRM, or reduced model, is rejected, favoring the full, unconstrained GRM. The AIC and BIC also prefer GRM by showing smaller values for GRM than the common-slope GRM.

```
> anova(mod.grm.ca, mod.grm)

Likelihood Ratio Table
               AIC      BIC   log.Lik    LRT df p. value
mod.grm.ca 31262.45 31534.91 -15574.23
mod.grm    30876.01 31210.60 -15368.00 412.45 13  <0.001
```

GRM calibration with the "mirt" package

The "mirt" package can also estimate the GRM. Like the "ltm" package, "mirt" uses the MML estimation, and the population ability distribution is assumed to follow a normal distribution. For model identification, the mean and the variance of the normal distribution are fixed as 0 and 1, respectively.

```
install.packages("mirt") #3.4_23
library(mirt) #3.4_24
setwd("c:/mystudy/RIRT/") #3.4_25
ddd<-read.csv("c3_grm2.csv") #3.4_26
head(ddd) #3.4_27
dim(ddd) #3.4_28
summary(ddd) #3.4_29
apply(ddd, 2, table) #3.4_30
```

#3.4_23. Install the "mirt" package on the computer using "`install.packages()`," if it is not already installed.

#3.4_24. Upload the "mirt" package onto the current R session using the "`library()`" function.

#3.4_25. The working directory where data and R output files are stored is established using "`setwd()`." The "`c3_grm2.csv`" file should be in the specified working directory folder.

#3.4_26. The raw data file has a comma-separated values (CSV) format with the variable names at the first row. To read a CSV data file, use the "`read.csv()`" function. The headings of the columns are automatically stored as the column names.

#3.4_27. The "`head()`" function is used to observe the first six rows, or test-takers', item response data. The item labels are also printed above each column ("Item1," "Item2," . . . , "Item14").

#3.4_28. Dimensions of the data set, "`ddd`," are printed using the "`dim()`" function. The number of rows or test-takers is 725, and the number of columns or items is 14.

#3.4_29. Descriptive statistics (minimum, maximum, mean, and 1st, 2nd, and 3rd, quartile) for every item are printed using the "`summary()`" function. Here, the responses for each item are shown to be on a four-point scale of 0, 1, 2, and 3. If only the mean or the standard deviation is desired, then the commands "`apply(ddd, 2, mean)`" and "`apply(ddd, 2, sd)`" can be used.

#3.4_30. Frequencies of item response categories are printed using the "`apply()`" function and requesting a "`table`" of responses for each item. See #3.1_9.

Item parameter estimation

```
mod.grm<-mirt(ddd, model=1, itemtype="graded", SE=T)
  #3.4_31
mod.grm #3.4_32
extract.mirt(mod.grm, what="converged") #3.4_33
extract.mirt(mod.grm, what="secondordertest") #3.4_34
coef(mod.grm, simplify=T, IRTpars=T) #3.4_35
coef(mod.grm, IRTpars=T, printSE=T) #3.4_36
coef(mod.grm, IRTpars=T) #3.4_37
```

#3.4_31. The "`mirt()`" function is used to fit the data to a unidimensional model. In the "`itemtype`," "`graded`" is specified to estimate the GRM.

#3.4_32. Overall convergence results with descriptive information criteria, i.e., AIC, AICc, BIC, and SABIC, including the final log-likelihood are shown from the object "`mod.grm`."

#3.4_33 and #3.4_34. Separate extractions of the convergence check results are done using "`what="converged"`" and "`what="converged"`" in the "`extract.mirt()`" function. For both checks, the output "`TRUE`" indicates no particular problems detected under the default convergence criteria in the estimation process.

#3.4_35. Item parameter estimates of the category boundary parameters (b_{ik}; these are "`b1`," "`b2`," and "`b3`" in the output) and the slope (a_i; "`a`" in the

output) are shown using the "`coef()`" function and "`IRTpars=T`." If "`IRTpars=F`," the parameters from the slope and intercept form are printed.

```
> coef(mod.grm, simplify=T, IRTpars=T)
$'items'
           a     b1     b2    b3
Item1  0.991 -0.595 -0.082 1.058
Item2  1.040 -0.797  0.518 1.487
...
Item13 0.961 -0.626  0.253 0.716
Item14 1.282 -1.493  0.140 1.170

$means
F1
0

$cov
  F1
F1 1
```

#3.4_36. The point estimates of item parameters and their standard errors are shown using the "`coef()`" function and "`printSE=T`."

```
> coef(mod.grm, IRTpars=T, printSE=T)
$'Item1'
        a     b1     b2    b3
par 0.991 -0.595 -0.082 1.058
SE  0.095  0.103  0.090 0.126

$Item2
        a     b1    b2    b3
par 1.040 -0.797 0.518 1.487
SE  0.094  0.108 0.094 0.145
...

$Item14
        a     b1    b2    b3
par 1.282 -1.493 0.140 1.170
SE  0.103  0.124 0.076 0.107

$GroupPars
    MEAN_1 COV_11
par      0      1
SE      NA     NA
```

#3.4_37. The point estimates of item parameters and their 95% CIs are printed using the "`coef()`" function and "`IRTpars=T`" without "`simplify=T`."

```
> coef(mod.grm, IRTpars=T)
$'Item1'
            a     b1     b2    b3
par     0.991 -0.595 -0.082 1.058
CI_2.5  0.804 -0.797 -0.258 0.811
CI_97.5 1.178 -0.392  0.094 1.304
...
$Item14
            a     b1     b2   b3
par     1.282 -1.493  0.140 1.17
CI_2.5  1.080 -1.736 -0.009 0.96
CI_97.5 1.483 -1.249  0.288 1.38

$GroupPars
        MEAN_1 COV_11
par          0      1
CI_2.5      NA     NA
CI_97.5     NA     NA
```

If one wants to express the category boundary parameters as the overall location, or difficulty, minus deviation parameterization as in the GPCM framework ($b_{ik} = b_i - d_{ik}$), the following steps can be taken.

```
grm.cbp<-coef(mod.grm, simplify=T, IRTpars=T)$items[,-1]
overall.diff<-apply(grm.cbp,1, mean)
deviat<-data.frame(-grm.cbp+overall.diff)
names(deviat)<-c("dev1", "dev2", "dev3")
cbind(overall.diff, deviat) #3.4_38
cbind(overall.diff, -deviat) #3.4_39
```

#3.4_38. "`cbind(overall.diff, deviat)`" shows the overall difficulty and the deviation defined in the GPCM parameterization.

```
> cbind(overall.diff, deviat)
      overall.diff      dev1        dev2       dev3
Item1   0.12694356 0.7216417  0.20891600 -0.9305577
Item2   0.40255794 1.1997641 -0.11537834 -1.0843858
...
Item13  0.11414058 0.7400525 -0.13842146 -0.6016311
Item14 -0.06117867 1.4316337 -0.20077579 -1.2308579
```

#3.4_39. To express the category boundary parameters in the location plus deviation form, as in the PCM and RSM Rasch family parameterization ($b_{ik} = b_i + d_{ik}$), the negative of the GPCM deviation parameters is applied ("`cbind(overall.diff, -deviat)`"). The overall difficulty values

are the same in both GPCM and Rasch family definitions, while the deviation is defined as being opposite in direction.

Person latent trait, or ability score estimation

```
fscores(mod.grm, method="EAP", full.scores=T, full.
  scores.SE = T) #3.4_40
```

#3.4_40. EAP estimates for person latent trait or ability scores are calculated using the "`fscores()`" function; "`method="EAP"`" is used to specify the EAP method rather than some other procedure, "`full.scores=T`" prints the scores of all test-takers rather than unique response patterns, and "`full.scores.SE = T`" prints the standard errors. The column titled "`F1`" shows the EAP latent trait estimates, and the "`SE_F1`" column shows the standard deviation of the individual person's posterior distribution, which is interpreted as the standard error.

CCC, TCC, and information plots

```
itemplot(mod.grm, 9) #3.4_41
plot(mod.grm, type = "trace") #3.4_42
plot(mod.grm) #3.4_43
plot(mod.grm, type="info") #3.4_44
itemplot(mod.grm, 1, type="info") #3.4_45
```

#3.4_41. CCCs are produced using the "`itemplot()`" function, where the second argument is the specific item requested. Here, the CCCs of item 9 are shown. CCCs are calculated from the cumulative response probabilities in Equation 3.4_2. See also #3.4_15.

#3.4_42. CCCs for all items are shown using the "`plot()`" function with "`type = "trace"`."

#3.4_43. The TCC for the test is produced with the function "`plot()`," and supplying no additional arguments.

#3.4_44. The test information plot is produced using the "`plot()`" function with "`type="info"`."

#3.4_45. "`itemplot()`" is used to print the information curve for a specific item, where the second argument is the specific item requested and "`type = 'trace'`." Here, the item information curve for item 1 is shown.

Model-data fit and model comparison

```
M2(mod.grm, type="M2*") #3.4_46
itemfit(mod.grm) #3.4_47
itemfit(mod.grm, empirical.plot=1) #3.4_48
personfit(mod.grm, method="EAP") #3.4_49
residuals(mod.grm, type="LD") #3.4_50
residuals(mod.grm, type="Q3") #3.4_51
```

#3.4_46. M2*, a limited information overall model fit test, is conducted using the "`M2()`" function and specifying "`type="M2*"`." The *p*-value of the M2* test is 0.833, which fails to reject the fitted GRM. In addition, the output contains descriptive model-data fit indices such as RMSEA. See #3.1_42 for more descriptions of the limited information test approaches.

```
> M2(mod.grm, type=" M2*")
            M2 df         p RMSEA RMSEA_5   RMSEA_95     SRMSR      TLI CFI
stats 39.46771 49 0.8327256     0       0 0.01482469 0.0219067 1.009099   1
```

#3.4_47. Item level fit of the model with the data is investigated with the S-X2 test using "`itemfit()`." The *p*-value of the test for every item is shown at the last column titled "`p.S_X2`."

```
> itemfit(mod.grm)
     item    S_X2  df.S_X2  p.S_X2
1   Item1 113.815       85   0.020
2   Item2  83.817       84   0.485
...
13 Item13  73.116       83   0.773
14 Item14  77.075       78   0.508
```

When the family-wise Type I error rate is 0.05, and the Bonferroni correction is applied, no item is statistically flagged.

#3.4_48. Model-based CCCs (solid curves) are compared with empirical CCCs (circles) in a graphic using the "`"itemfit()" function and "empirical.plot="`." The value supplied into "`empirical.plot=1`" is the item for which the graphic is produced; here, it is item 1.

#3.4_49. The l_z person fit index ("`Zh`") is calculated. A very low value (e.g., < –3) may indicate a potential aberrant response pattern. The following can be used to better identify the flagged responses.

```
p.fit <- cbind(1:725,ddd, personfit(mod.grm, method="EAP"))
head(p.fit)
p.fit[order(p.fit[,16]),]
```

In this example data, three test-takers' responses were flagged using the "`Zh`" statistic with the "`Zh`" < –3 rule: person 214, 532, and 642.

#3.4_50. LD-X2 is calculated for each pair of items for examining local dependence using the "`residuals()`" function. The lower diagonal elements in the output are signed LD-X2 values, and the upper diagonal elements are signed Cramer V coefficients calculated using the LD-X2. A large absolute value in LD-X2 indicates a potential violation of local independence. See #2.1_57.

#3.4_51. The Q3 index for assessing local dependence is calculated for every item using the "`residuals()`" function. The larger in its absolute value is, the more serious the local dependence is. See #2.1_58 for more information on Q3 and a set of code to print the results in an ordered manner for easily identifying potentially dependent item pairs.

A submodel of the GRM, one where the item slopes are assumed to be the same, may be compared with the full GRM to evaluate the usefulness of the unconstrained GRM. Because the common-slope GRM is nested within GRM, the log-likelihood ratio test can be used for the two model comparison.

```
spec<-'F = 1-14
CONSTRAIN = (1-14, a1)' #3.4_52
mod.grm.ca<-mirt(ddd, model=spec, itemtype="graded",
SE=T) #3.4_53
anova(mod.grm.ca, mod.grm) #3.4_54
```

#3.4_52. The equal item slope constraint for all 14 items is specified in the two-line syntax contained in single quotations.

#3.4_53. The common-slope GRM is estimated using the model syntax created in the previous step by including "`model=spec`." The estimated parameters of the constrained GRM can be observed using "`coef(mod.grm.ca, simplify=T, IRTpars=T)`." The common item slope was estimated to be 1.379.

#3.4_54. The log-likelihood ratio test between the reduced, or nested, GRM and the full, or unconstrained, GRM is done using the "`anova()`" function. The *p*-value of the test is near zero, favoring the full or unconstrained GRM and rejecting the common-slope or reduced GRM under an $\alpha = 0.05$ level of significance. Information criteria (AIC, AICc, BIC, and SABIC) also prefer the GRM to the common-slope GRM by having smaller values.

```
> anova(mod.grm.ca, mod.grm)
Model 1: mirt(data = ddd, model = spec, itemtype = "graded", SE = T)
Model 2: mirt(data = ddd, model = 1, itemtype = "graded", SE = T)
       AIC     AICc    SABIC       HQ      BIC    logLik      X2  df   p
1 22454.94 22460.50 22515.61 22531.05 22652.15 -11184.47     NaN NaN NaN
2 22343.82 22353.38 22422.83 22442.94 22600.65 -11115.91 137.122  13   0
```

3.5 Nominal response model (NRM) application

All polytomous response models introduced previously are for ordered item response data. When item responses are nominal, the nominal response model (NRM; Bock, 1972) may be used. For instance, the NRM may be considered for distractor, or choice option, analysis in a test that consists of multiple-choice items, for examining the order of categories empirically, or for modeling testlets, which are sets of items that form clusters of items in a test due to common stimuli. Typically, when

multiple-choice items are used for a test, the responses are dichotomized, where a correct answer is scored as one category (usually scored 1) and all incorrect option choices are aggregated to the other category (usually scored 0). The NRM maintains all item option responses by treating all option responses as nominal.

We have previously addressed that the RSM is a submodel of the PCM, which is a submodel of the GPCM. The RSM, PCM, and GPCM (but not GRM) are all submodels of the NRM, which is the most general case of the "divided-by-total" model (Thissen & Steinberg, 1986). Under the NRM, the item slope parameter is estimated for each category within the items. The NRM CRF for modeling the response in category $k \in \{0,1,2,\ldots,m_i - 1\}$ of the ith item has the following slope and intercept form:

$$P\left(X_{ij} = k|\, \theta_j\right) = \frac{\exp\left(a_{ik}\theta_j - c_{ik}\right)}{\sum_{h=1}^{m_i-1}\exp\left(a_i\theta_j - c_{ih}\right)}\,, \tag{3.5_1}$$

where X_{ikj} is the jth person's response in the category k for the ith item, θ_j is the jth person's latent trait or ability score, a_{ik} is the item slope parameter for the kth category in the ith item, and c_{ik} is the intercept parameter for the ith item category k.

The category with the largest estimated slope for an item always has a monotonically increasing CCC along the latent trait or θ scale, and the category with the smallest slope for an item always has a monotonically decreasing CCC along the θ scale. The categories with estimated slope values between the largest and the smallest takes either the shapes of unimodal or bell-shaped curves, or a monotonically increasing (or decreasing) shape along the θ scale.

The intercept parameter (c_{ik}) in NRM is interpreted as the popularity or attractiveness of the kth category. The magnitude of a_{ik} is useful for interpreting the ordering of categories. For example, in a three-category item, the values of $a_{i0} = 1$, $a_{i1} = 0$, and $a_{i2} = -1$ are interpreted as category 2 being the lowest, category 1 being the middle, and category 0 being the highest. If the ordering of the two categories are the same, $a_{ik} = a_{i(k-1)}$, but $c_{ik} > c_{i(k-1)}$, then the CCC of the category k is higher than that of the category $k - 1$. Note that the intercept parameters in NRM are not the category-intersection parameter or the category boundary parameters, as was the case in other polytomous models.

For model identification, in addition to the ability population distribution following a normal distribution with its mean and variance fixed as 0 and 1, respectively, both $\Sigma_k a_{ik} = 0$ and $\Sigma_k c_{ik} = 0$ are used. Another set of constraints used in some IRT programs for the NRM is $a_{i(k=0)} = 0$ and $c_{i(k=0)} = 0$. The "mirt" package is used to calibrate the NRM in the following section.

NRM calibration with the "mirt" package

The actual model used for the estimation of the NRM in "mirt" does not use the parameterization shown in Equation 3.5_1. The estimation model for the NRM in "mirt" employs a newly ereparametrized form where the item category slope is expressed in terms of an overall item slope (a_i) multiplied by the kth category parameter (a_{ik}) called the scoring function parameter for the category k in the ith item:

$a_{ik} = a^{\star}_{ik} a_i$. See Chalmers (2019) and Thissen, Cai, and Bock (2010) for more information of the new parameterization of NRM. In "mirt'" the initial estimates of the new multiplicative parameterization are transformed into the estimates of a_{ik} and c_{ik} in Equation 3.5_1, when item parameter estimates are requested with "`IRTpars=T`."

NRM has the most flexible form of all divided-by-total models, freely estimating the slope and intercept parameters for every category in each item. At the cost of theoretical flexibility, the NRM seems to suffer from the difficulty of estimating an increased number of model parameters. As far as the author's (somewhat limited) experiences go, the NRM estimation in some IRT software, including "mirt," does not always appear to be stable, showing very different sets of parameter estimates depending on different starting values specified for the estimation of the model. A consequence from this is that decisions based on the item category slope parameter estimates regarding the order of categories change due to different estimates from different starting values for item parameters. Because of this, we recommend that readers be cautious when the default starting values are accepted and used to estimate the NRM when using "mirt." Readers should try different set(s) of starting values to check if the estimates stay stable across the different initial value constraints. (For the NRM application in this section, we focus on the item parameter estimation.)

An approach to obtain stable and less biased estimates of the category slope parameters (a_{ik}) is to define starting values for the theoretically expected highest and the lowest categories (Chalmers, 2019). This means that the starting values of the scoring function parameters ($a^{\star}_{ik}$) are set to reflect the rank order of the theoretically expected highest and the lowest categories. For example, consider a multiple-choice item with four response categories (0, 1, 2, 3) and strategic distractors chosen. Category 2 is the correct response, so it is expected to be the highest scoring category. Response category 0 is expected to be the lowest scoring category, corresponding to a completely wrong answer. So, $a^{\star}_{i2}$ is set to equal 3.0 to reflect that category 2 is the highest among the four options; the value of 3.0 is chosen to correspond to the highest category. $a^{\star}_{i0}$ is set to equal 0 to reflect that category 0 is the lowest among the four options, and it is set to the value of 0 to correspond to the lowest category. The middle categories' starting values may be selected to be any number between the lowest and highest; e.g., $a^{\star}_{i1} = 0.5$ and $a^{\star}_{i3} = 0.5$.

These types of starting values are illustrated in the example shown here. Although this approach mitigates the chance of biased estimates, selecting starting values to match with the order of the categories may be viewed as burdensome because it may not always be feasible.

Item parameter estimation

```
install.packages("mirt") #3.5_1
library(mirt) #3.5_2
setwd("c:/mystudy/RIRT/") #3.5_3
ddd<-read.table("c3_nrm.dat", header=T) #3.5_4
dim(ddd) #3.5_5
summary(ddd) #3.5_6
apply(ddd, 2, table) #3.5_7
```

#3.5_1. The "mirt" package should be installed on a computer using "`install.packages()`" prior to loading or using features of the package.
#3.5_2. Upload the "mirt" package into the current R session using the "`library()`" function.
#3.5_3. Set up a working directory where data files exist and R output files are saved using "`setwd()`." The "`c3_nrm.dat`" file should be stored in the specified working directory folder.
#3.5_4. The "`c3_nrm.dat`" data is a space-delimited format data set; the "`read.table()`" function is used to read in the data. It has responses from four items that are scored 1, 2, 3, and 4 for each item. The data file also contains the variable names in the first row ("it1," "it2," "it3," and "it4"), so ""`header=T`" is included in the function.
#3.5_5. Dimensions of the data "`ddd`" are printed using "`dim()`." The number of rows or test-takers is 1995, and the number of columns or items is four.
#3.5_6. Descriptive statistics are printed for every item using "`summary()`."
#3.5_7. Frequencies of item category responses are printed using "`apply()`" and requesting a "`table`" of responses for each item.

```
mod.nrm.dv<-mirt(ddd, model=1, itemtype="nominal", SE=T)
mod.nrm.dv
coef(mod.nrm.dv, simplify=T, IRTpars=T)
```

With the "`mirt()`" function, the NRM is estimated with the default starting values for item parameters with "`itemtype="nominal"`." Its estimation results are shown in the "`mod.nrm.dv`" object using "`coef()`." The object shows no particular estimation problems detected under the default convergence criteria. We show the results from this default setting to compare with those from a set of new starting values that reflect the user-specified order of the highest and the lowest categories.

```
> coef(mod.nrm.dv, simplify=T, IRTpars=T)
$'items'
        a1     a2     a3     a4     c1     c2     c3     c4
it1 -0.711 -0.328  0.225  0.814 -0.949  0.228  0.894 -0.173
it2  0.179  0.144 -0.528  0.206  0.004 -0.137  0.575 -0.441
it3 -0.354 -0.788  1.434 -0.293 -0.298  0.823 -0.683  0.158
it4 -1.003  0.817 -0.254  0.440  0.939 -0.278 -0.021 -0.641

$means
F1
 0

$cov
   F1
F1  1
```

In the result from the default starting values, the category slope estimates for item 1 show "a1" < "a2" < "a3" < "a4" (–0.711 < –0.328 < 0.225 < 0.814). This suggests the following category order for item 1: 1, 2, 3, and 4. For item 2, "a3" < "a2" < "a1" < "a4"; thus, the rank order of the categories is 3, 2, 1, and 4. For item 3, "a2" < "a1" < "a4" < "a3" indicates the category order of 2, 1, 4, and 3. For item 4, "a1" < "a3" < "a4" < "a2" shows the order of categories as 1, 4, 2, and 3. Next, we fit NRM with a new set of user-specified starting values.

```
spec<-'F = 1-4
       START = (2, ak0, 3)
       START = (2, ak1, 0.5)
       START = (2, ak2, 0.5)
       START = (2, ak3, 0)
       START = (3, ak0, 0.5)
       START = (3, ak1, 0.5)
       START = (3, ak2, 3)
       START = (3, ak3, 0)
       START = (4, ak0, 0)
       START = (4, ak1, 3)
       START = (4, ak2, 0.5)
       START = (4, ak3, 0.5)' #3.5_8
mod.nrm<-mirt(ddd, model=spec, itemtype="nominal", SE=T)
  #3.5_9
mod.nrm #3.5_10
coef(mod.nrm, simplify=T, IRTpars=T) #3.5_11
```

#3.5_8. A single factor structure for all items is established in the first line ("F = 1-4"). Following, starting values are set for the category scoring function parameters using the form "START = (item, parameter, value)." The parameters "ak0," "ak1," "ak2," and "ak3" correspond to the scoring function parameters for categories 0, 1, 2, and 3 (or 1, 2, 3, and 4 for this data).

In the new parameterization of the NRM parameterization, which has the multiplicative form of $a^{\star}_{ik}a_i$, usually the scoring function parameters for the lowest category $a^{\star}_{i(k=0)}$ ("ak0" in the "mirt" notation) and the highest category $a^{\star}_{i(k=m_i-1)}$ ("ak3" in the "mirt" notation) are set to equal 0 and $m_i - 1$ for model identification. "mirt" adopts this constraint. Also, the default starting values in "mirt" expects category 1 and category 4 to be lowest and highest, which are the theoretically expected order we assume here. For the first item, the default starting values were used, since the theoretical ordering followed the program default ordering. The items listed in the model specification are 2, 3, and 4. For each item, the value is set to 3 for the highest category, and 0 for the lowest category. The middle categories could take any values between the lowest and the highest starting values. Currently 0.5 is given for the second and the third category.

#3.5_9. The NRM is fit to the data using "`model=spec`" to link the user-provided staring values from #3.5_8 to the data with "`itemtype="nominal"`."

#3.5_10. Typing "`mod.nrm`" shows there was no specific estimation issue detected under the default convergence criteria.

#3.5_11. The estimates of a_{ik} and c_{ik} are shown using the "coef()" function, including "`IRTpars=T.`"

```
> coef(mod.nrm, simplify=T, IRTpars=T)
$'items'
        a1     a2     a3     a4     c1     c2     c3     c4
it1 -0.703 -0.394  0.232  0.866 -0.927  0.216  0.902 -0.191
it2  1.060  0.326 -0.499 -0.887 -0.237  0.002  0.732 -0.497
it3  0.841 -1.131  1.674 -1.385 -0.470  1.034 -0.692  0.128
it4 -1.020  0.760 -0.212  0.472  0.934 -0.242 -0.027 -0.666

$means
F
0

$cov
  F
F 1
```

The category slope parameters with the user-provided starting values indicate the order of categories as follows: "`a1`" < "`a2`" < "`a3`" < "`a4`" (–0.703 < –0.394 < 0.232 < 0.866) for item 1; "`a4`" < "`a3`" < "`a2`" < "`a1`" for item 2; "`a4`" < "`a2`" < "`a1`" < "`a3`" for item 3; and "`a1`" < "`a3`" < "`a4`" < "`a2`" for item 4. For items 2 and 3, we observe very different estimates of a_{ik} and c_{ik} between the two estimations using the default values ("`mod.nrm.dv`") and the user-provided starting values ("`mod.nrm`"). Therefore, conclusions about the ordering of categories for items 2 and 3 are very different based on the two models.

Users can also compare the item fit statistic, S-X2 using the following "`itemfit()`" tests. Items 2 and 4 do not fit the model using the default values (using the Bonferroni correction). However, all the items fit well to the model using the pre-defined starting value.

```
itemfit(mod.nrm.dv)
itemfit(mod.nrm)
```

The authors tried two additional sets of starting values – one set had the overall slope parameter set to 0 and the other set had zeros for the overall slope and the scoring function parameters for the middle categories. These two different sets also resulted in very similar item parameters as those in #3.5_11 ("`mod.nrm`"). In addition, a different set of starting values reflecting the rank order of the highest and the lowest categories was tried. (The syntax for the NRM estimation with this set

of staring values is shown here.) The parameter estimates shown by "mod.nrm2" were very similar to those shown by "mod.nrm." (The estimates were different at the second or third decimal point.)

```
spec<-'F = 1-4
       START = (2, ak0, 4)
       START = (2, ak1, 2.5)
       START = (2, ak2, 2.5)
       START = (2, ak3, 1)
       START = (3, ak0, 2.5)
       START = (3, ak1, 2.5)
       START = (3, ak2, 4)
       START = (3, ak3, 1)
       START = (4, ak0, 1)
       START = (4, ak1, 4)
       START = (4, ak2, 2.5)
       START = (4, ak3, 2.5)' #3.5_8
mod.nrm2<-mirt(ddd, model=spec, itemtype="nominal", SE=T)
mod.nrm2
coef(mod.nrm2, simplify=T, IRTpars=T)
```

The data set used in this illustration was simulated data, where the authors set the data-generating item parameter values with which the results from the user-specified starting were checked. From the known true values, we confirmed that the user-specified starting values in #3.5_8 provided the least biased estimates.

Currently, as stated before, it is recommended that users provide starting values that reflect some information about the order of categories – for example, for the highest and the lowest in the estimation of NRM. Also, when default starting values that are embedded in a program are used for the estimation of NRM, it is strongly recommended that users check and compare different set(s) of starting values.[1]

Because "mirt" uses the new reparameterization of NRM ($a^{\star}_{ik}a_i$ rather than the conventional a_{ik}), the command for requesting the standard error for the NRM parameterization in Equation 3.5_1 (i.e., tying "`coef(mod.nrm, IRTpars=T, printSE=T)`") does not work in the current version of "mirt" used in this application.

If one wants to see the CCCs for an item, the first command here shows them for each item. This command produces CCCs for item 1. The second command prints the CCCs for all items in the data set.

```
itemplot(mod.nrm, 1)
plot(mod.nrm, type = "trace")
```

Note

1 Another starting value approach as a stability check of the current set of estimates, which is less burdensome on users, might be to use zeros for all these free parameter starting values. These zero starting values could result in negative overall slope estimates, which may

not be considered sensible in the new parameterization of NRM, but the category slope parameter, a_{ik}, in the traditional NRM parameterization seems to be recovered better than the default starting values in the author's (limited) trials.

References

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, *43*, 561–573.

Bock, R. D. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. *Psychometrika*, *37*, 29–51.

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

Cai, L., & Monro, S. (2014). *A new statistic for evaluating item response theory models for ordinal data*. Los Angeles, CA: National Center for Research on Evaluation, Standards, & Student Testing, Technical Report.

Chalmers, P. (2019). Package 'mirt'. *User Guide*. Retrieved from https://cran.r -project.org/web/packages/mirt/mirt.pdf

de Ayala, R. J. (2009). *The theory and practice of item response theory*. New York, NY: Guilford Press.

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

Maydeu-Olivares, A., & Joe, H. (2006). Limited information goodness-of-fit testing in multidimensional contingency tables. *Psychometrika*, *71*, 713–732.

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

Samejima, F. (1969). *Estimation of latent ability using a response pattern of graded scores* (Psychometric Monograph No. 17). Richmond, VA: Psychometric Society. Retrieved from www.psychometrika.org/journal/online/MN17.pdf

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. 43–76). New York, NY: Routledge Taylor & Francis Group.

Thissen, D., & Steinberg, L. (1986). A taxonomy of item response models. *Psychometrika*, *51*, 567–577.

Wilson, M., & Masters, G. N. (1993). The partial credit model and null categories. *Psychometrika*, *58*, 87–99.

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

Yen, W. M. (1993). Scaling performance assessments: Strategies for managing local item dependence. *Journal of Educational Measurement*, *30*, 187–213.

4

UNIDIMENSIONAL IRT WITH OTHER APPLICATIONS

In this chapter, four IRT applications are illustrated. The first application is for data that consist of both dichotomous and polytomous item responses together. This mixed format response data may result from a psychological test that consists of both yes/no questions and Likert style items. Another example of mixed format data is responses collected from a cognitive test that includes dichotomously scored multiple-choice or true/false items together with constructed-response items scored based on a standardized polytomous scoring rubric.

The second application presented in this chapter is when data are from multiple populations, e.g., when the same or multiple test forms are administered to two (or more) different ability groups. In an MML estimation context, the distinct population differences should be modeled. Concurrent calibration of test score equating in IRT and differential item function (DIF) analysis are introduced under the multiple group IRT application. The third application demonstrates a model estimation when some of the item parameters are fixed and all of the other model parameters are freely estimated, which is another example of test score equating called fixed item parameter calibration (FIPC). The last is an application of IRT latent regression where the outcome variable is a person's latent trait or ability score, which is regressed upon person covariates.

For these applications, we focus on the estimation of unidimensional model item parameters and population ability distribution parameters using the "mirt" package. Details of some functions that have been previously explained are only briefly presented here, and readers are encouraged to refer to the original command explanations for additional instructions.

4.1 Mixed format response IRT modeling

The first scenario assumed here is from a psychological test, where item responses are from yes/no questions and Likert style questions. A reasonable choice is to use

the 2PLM for dichotomous responses from the yes/no questions and either the GRM or GPCM for polytomous responses from the Likert style questions. The "2PLM + GRM" application or "2PLM + GPCM" is possible. The "2PLM + GRM" application is illustrated.

The second scenario of mixed format response modeling is from an educational setting, where a test consists of dichotomously scored items, e.g., multiple-choice or true/false items, and polytomously scored items, e.g., constructed-response items. For this situation, an appropriate model for the dichotomous items may be the 3PLM for dichotomous response items and the GRM or GPCM for the polytomous response items. The "3PLM + GPCM" application is illustrated.

2PLM + GRM calibration

Because the 2PLM is a submodel of the GRM (or GPCM), the estimation of the mixed format data with these models can be done in two ways. One procedure is to specify the 2PLM for the dichotomous items and the GRM separately for the polytomous items. Another way of fitting the mixed format data when the dichotomous model is a submodel of the polytomous model is to specify the polytomous mode, the GRM in this case, to all items. Because the GRM reduces to the 2PLM when the data are dichotomous, the appropriate model is still applied to each item type.

```
install.packages("mirt") #4.1_1
library(mirt) #4.1_2
setwd("c:/mystudy/RIRT/") #4.1_3
ddd<- read.table("c4_2PL_GRM.dat", header=T) #4.1_4
dim(ddd) #4.1_5
apply(ddd, 2, table) #4.1_6
```

(For details on #4.1_1 through #4.1_5, see #2.1_38 and #2.1_39 for the installation of "mirt" and #2.1_4 through #2.1_6 for reading in the data.)

#4.1_1. The "mirt" package is installed.
#4.1_2. Upload the "mirt" package onto the current session.
#4.1_3. Set up a working directory where the folder contains data files and R output files are stored.
#4.1_4. The data file "c4_2PL_GRM.dat" is read into R and stored under the name of "`ddd`" using "`read.table()`." The data file contains the variable names as the first row, so "`header=T`" is used. The items 1 through 5 are dichotomous response items, while items 6 through 8 are polytomous response items. The polytomous items have response options coded as 0, 1, 2, 3, or 4.
#4.1_5. Dimensions of the data set are provided. The number of test-takers is 1200 (which is the number of rows), and the number of items is eight (which is the number of columns).
#4.1_6. The "`apply()`" command shows the frequency of category responses for each item.

```
spec.it <- c(rep("2PL",5),rep("graded",3)) #4.1_7
mod.2plgrm <- mirt(ddd, model=1, itemtype=spec.it, SE=T)
  #4.1_8
mod.2plgrm #4.1_9
extract.mirt(mod.2plgrm, what="converged") #4.1_10
extract.mirt(mod.2plgrm, what="secondordertest") #4.1_11
coef(mod.2plgrm, simplify=T, IRTpars=T) #4.1_12
coef(mod.2plgrm, IRTpars=T, printSE=T) #4.1_13
```

#4.1_7. Because two models are used, i.e., the 2PLM and GRM, the "`spec.it`" object is used to specify the item types. The object is a character string that repeats "`2PL`" five times to specify that the first five items are modeled under the 2PLM, and "`graded`" is repeated three times to specify that the next three times are modeled under the GRM. This object is then used for the required "`itemtype`" argument used in the next "`mirt()`" command. "`spec.it`" contains eight character values ("2PL," "2PL," "2PL," "2PL," "2PL," "graded," "graded," "graded").

#4.1_8. The unidimensional ("`model=1`") mixed model is fit with the "`mirt()`" function. Using the "spect.it" object for the "itemtype" argument, the 2PLM + GRM model is estimated. The resulting calibration is stored as "`mod.2plgrm`."

#4.1_9. Convergence check is done. This prints the final log-likelihood and some information criteria (AIC and BIC).

#4.1_10 and #4.1_11. Separate extractions of the "`mirt()`" convergence check results are printed. The "`TRUE`" output for both indicates no particular estimation issues detected with the program default check setting. See #2.2_30 and #2.2_31.

#4.1_12. Estimated item coefficients are printed with the "`coef()`" function. The slope or discrimination and difficulty parameter estimates of the 2PLM for items 1 through 5 are shown under the "`a`" and "`b`" columns, respectively. "mirt" uses a more general IRF where the lower asymptote ("`g`") and the upper asymptote ("`u`") are parameterized. For the 2PLM, their values are fixed as 0 and 1, respectively.

```
> coef(mod.2plgrm, simplify=T, IRTpars=T)
$'items'
          a      b  g  u     b1     b2    b3    b4
Item1 0.859  1.559  0  1     NA     NA    NA    NA
Item2 1.763 -1.448  0  1     NA     NA    NA    NA
Item3 0.688 -0.296  0  1     NA     NA    NA    NA
Item4 2.374 -0.489  0  1     NA     NA    NA    NA
Item5 1.125  0.426  0  1     NA     NA    NA    NA
```

```
Item6 1.330     NA NA NA -1.630 -0.614 0.492 1.219
Item7 0.913     NA NA NA -0.596 -0.233 0.111 0.392
Item8 2.248     NA NA NA -0.295  0.055 0.590 1.382

$means
F1
 0

$cov
  F1
F1 1
```

The slope and the category boundary parameter estimates of the GRM for items 6, 7, and 8 are shown under the "`a`," "`b1`," "`b2`," "`b3`," and "`b4`" columns, respectively. See also #3.4_35 for the GRM output. The ability population mean and variance are fixed as 0 and 1, respectively, for the model identification purpose.

#4.1_13. The item parameter estimates and their standard errors are both printed. See #2.2_34 for the 2PLM explanation and #3.4_36 for the GRM explanation.

```
> coef(mod.2plgrm, IRTpars=T, printSE=T)
$'Item1'
         a     b  g  u
par  0.859 1.559  0  1
SE   0.099 0.165 NA NA

$Item2
        a      b  g  u
par 1.763 -1.448  0  1
SE  0.182  0.097 NA NA
...

$Item5
        a     b  g  u
par 1.125 0.426  0  1
SE  0.103 0.070 NA NA

$Item6
        a     b1     b2    b3    b4
par 1.330 -1.630 -0.614 0.492 1.219
SE  0.091  0.105  0.065 0.063 0.087
...
```

```
$Item8
        a     b1    b2   b3    b4
par 2.248 -0.295 0.055 0.59 1.382
SE  0.171  0.047 0.045 0.05 0.074

$GroupPars
    MEAN_1 COV_11
par      0      1
SE      NA     NA
```

Another way to estimate the 2PLM+GRM mixed format model in "mirt" is to use the GRM for all items because the 2PLM is a special case of the GRM when response data have only two categories. The command to estimate the 2PLM+GRM becomes simpler, which is shown here.

```
mod.grm<-mirt(ddd, model=1, itemtype="graded", SE=T)
coef(mod.grm, simplify=T, IRTpars=T)
```

The presentation of the estimation results for item parameters for the "`mod.grm`" object is different from those given in #4.1_8 for the "`mod.2plgrm`" object. However, the item parameter estimates from the two procedures are identical. The estimation results for the "`mod.grm`" object are provided.

```
> coef(mod.grm, simplify=T, IRTpars=T)
$'items'
          a     b1     b2    b3    b4
Item1 0.859  1.559     NA    NA    NA
Item2 1.763 -1.448     NA    NA    NA
Item3 0.688 -0.296     NA    NA    NA
Item4 2.374 -0.489     NA    NA    NA
Item5 1.125  0.426     NA    NA    NA
Item6 1.330 -1.630 -0.614 0.492 1.219
Item7 0.913 -0.596 -0.233 0.111 0.392
Item8 2.248 -0.295  0.055 0.590 1.382

$means
F1
 0

$cov
  F1
F1 1
```

The GRM is used for both dichotomous and polytomous response items, so all item parameter estimates are shown under the GRM parameterization. All item slope estimates are shown under the column "`a`." The item difficulty estimates of

the 2PLM items are shown under the column "b1." All item category boundary parameter estimates of the GRM items are shown under the columns of "b1," "b2," "b3," and "b4."

3PLM + GPCM calibration

We assume here that the "mirt" package" is installed and uploaded into the current R session, and the working directory has been set. The commands for these are the same as #4.1_1, #4.1_2, and #4.1_3. We start by reading a data set, "c4_3PL_GPCM.dat."

```
ddd<-read.table("c4_3PL_GPCM.dat") #4.1_13
names(ddd)<-c(paste("MC", 1:10, sep=""), paste("CR",1:5,
  sep="")) #4.1_14
dim(ddd) #4.1_15
apply(ddd,2,table) #4.1_16
table(ddd$CR1) #4.1_17
```

#4.1_13. The data file is ready by "`read.table()`." The data set has ten dichotomously scored items and five polytomously scored items. There are no column names in the file, so the "`header=T`" is not included.

#4.1_14. This line of command is to provide the column labels for the items. The default item (variable) names ("V1" through "V15") are replaced by "MC1," "MC2," . . . , "MC10," "CR1," "CR2," . . . , "CR5." "MC" is used to specify the ten multiple-choice, dichotomously scored items that are calibrated by the 3PLM. "CR" is used to specify the five constructed-response items that are polytomously scored and calibrated by the GPCM. Each CR item has four category response options scored 0, 1, 2, or 3. See also #2.1_8 for creating the item names.

#4.1_15. Dimensions of the data set is shown by "`dim()`." The number of test-takers (rows) is 2500 and the number of items (columns) is 15.

#4.1_16. The frequency analysis of items is conducted using "`apply()`" with "`table()`."

#4.1_17. Users can also extract the frequency information for a particular item using "`table()`." In this example, the option frequency for "CR1" is requested. If the frequency of "MC9" is desired, "`table(ddd$MC9)`" can be used.

In the estimation of the 3PLM+GPCM, the lower asymptote parameter, or pseudo-guessing parameter, in the 3PLM is specified a priori for stable estimations. This prior specification ("`spec`") and the specification of the item types ("`spec.it`") are created before the data are calibrated to the mixed format model. Both of these are used in the "`mirt()`" command.

```
spec<- ' F = 1-15
 PRIOR = (1-10, g, norm, -1.3, 2)' #4.1_18
spec.it<-c(rep("3PL",10),rep("gpcm",5)) #4.1_19
mod.3plgpcm<-mirt(ddd, model=spec, itemtype=spec.it,
  SE=T) #4.1_20
mod.3plgpcm #4.1_21
extract.mirt(mod.3plgpcm, what="converged") #4.1_22
extract.mirt(mod.3plgpcm, what="secondordertest") #4.1_23
coef(mod.3plgpcm, simplify=T, IRTpars=T) #4.1_24
coef(mod.3plgpcm, IRTpars=T, printSE=T) #4.1_25
```

#4.1_18. The first line is to specify a unidimensional model for all 15 items. The "`PRIOR`" distribution for the pseudo-guessing parameter in the 3PLM for all MC items ("`1-10`") is defined on the second line. The current specification uses a normal distribution ("`norm`"), with its mean equal to −1.3 and standard deviation equal to 2 for the logit of the pseudo-guessing parameter ("`g`"). This prior is assuming the pseudo-guessing parameter values are around .2. All other item parameters in the 3PLM and GPCM are estimated without priors. See also #2.4_8.

#4.1_19. Similar to #4.1_7, this is to specify the IRT model that is applied to each item. The ten "MC" items are fit to the 3PLM, while the five "CR" items are fit to the GPCM.

#4.1_20. The two R objects created to specify the prior for the 3PLM pseudo-guessing parameter and the item types for using different models, i.e., "`spec`" and "`spec.it`," respectively, are used in the "`mirt()`" estimation. The calibrations are stored as the object "`mod.3plgpcm`."

#4.1_21. Convergence check results and the fit index DIC are shown. See also #2.4_11.

#4.1_22 and #4.1_23. Separate extractions of the convergence checks are shown. An output of "`TRUE`" in both indicates no particular issue detected in the estimation process under the default setting.

#4.1_24. The item parameter estimates are shown.

```
> coef(mod.3plgpcm, simplify=T, IRTpars=T)
$'items'
         a      b     g  u     b1    b2    b3
MC1  1.618  1.367 0.200  1     NA    NA    NA
MC2  1.954 -0.498 0.038  1     NA    NA    NA
...
MC10 0.987 -0.215 0.076  1     NA    NA    NA
CR1  1.002     NA    NA NA  0.170 0.476 0.851
CR2  1.071     NA    NA NA  0.522 0.879 1.546
...
CR5  1.193     NA    NA NA -0.991 0.219 1.616
```

```
$means
F
0

$cov
  F
F 1
```

The discrimination, difficulty, and pseudo-guessing parameter estimates are shown under the "a," "b," and "g" columns for the 3PLM items. The item slope or discrimination and the item threshold parameter (category-intersection parameter) estimates of the CR items by GPCM are shown under the "a," "b1," "b2," and "b3" columns. See #4.1_12.

#4.1_25. The item parameter estimates and their standard errors ("SE") are shown together. See #4.1_13.

```
> coef(mod.3plgpcm, IRTpars=T, printSE=T)
$'MC1'
        a     b    g  u
par 1.618 1.367 0.20  1
SE  0.238 0.083 0.02 NA
...

$MC10
        a      b     g  u
par 0.987 -0.215 0.076  1
SE  0.104  0.186 0.069 NA

$CR1
        a    b1    b2    b3
par 1.002 0.170 0.476 0.851
SE  0.049 0.099 0.068 0.074
...
$CR5
        a     b1    b2    b3
par 1.193 -0.991 0.219 1.616
SE  0.057  0.118 0.049 0.073

$GroupPars
    MEAN_1 COV_11
par      0      1
SE      NA     NA
```

In both coefficient outputs, the mean and variance of the ability score distribution are 0 and 1, respectively, for model identification.

4.2 Multiple group IRT modeling

When two different groups of test-takers who have different ability distributions (e.g., heterogeneous mean and standard deviations) take a test, these should be modeled to reflect the ability differences in the MML estimation method. For example, when a test is administered to high and low ability examinee groups or different grade levels, the multiple group IRT (Bock & Zimowski, 1997) is a way to handle the group differences. Also, when a researcher wants to investigate group differences such as gender, the multiple group IRT modeling can be useful. The application of multiple group IRT can be extended for the investigation of differential item functioning (DIF) or for test score equating when two heterogeneous groups take different test forms that share a set of common items. A common test score equating situation is where multiple test forms with a set of common items linking the different test forms are administered to multiple heterogeneous groups of test-takers. The application of the multiple group IRT in this occasion is known as concurrent calibration in the test score equating literature.

In this section, three applications of the multiple group IRT are illustrated. The first application is a situation where the same test of multiple-choice items is administered to two different groups that vary in their abilities, and the multiple-choice items are dichotomously scored. The second application is when two different test forms that share a set of common items (or linking or anchor items) are administered to two different ability groups (e.g., different grade levels). This is also known as non-equivalent groups anchor test (NEAT) data collection design. A third application presented is to investigate DIF, where the same set of items is presented to two different groups, and the functioning of a specific item is investigated when the groups' performance on the other items is matched. "mirt" is used to illustrate the multiple group IRT modeling.

A test administered to two different ability groups

In this scenario, the same 35 dichotomously scored multiple-choice (MC) item responses are collected from two different ability groups. We start with the assumption that the installation of "mirt," the uploading of "mirt," and the setting up of a working directory are done. See #4.1_1, #4.1_2, and #4.1_3.

```
ddd<-read.table("c4_MGIRT1.dat") #4.2_1
GID<-as.character(ddd[,36]) #4.2_2
ddd<-ddd[,-36] #4.2_3
names(ddd)<-paste("MC", 1:35, sep="") #4.2_4
```

#4.2_1. The "`read.table()`" function is used to read in the "`c4_MGIRT1.dat`," which contains 5100 test-takers' responses to 35 dichotomous items. The variable after the last 35th item is a group indicator variable, "G1" and "G2" for group 1 and group 2, respectively. The data file has a space-delimited

format for which "`read.table()`" is effective to read the data. The R object "ddd" created by "`read.table()`" is called a data frame. The default variable names are "V1," "V2," . . . , "V36." The number of test-takers for group 1 is 2500 and the number of test-takers for group 2 is 2600. This can be shown using "`table(ddd$V36)`."

#4.2_2. In "ddd," the group indicator variable ("V36") is read in as a factor (categorical variable). Readers can use the "`head(ddd)`" function to observe the first six rows of the data, where columns 1 through 35 contain item responses and column 36 contains a group identification. By default, when the "`read.table()`" function is used, columns containing only numbers are stored as integers, and columns that contain a categorical grouping are stored as factors. This can be viewed using "`class(ddd[,1])`" or "`class(ddd[,36])`." Here, the 36th variable ("V36"), which is the group variable, is extracted from "ddd" and stored as a character variable using "`as.character()`" named "GID." A character-type variable is required for the group indicator specification for multiple group IRT in "mirt."

#4.2_3. For simplicity, the last group indicator variable in "ddd" is removed from "ddd," so that "ddd" contains only the item response data.

#4.2_4. The new item labels, "MC1," "MC2," . . . , "MC35," are created and replace the default variable names. See #2.1_8.

```
spec<- ' F = 1-35
 PRIOR = (1-35, g, norm, -1.5, 2)' #4.2_5
spec.inv<-c(names(ddd), "free_var","free_means") #4.2_6
mod.mgi<-multipleGroup(ddd, model=spec, group=GID,
  invariance=spec.inv,
itemtype="3PL", SE=T) #4.2_7
mod.mgi #4.2_8
extract.mirt(mod.mgi, what="converged") #4.2_9
extract.mirt(mod.mgi, what="secondordertest") #4.2_10
coef(mod.mgi, simplify=T, IRTpars=T) #4.2_11
coef(mod.mgi, printSE=T) #4.2_12
```

#4.2_5. The 3PLM is specified for all items. The prior distribution of the logit of the lower asymptote parameter, "g," in the 3PLM is specified to follow a normal distribution with a mean of −1.5 and a standard deviation of 2. Setting the prior mean as −1.5 means that the lower asymptote parameter values are expected to be around 0.18 on average. #See 2.4_8.

#4.2_6. Another specification needed in order to run the multiple group IRT model is the specific items whose parameters are invariant across groups. The "`names(ddd)`" calls the names of all the items to specify that all items and their parameters (slope, intercept, and lower asymptote parameters) are invariant or equal across the two groups. The mean and the variance of the

ability distribution of the second group (group 2, "G2") are free parameters (using "`free_means`" and "`free_var`"), and the mean and variance of the first group (group 1, "G1") or the reference group are fixed as 0 and 1, respectively.

A less efficient specification for "spec.inv" is "`spec.inv<-c("MC1", "MC2", "MC3", ..., "MC35""free_var,""free_means")`." The "`name(ddd)`" is used to extract the item names in a more efficient manner. The last two arguments, "`free_var`" and "`free_means`," are for specifying the second group's freely estimated ability distribution mean and variance.

#4.2_7. "`multipleGroup()`" is used to perform the multiple group analysis. For the "`model`" argument, the previously defined "spec" (#4.2_5) is used. The character variable group indicator "`GID`" (#4.2_2) is used for the "`group`" argument. "`speci.inv`" from #4.2_6 is used for the "`invariance`" argument. The "`itemtype`" is specified to the "`3PL`." The standard errors of the parameters are also estimated with "`SE=T`." The results are stored in the object "`mod.mgi`." For more information on this function use "`?multipleGroup`."

#4.2_8. Convergence check and DIC results are shown. See also #2.4_11.

#4.2_9 and #4.2_10. Two convergence check results are extracted separately. "`TRUE`" for both indicates no specific issues detected in the estimation process.

#4.2_11. Estimates of the item parameters for the 35 items and the freely estimated population ability distribution parameters (mean and variance) of the second group ("`G2`") are shown. (The ability distribution parameters of the first group ["`G1`"] were fixed to 0 and 1 for the mean and variance, respectively.) The presentation of the estimates is shown for group 1 ("`G1`") and for group 2 ("`G2`") separately, but the values are equivalent since all item names were include in "`speci.inv.`" Because the same test is administered, the item parameter estimates are assumed to be the same for the two groups. (Precisely speaking, in this application of the multiple group IRT modeling, item parameter invariance or no DIF for the same items is assumed across groups.)

```
> coef(mod.mgi, simplify=T, IRTpars=T)
$'G1'
$'items'
          a      b     g  u
MC1   1.511 -1.070 0.138  1
MC2   0.930 -0.295 0.271  1
...
MC35 1.469 -1.787 0.216  1

$means
F
0
```

```
$cov
  F
F 1

$G2
$'items'
         a      b     g  u
MC1  1.511 -1.070 0.138  1
MC2  0.930 -0.295 0.271  1
MC3  0.804  1.325 0.210  1
...
MC35 1.469 -1.787 0.216  1

$means
    F
0.526

$cov
      F
F 1.395
```

For the second group, the estimated population mean is 0.526, and its variance estimate is 1.395. The first group mean and variance are fixed as 0 and 1, respectively. Thus, the second group has a higher ability, on average, and a larger variation in ability among test-takers than the first group.

#4.2_12. The current "`multipleGroup()`" function in "mirt" does not provide the standard errors for the usual parameterization for the item discrimination (a_i), difficulty (b_i), and pseudo-guessing (g_i) parameters. Instead, it uses the slope and intercept form (a_i and d_i, respectively) with the logit of the pseudo-guessing parameter (logit(g_i)). The command line shown here provides the standard error of item slope, intercept, and logit of the pseudo-guessing parameter.

Two test forms administered to two different ability groups

In this scenario, two different test forms, say, Form A and Form B, are administered to group 1 and group 2, respectively, who are also different in their abilities. Forms A and B have a subset of common items that are the same across the two forms, and the remaining items to each form are unique. The set of common items is called the anchor or linking set of items. This design is called a non-equivalent anchor test design, or NEAT. In a large-scale assessment program, this kind of multiple test form administration to different ability groups (e.g., different grades such as grade 4 and grade 5) is not uncommon, especially in the context of, what is called, vertical scaling.

With the assumption of the item parameter invariance for the common items (or equivalently, no DIF for the linking items), a concurrent estimation of all model

parameters (including ability parameters) is one of the linking or equating methods to calibrate the data set, establishing a common scale between different test-taker groups and between different test forms. Readers are referred to Kolen and Brennan (2014) for more information on what the equating, the linking, and the scaling are, and for other equating methods for this NEAT design and other data collection designs.

Again, we start with the assumption that "mirt" is installed and loaded onto the R session, and a working directory is set (#4.1_1, #4.1_2, and #4.1_3). The commands to conduct the concurrent calibration in the two different test forms and the two heterogeneous groups of test-takers are essentially the same, as the previous multiple group IRT application for a single test form administered to two different ability groups. The key difference is in the data preparation.

```
ddd<-read.table("c4_MGIRT2.dat") #4.2_13
dim(ddd) #4.2_14
GID<-as.character(ddd[,71]) #4.2_15
ddd<-ddd[,-71] #4.2_16
ddd[1:4000,41:70]<-NA #4.2_17
ddd[4001:8000,1:30]<-NA #4.2_18
```

#4.2_13. The dichotomously scored item response data, "c4_MGIRT2.dat," which has a space-delimited format between variables, is read and stored as a data frame R object, "`ddd`." The total number of items in the data set is 70, and the last column is a group indicator variable. "G1" indicates group 1, and "G2" indicates group 2. By default, the variable names are "V1," "V2," . . . , "V70," and "V71." Also, the group indicator variable ("V71") is recognized as factor (categorical variable) by default. To check the names of the variables in the data set "ddd," use "`names(ddd)`."

There are two groups, each of which has 4000 test-takers (shown with "`table(ddd$V71)`"). The 70 items are from the two test forms: Form A (40 items) and Form B (40 items). Forms A and B share a set of ten common items, to which all test-takers have responses. The number of unique items across the two forms is 60, where each item is taken by test-takers in only one of the two groups.

The data file consists of two big clusters. One cluster of data is from group 1 taking Form A (from the first row to the 4000th row in the data file), and the other is from group 2 taking Form B (from the 4001st row to the last 8000th row in the data file). In addition to the dichotomous item responses of 1 and 0, the value 9 exists to denote a missing response. (Later, the 9 is changed to NA.) Note that in this NEAT design, those 9s are systematically missing by the data collection design, and they do not pose problems in the calibration of multiple IRT model when data are collected under the usual standardized test setting.

Group 1 (having 4000 test-takers) took Form A; thus, dichotomous responses (0 or 1) are present in columns 1 through 40 (corresponding to items 1 through 40),

and 9s are present in the 41st through 70th columns. Group 2 (having 4000 test-takers) took Form B and its test-takers' responses are arranged from the 31st to the 70th item response column, and the 1st through the 30th columns are filled with 9s. The item responses from the 31st to 40th columns are those from the common linking or anchor items.

#4.2_14. Dimension of the data set "ddd." The total number of test-takers from both groups, or the number of rows, is 8000. The total number of columns is 71, which contains 70 items and the group indicator variable.

#4.2_15. As in the previous application of the multiple group IRT, the group indicator in "mirt" should be a character variable; thus, this command saved the last variable in the data ("V71") as "GID" and converts it to a character variable. See #4.2_2.

#4.2_16. For simplicity, the group indicator variable is removed from "ddd" so that "ddd" contains only item response data.

#4.2_17 and #4.2_18. In R, the missing data are denoted by "NA." For "mirt" to recognize what the missing data are, "NA" replaces the value of 9 (missing data code) in the data file. #4.2_17 replaces the first group's missing data to items 41 through 70 to "NA"; #4.2_18 replaces the second group's missing data to items 1 through 30 to "NA."

The model parameter estimation follows the same syntax used in the previous multiple group IRT application (#4.2_5). Here responses are assumed to be from multiple-choice item tests; thus, the 3PLM is adopted and its pseudo-guessing parameter is estimated by employing a prior distribution for the logit of the pseudo-guessing parameter, i.e., $logit(g_i) \sim N(-1.5,3^2)$. Specifying the mean equal to 1.5 in the prior distribution means that the pseudo-guessing parameter values are thought to be around 0.18. The value of the standard deviation in this prior distribution is specified as 3, which is larger than the previous 3PLM application example. A larger value of standard deviation in the prior distribution leads to less shrinkage of the estimates compared with a smaller standard deviation in the prior distribution.

```
spec<- ' F = 1-70
         PRIOR = (1-70, g, norm, -1.5, 3)' #4.2_19
spec.inv<-c(names(ddd), "free_var", "free_means") #4.2_20
mod.mgi<-multipleGroup(ddd, model=spec, group=GID,
invariance=spec.inv, itemtype="3PL", SE=T) #4.2_21
mod.mgi #4.2_22
coef(mod.mgi, simplify=T, IRTpars=T) #4.2_23
```

#4.2_19. The unidimensional 3PLM item loading structure and the prior distribution for the logit of the pseudo-guessing parameter are specified. See #4.2_5.

#4.2_20. The specification of invariant items (item 1 through item 70) and free parameters (the second group mean and variance) are defined. The first group mean and variance are fixed as 0 and 1, respectively. See #4.2_6.

#4.2_21. The three R objects "`spec`," "`spec.inv`," and "GID," are used in the estimation using the "`multipleGroup()`" function. See #4.2_7.

#4.2_22. Convergence results under the default check environment are provided. Because the 3PLM is estimated with the pseudo-guessing parameter prior distribution, the final optimization function value is shown under "`Log-posterior`" and a descriptive model index "`DIC`," which is a Bayesian analogue of AIC, is printed. See #2.4_11.

#4.2_23. The estimated item and the population parameters are shown.

```
> coef(mod.mgi, simplify=T, IRTpars=T)
$'G1'
$'items'
        a      b     g  u
V1  1.332  0.488 0.127  1
V2  1.220 -0.715 0.234  1
...
V70 1.434  0.293 0.040  1

$means
F
0

$cov
  F
F 1

$G2
$'items'
        a      b     g  u
V1  1.332  0.488 0.127  1
V2  1.220 -0.715 0.234  1
...
V70 1.434  0.293 0.040  1

$means
    F
0.266

$cov
      F
F 1.042
```

The item parameter estimates of the 70 items from both Forms A and B are shown for group 1 ("`G1`") and group 2 ("`G2`") separately. Item 1 through item 40 ("V1"

through "V40") are for Form A. Item 31 through 70 ("V31" through "V70") are for Form B. The anchor (common) item estimates are those of item 31 through 40 ("V31" through "V40"). The first group ("`G1`") mean and variance are fixed as 0 and 1, respectively. The second group ("`G2`") mean and variances were estimated to be 0.266 and 1.042, respectively, indicating that the second group has a higher overall average ability and slightly more variability than the first group's ability distribution.

The request of the standard errors for the conventional parameterization of the discrimination (a_i), difficulty (b_i), and pseudo-guessing (g_i) parameter estimates are not available in the current version of "mirt." Thus, using "`coef(mod.mgi, IRTpars=T, printSE=T)`" does not produce the standard errors, but "`coef(mod.mgi, printSE=T)`" produces standard errors for the estimates of the parameterization used for the actual estimation, which are slope (a_i), intercept (d_i), and logit of the pseudo-guessing, logit(g_i).

For person ability score estimation, the "`fscores()`" command is used, similar to the previous chapters (see #2.4_19). For the EAP ability estimator, the following command is used.

```
fscores(mod.mgi, method="EAP", full.scores=T, full.scores.SE = T)
```

Because this is the multiple group IRT application where two different group ability distributions are modeled, the maximum likelihood (ML) estimator may be preferred, in which case the ML ability estimates are obtained by "`fscores(mod.mgi, method="ML")`." Note again that the ML estimator does not have finite estimates for the 0 and perfect scores; thus, negative infinity ("`-Inf`") and positive infinity ("`Inf`") are shown for those scores. Using "`fscores(mod.mgi, method="ML", full.scores=T, full.scores.SE = T)`" appears to fail to properly print the standard errors in the current "mirt" version.

Application to differential item functioning

Differential item functioning (DIF) is a statistical procedure to detect differential performance between (typically) two groups on an item when the groups are equated at the same ability level. If the difference in the performance between the two compared groups is 0, when controlling for their abilities, there is no DIF (see, e.g., Paek (2018) for the concept of DIF and its origin). There are many methods to detect DIF.

Here an application of multiple group IRT using the likelihood ratio approach for dichotomous item response data is illustrated for the investigation of an item for its DIF. For a more thorough treatment of DIF and its detection methods, readers are referred to Holland and Thayer (1988) and Holland and Wainer (1993). We start with reading the data file, assuming that "mirt" is installed and uploaded onto the current R session, and a working directory is set up (#4.1_1, #4.1_2, and #4.1_3).

```
ddd<-read.table("c4_MGIRT3.dat", header=T) #4.2_24
dim(ddd) #4.2_25
table(ddd$GID) #4.2_26
names(ddd) #4.2_27
GID<-as.character(ddd$GID) #4.2_28
ddd<-ddd[,-21] #4.2_29
```

#4.2_24. The data file is "c4_MGIRT3.dat." The space-delimited format file is read in by "`read.table()`." The first row in the file contains the variable names (20 items, "`it1`", "`it2`", . . . "`it20`," and a group indicator variable, "`GID`"). Reading the variable names is enabled by adding the option "`header=T`." Item responses and a character value for the group indicator begin on the second row. The last column is a group indicator variable ("`GID`") with the value "`g1`" for group 1, or "`g2`" for group 2.

#4.2_25. Dimensions of the data object "`ddd`" are shown. The number of test-takers is 1300 and the number of column variables is 21. The items are the first 20 columns, and the group indicator variable is in column 21.

#4.2_26. The "`GID`" variable indicates the group assignment, either "`g1`" or "`g2`." The "`table()`" function is used to show the count of observations within each level of the variable. "`g1`" has 1000 test-takers and "`g2`" has 300 test-takers. In DIF analysis, it is not uncommon to observe unbalanced sample sizes for the two compared groups. The focal group, which is typically a minority group, tends to have a smaller sample size than the reference group when doing DIF analysis.

#4.2_27. This command shows the variable names in "`ddd`." Names for the items are "`it1`", "`it2`", ..., "`it20`," and the group indicator variable name is "`GID.`"

#4.2_28. The group indicator variable is extracted and converted into a character. By default, "`read.table()`" converts the "`GID`" variable to a factor. In order to use the identifier in multiple group IRT in "`mirt`," the variable should be converted to a character string variable. See #4.2_15.

#4.2_29. The 21st variable in "`ddd`," which is "`GID`" is removed so that "`ddd`" contains only the item response data.

To conduct DIF analysis using the likelihood ratio test, a DIF model, which allows the suspected DIF item parameters to be unequal in their estimations, and a non-DIF model, where all item parameters are invariant, or equal, across groups, are fit to data. Then the two models are compared. This is equivalent to comparing the item parameter differences between the two groups. The null hypothesis assumes no DIF, where the item parameters between two groups are equal.

The illustration here examines the last item (item 20) for DIF using the 2PLM multiple group IRT modeling. If the null hypothesis holds and there is no DIF, the item parameter estimates of item 20 would be equal for the two groups, after controlling for the latent trait or ability θ. If item 20 does display DIF, the item

parameter estimates of item 20 would be different for the two groups. The No-DIF model is fit first, with all item parameters considered as invariant.

```
spec.inv<-c(names(ddd), "free_var","free_means") #4.2_30
mod.NoDIF<-multipleGroup(ddd, model=1, group=GID,
invariance=spec.inv,
itemtype="2PL", SE=T) #4.2_31
mod.NoDIF #4.2_32
coef(mod.NoDIF, simplify=T, IRTpars=T) #4.2_33
```

#4.2_30. "`spec.inv`" is created to specify the invariance of all items. "`names(ddd)`" contains all item names in "ddd," and "`free_var`" and "`free_means`" specify the mean and variance of the second group ("`g2`") ability distribution to be free parameters. See #4.2_6.

#4.2_31. The No-DIF model is estimated. "`model=1`" indicates a unidimensional model is used. "`GID`" provides the group information. "`spec.inv`" created in the previous step is used to provide information on which all item parameters are treated invariant across groups. #4.2_7.

#4.2_32. Convergence check results are shown. There is no particular sign of problems in this application.

#4.2_33. The estimated item coefficients are printed. Because of the specified invariance, the item parameter estimates are the same across the two groups. The mean and variance of the first group ("`g1`") were fixed to be 0 and 1, while the mean and variance of the second group ("`g2`") were freely estimated. The estimates of the mean (–0.446) and variance (0.897) show that the average ability of the second group is lower than that of the first group, and its variability is slightly less.

```
spec.inv<-c(names(ddd)[-20], 'free_var','free_means')
#4.2_34
mod.DIF<-multipleGroup(ddd, model=1, group=GID,
invariance=spec.inv,
itemtype="2PL", SE=T) #4.2_35
mod.DIF #4.2_36
coef(mod.DIF, simplify=T, IRTpars=T) #4.2_37
anova(mod.NoDIF, mod.DIF) #4.2_38
```

#4.2_34. Next, the invariance specification of the DIF model is defined. This line specifies a DIF model where the studied item's parameters (i.e., item 20) are freely estimated for each of the two groups, while the other items (i.e., items 1 through 19) are treated as invariant across two groups. "`names(ddd)[-20]`" extracts the names of the items that are assumed as invariant, all except the 20th item. Item parameters of these items are specified to be equal for both groups. Because item 20 in not included, the item parameters

of item 20 for each of the two groups are freely estimated. Also, the second group's ability mean and variance are freely estimated.

#4.2_35. The multiple group model is fit according to the group indicator ("`GID,`" defined in #4.2_28), and the specified invariance of the 19 items ("`spec.inv,`" defined in #4.2_34), similar to #4.2_31 with the modified invariance specification.

#4.2_36. Convergence check results are shown. No issue in this application was detected.

#4.2_37. Estimated item parameters and the second group's ability estimated mean and variance are shown.

```
> coef(mod.DIF, simplify=T, IRTpars=T)
$'g1'
$'items'
          a      b g u
it1   1.218  0.956 0 1
it2   1.015  0.726 0 1
...
it20 0.978 -0.460 0 1

$means
F1
 0

$cov
   F1
F1 1

$g2
$'items'
          a     b g u
it1   1.218 0.956 0 1
it2   1.015 0.726 0 1
...
it20 0.985 0.027 0 1

$means
    F1
-0.428

$cov
      F1
F1 0.894
```

Item 1 through item 19 have the same estimates for both groups, "`g1`" and "`g2`." The studied item 20 has different item parameter estimates for the two groups. The estimated discrimination and difficulty for item 20 from group 1 are 0.978

and –0.460, respectively. The estimated item 20 discrimination and difficulty for group 2 are 0.985 and 0.027. The item response functions (which models the probability of a correct response) for the two groups have somewhat similar estimated discrimination, or slope parameters (i.e., 0.978 for group 1 and 0.985 for group 2). However, the functions vary clearly in their estimated difficulty for item 20. The item difficulty for group 2 (0.027) is much higher than that for group 1 (–0.460), which means that the probability of a test-taker from group 1 with ability θ correctly responding to item 20 is higher than that of a test-taker from group 2 with the same θ ability. This can be observed using "`itemplot(mod.DIF, 20)`," which displays the ICCs of item 20 for the two groups.

The difference in the difficulty estimates between group 2 and group 1 is $0.027 - (-0.46) = 0.487$. In addition, the overall ability of group 2 (–0.428) is lower than that of group 1 (fixed as 0).

#4.2_38. To test the observed difference in the slope and the difficulty between two groups, the No-DIF (reduced model) and DIF (full model) models are compared using the log-likelihood ratio test with the "`anova()`" function.

```
> anova(mod.NoDIF, mod.DIF)
Model 1: multipleGroup(data = ddd, model = 1, group = GID, invariance =
spec.inv, itemtype = "2PL", SE = T)
Model 2: multipleGroup(data = ddd, model = 1, group = GID, invariance =
spec.inv, itemtype = "2PL", SE = T)
       AIC     AICc    SABIC       HQ      BIC    logLik     X2  df      p
1 26689.32 26692.19 26773.05 26770.79 26906.46 -13302.66    NaN NaN    NaN
2 26683.01 26686.17 26770.73 26768.37 26910.50 -13297.51 10.303   2  0.006
```

The last column, "p," reports the *p*-value of the test. The *p*-value is less than 0.01. The No-DIF model (null hypothesis) is rejected; thus, concluding that the differences in item parameter estimates for item 20 are statistically significant. This is supportive of the presence of DIF for item 20 between the two groups.

Previously in #4.2_37, we observed that the slope estimates were similar while difficulty parameter estimates were very different. One may wonder if the equal slope assumption is tenable, and if the DIF observed is due to the difficulty difference. For this purpose, a model where the equal slope assumption is imposed, and the difficulty is freely estimated for each of the two group is fit. Then this model can be compared with the No-DIF model.

```
spec<-'F = 1-20
CONSTRAINB =(20, a1)' #4.2_39
spec.inv<-c(names(ddd)[-20], 'free_var','free_means')
#4.2_40
mod.EsDIF<-multipleGroup(ddd, model=spec, group=GID,
invariance=spec.inv, itemtype="2PL", SE=T) #4.2_41
anova(mod.EsDIF,mod.DIF) #4.2_42
```

#4.2_39. This specifies that 20 items measure a single construct ("`F`") and the studied item, item 20, has a constraint that the slope parameters are equal between the two groups.

#4.2_40. The item parameters of all items except the studied item (i.e., item 20) are specified as invariant across groups. Also, the second group's ability mean and variance are treated as free parameters. This is similar to #4.2_34.

#4.2_41. The equal slope constraint model for the studied item ("`mod.EsDIF`") is estimated using the group indicator ("`GID`," #4.2_28), the specified model with equal slopes for item 20 ("`spec`," #4.2_39), and the invariance specification of items 1 through 19 ("`spec.inv,`" #4.2_40) created in the previous steps.

#4.2.42. Comparing the equal slope constraint DIF model with the No-DIF model is done by the log-likelihood ratio test. The format of the presentation is the same as #4.2_38. The *p*-value of the test is 0.976, and the equal slope constrained DIF model is retained. Thus, the observation regarding similar slope estimates for item 20 between the two groups in the DIF model estimation results is supported, and the DIF of item 20 is due to the differences in the item's estimated difficulty.

It is noted that "mirt" provides a separate function called "`DIF()`." This special function is another way to conduct DIF analysis, which is not covered here.

4.3 Fixed item parameter calibration

Previously, a concurrent calibration procedure was illustrated where the multiple group IRT calibration was conducted to create a common scale when different test forms are administered that share a set of common items (or anchor items). In this section, an IRT model estimation with a fixed set of item parameter values is illustrated. Unlike the concurrent calibration, this fixed item parameter calibration (FIPC) assumes the values of the common items are known from a previous separate test calibration. As in the previous multiple group concurrent estimation, the same item parameter invariance for the common items is assumed in FIPC. The FIPC is a straightforward procedure to establish a common scale between two different test forms when a set of common items between the two test forms exists. The number of common items between the two test forms could be small (e.g., 10% to 20% of the items in each test), nearly half of a test, or almost all items between the two test forms. The FIPC approach is also used for test equating with an item bank approach. See Kolen and Brennan (2014) for other alternatives to achieve a common scale between two test administrations or multiple test forms using IRT. The illustration here is made with the 3PLM calibration of dichotomous response data using the "`mirt`" package.

Suppose that a 5th grade math test having 40 multiple-choice items was administered to a random sample of a few thousand 5th grade students in a county in the beginning of a fall semester. At the end of the fall semester, a different math

test having the same number of multiple-choice items was administered to another random sample of a few thousand 5th grade students in the same county. The end-of-fall-semester test contains a set of common items (four anchor items) with the beginning-of-fall-semester math test. The values of the four common item parameters obtained in the first test calibration are fixed in the calibration of the end-of-fall-semester test calibration, while all the unique item parameters and the person ability population parameters are freely estimated.

```
ddd<-read.table("c4_FIP_3PL.dat", header=T) #4.3_1
dim(ddd) #4.3_2
```

#4.3_1. An item response data file "`c4_FIP_3PL.dat`" is read into R. In the provided situation, this is the item response data of the end-of-fall-semester test. The data file has the space-delimited format. The first row contains variable or item names (e.g., "MC1," "MC2," . . . "MC40") for the 40 items. Using the "read.table()" command, the header and item responses are easily read and stored in R as an object named "`ddd`."

#4.3_2. This command shows the number of rows (test-takers) and the number of columns (items). The data set, "`ddd`," has 2000 test-takers and 40 items. The common items are items 8, 16, 25, and 32.

```
anchor.adg<-read.table("c4_anchor_adg.dat")
```

The four anchor items' parameter information (items 8, 16, 25, and 32) is read from "`c4_anchor_adg.dat`" and saved as "`anchor.adg`." In the situation of 5th grade testing, these are the estimated item parameters of the four anchor items from the beginning-of-fall-semester test. This file is not used in the FIPC analysis, but the item parameter estimates are observed and used to specify the model.

The "`mirt`" program uses the slope and intercept parameterization, i.e., $a_i\theta + d_i$. Thus, the provision of the item parameter should be the intercept value, not the difficulty parameterization value. The R object "`anchor.adg`" shows the item number, slope, intercept, and pseudo-guessing parameter values of the anchor items.

```
> anchor.adg
  Item.No     a      d     g
1       8 1.146 -0.433 0.233
2      16 1.168 -0.630 0.161
3      24 1.133  1.059 0.176
4      32 0.997 -1.063 0.227
```

The intercept parameter can be converted to the more commonly known and used difficulty parameter ($b_i = -d_i/a_i$). (This is not required for fitting the model, but it is useful when observing the results for comparison purposes.) The resulting

parameters of the anchor items are shown here. The calculations of the difficulty can be done using "`(-1) * anchor.adg$d/anchor.adg$a`."

```
   Item.No     a      b     g
 8       8 1.146  0.378 0.233
16      16 1.168  0.539 0.161
24      24 1.133 -0.935 0.176
32      32 0.997  1.066 0.227
```

The values of "a," "d," and "g" for the four anchor items are fixed in the calibration of the 40-item test.

```
spec<-'F = 1-40
 START=(8,a1,1.146),(8,d, -0.433),(8,g,0.233)
 FIXED=(8,a1),(8,d),(8,g)
 START=(16,a1,1.168),(16,d, -0.630),(16,g,0.161)
 FIXED=(16,a1),(16,d),(16,g)
 START=(24,a1,1.133),(24,d, 1.059),(24,g,0.176)
 FIXED=(24,a1),(24,d),(24,g)
 START=(32,a1,0.997),(32,d, -1.063),(32,g,0.227)
 FIXED=(32,a1),(32,d),(32,g)
 FREE = (GROUP, MEAN_1)
 FREE = (GROUP, COV_11)
 PRIOR = (1-7,9-15,17-23,25-31,33-40, g, norm, -1.5, 4)' #4.3_3
mod.fip3pl<-mirt(ddd, model=spec, itemtype="3PL", SE=T) #4.3_4
mod.fip3pl #4.3_5
coef(mod.fip3pl, simplify=T, IRTpars=T) #4.3_6
coef(mod.fip3pl, IRTpars=T, printSE=T) #4.3_7
```

#4.3_3. The parameter values of the four anchor items are fixed using the "`START`" and "`FIXED`" commands. First, a unidimensional model is defined by specifying that items 1 through 40 load onto a single construct denoted by "`F`." The "`START`" command provides the starting value of the anchor items' slope ("`a1`"), intercept ("`d`"), and pseudo-guessing ("`g`") parameters, which are item parameters provided in the "`anchor.adg`" file from the beginning-of-fall-semester test calibration. The "`FIXED`" command fixes the estimates of the slope, intercept, and pseudo-guessing parameters of the four items to be equal to the previously defined starting values.

The ability population parameter mean and variance are specified as free parameters by the "`FREE`" commands. Lastly, the prior for the logit of the pseudo-guessing parameter for the unique (non-common) items is specified, which follows a normal distribution with a mean equal to −1.5 and standard deviation equal to 4. The specification of the mean of the normal prior for the logit of the pseudo-guessing parameter as −1.5 means the pseudo-guessing parameter values are around 0.18.

#4.3_4. The 3PLM parameters are estimated with the fixed parameter values for the anchor items as defined in "`spec`," and saved under the R object name, "`mod.fip3pl`."

#4.3_5. The convergence check results are printed. There are no particular issues found in this application.

#4.3_6. The point estimates of the model parameters are shown.

```
> coef(mod.fip3pl, simplify=T, IRTpars=T)
$'items'
         a     b     g u
MC1  0.992 0.810 0.178 1
MC2  1.688 0.517 0.259 1
...
MC8  1.146 0.378 0.233 1
...
MC16 1.168 0.539 0.161 1
...
MC40 1.419 1.598 0.222 1

$means
    F
0.197

$cov
      F
F 1.002
```

The parameters of the anchor items (items 8, 16, 24, and 32) were fixed based on the pre-defined specification. For example, the values of the parameters for item 8 are 1.146, 0.378, and 0.233 for the item slope, difficulty, and pseudo-guessing parameters, respectively. They are the fixed anchor item parameter values shown previously. All the other item and population distribution parameters are estimated. The ability population mean and variance are estimated to be 0.197 and 1.002, respectively, in this application.

#4.3_7. Both point estimates and their standard errors are shown.

```
> coef(mod.fip3pl, IRTpars=T, printSE=T)
$'MC1'
        a     b     g  u
par 0.992 0.810 0.178  1
SE  0.186 0.208 0.068 NA
...
$MC8
        a     b     g  u
par 1.146 0.378 0.233  1
SE     NA    NA    NA NA
...
```

```
$MC40
        a     b     g  u
par 1.419 1.598 0.222  1
SE  0.281 0.117 0.029 NA

$GroupPars
    MEAN_1 COV_11
par  0.197  1.002
SE   0.037  0.085
```

For the anchor item parameters, because their values are fixed, "NA" is printed for the standard error.

```
fscores(mod.fip3pl, method="EAP", full.scores=T, full.
  scores.SE = T) #4.3_8
```

#4.3_8. The person latent trait, or ability, is estimated as usual using the "`fscores()`" command. The first column ("F1") is for the EAP ability estimates, and the second column ("SE_F1") is the standard errors of the ability estimates.

4.4 Latent regression

Latent regression in IRT is an extension of an IRT model where person level covariates are included, and their effects on the latent trait(s) are estimated simultaneously with the item parameters. The latent regression with IRT implements a regression of the ability (or latent trait) on the observed variables (e.g., gender, social-economic-status, pre-test score, etc.). The model estimates not only the item parameters, but also the population regression coefficients directly from the data. If a single observed covariate is a categorical or dummy variable, one may also use the multiple group IRT modeling illustrated in section 4.2. However, the latent regression modeling allows other continuous covariates in addition to categorical variables, as in the case of usual regression modeling. An IRT model with the latent regression could be represented in a few ways. Here, a hierarchical model presentation is provided.

Suppose a test with 20 items was administered to 1500 test-takers. The responses were dichotomously scored (e.g., 0 = incorrect/no and 1 = correct/yes). There were two distinct groups of test-takers who were assumed to have different ability levels (e.g., male and female). A researcher would like to investigate gender differences controlling for another observed covariate, e.g., a standardized test score. The latent trait regression may be a choice for this purpose. The illustration of latent regression is made with dichotomous item response data following unidimensionality and two observed covariates where one is a group indicator (or dummy) variable (coded 0 and 1) and the other is a standardized continuous covariate (e.g., a standardized test score).

The "`mirt`" package is used for the illustration here in which the 2PLM is employed. It should be noted that the "`mirt`" package provides a larger model framework that allows fixed and random effects modeling for multiple person abilities (i.e., multidimensionality) and the item parameters in several IRT models (e.g., Rasch, 2PLM, 3PLM, 4PLM, and models for polytomous response data). The latent regression modeling in IRT is a subcase of the extended mixed-effects IRT modeling. See Chalmers (2015) and De Boeck and Wilson (2004) for more details on this extended or explanatory IRT modeling.

The hierarchical presentation of the unidimensional 2PLM latent regression is

$$P\left(X_{ij} = 1 | \theta_j\right) = \frac{\exp\left(a_i\theta_j + d_i\right)}{1 + \exp\left(a_i\theta_j + d_i\right)}, \text{ and} \tag{4.4_1}$$

$$\theta_j = \beta_0 + \sum_p \beta_p W_{jp} + \epsilon_j. \tag{4.4_2}$$

All of the terms in Equation 4.4_1 are defined the same as before (refer to #2.2_5). The person latent trait, θ, is regressed upon P number of observed person level observed covariates ($p = 1, 2, \ldots, P$). W_{jp} is the pth covariate, β_0 is the intercept, β_p is the slope coefficient, and ϵ_j is the jth person residual, which is assumed to follow a normal distribution with a mean of 0 and a variance of σ_ϵ^2. This variance is a conditional variance of θ when there exists some W_{jp} in the model. The latent regression model shown earlier is reduced to the regular 2PLM when there are no W_{jp} covariates. The model identification constraints in the latent regression are $\beta_0 = 0$ and $\sigma_\epsilon^2 = 1$, which is the "`mirt`" default when using the 2PLM latent regression. An alternative model identification constraint may be to set, e.g., $a_1 = 1$ and $\beta_0 = 0$, so that σ_ϵ^2 becomes a free parameter.

The IRT latent regression model can be estimated by the "`mirt()`" function or the "`mixedmirt()`" function, both in the "mirt" packages. Both methods are provided. The use of the "`mirt()`" function to estimate the latent regression is slightly easier than the "`mixedmirt()`" in that the syntax with the "`mirt()`" function is shorter. For a multidimensional latent regression with only a few dimensions (e.g., 2-dimensional) or a unidimensional latent regression, the use of the "`mirt()`" function may estimate the model faster than the "`mixedmirt()`" function. The former uses the MML EM estimation method while the latter uses the Metropolis-Hastings Robins-Monro (MHRM; Cai, 2010) estimation method, which becomes more efficient as the model size increases (e.g., more than 3-dimensional 2PLM latent regression with multiple observed covariates). Both the MML EM and MHRM are full information maximum likelihood estimation approaches. (The utility of MHRM in the estimation of higher-dimensional IRT modeling is described in detail in section 5.7.)

The current "`mirt()`" function does not support the estimation of the information matrix for the latent regression modeling. The "`mixedmirt()`" is more specialized for the extended mixed-effects or explanatory IRT modeling, where a high-dimensional model is allowed to have fixed and random effects for the person and item parameters. Mean-centering or standardization of the covariates

(so that the rescaled covariates range from −10 to 10) are strongly recommended when using the "`mixedmirt()`" function for the extended mixed-effects IRT modeling (including the latent regression) to avoid floating point underflow in the process of the MHRM estimation (Chalmers, 2015).

We start the illustration assuming the "`mirt`" package is loaded in R and the working directory where data files are stored has been set (#4.1_2 and #4.1_3).

```
ddd<-read.table("c4_latent_reg.dat", header=T) #4.4_1
head(ddd) #4.4_2
table(ddd$x1) #4.4_3
opred<-ddd[,21:22] #4.4_4
ddd<-ddd[,1:20] # 4.4_5
```

#4.4_1. A dichotomous response data file is read into R by the "`read.table()`" command. The data file has a header (the first row) which shows the variable names for the 20 items ("It1," "It2," . . . , "It20") and two covariates ("x1," and "x2"). The names of the variables and the responses are read into R. The data file is stored as a data frame labeled as "`ddd`."

#4.4_2. This command shows the first six test-takers' item responses and the values of the two person-level observed covariates. "x1" is a dummy variable (0 or 1) and "x2" is a continuous variable (e.g., standardized test score).

#4.4_3. The "`table()`" command for "x1" shows the frequencies of test-takers within the two groups. The group coded as "`0`" has 730 test-takers, and the other group coded as "`1`" has 770 test-takers.

#4.4_4. The two person-level observed covariates are saved separately as a data frame object with the name "`opred`."

#4.4_5. The item response data are separately extracted and stored in the same R object name "`ddd`," excluding the covariate columns. "`ddd`" is now a data frame thstthat contains only item responses.

Using the "`mirt()`" function directly in the "mirt" package

```
mod.lr<-mirt(ddd, model=1, itemtype= "2PL",
covdata=opred, formula = ~x1+x2) #4.4_6
mod.lr #4.4_7
coef(mod.lr, simplify=T) #4.4_8
```

#4.4_6. The 2PLM latent regression IRT is estimated using "`mirt()`." "`model=1`" indicates a unidimensional model, and "`itemtype= "2PL"`" specifies the 2PLM. Two distinct, additional arguments in the command are "`covdata`" and "`formula`." The "`covdata`" option is to specify the data frame object which contains the observed covariates, here "`opred`." The "`formula`" argument provides the linear model regression formula. The formatting follows the regular R regression from the "`lm()`" function (or

linear model) model specification form (i.e., dependent variable ~ independent variable(s)). After the equal sign, "`~x1+x2`" is specified to allow the latent regression modeling of the "`x1`" and "`x2`" variables defined in the covariate data.

#4.4_7. The model estimation check is shown. The estimation finishes pretty quickly because no information matrix is estimated.

#4.4_8. The point estimates of the model are printed. Because "`IRTpars=T`" is not included in the "`coef()`" function, as it sometimes is, the slope ("`a1`") and intercept ("`d`") are printed. If "`IRTpars=T`" is included, then the discrimination ("`a`") and difficulty ("`b`") are printed.

```
> coef(mod.lr, simplify=T)
$'items'
        a1      d  g  u
It1  1.047  1.120  0  1
It2  1.176 -1.551  0  1
...
It19 0.803  0.454  0  1
It20 1.091 -1.377  0  1

$means
F1
 0

$cov
   F1
F1 1

$lr.betas
                F1
(Intercept) 0.000
x1  0.219
x2  0.737
```

In addition to the item parameter estimates, the regression coefficients estimated for "`x1`" and "`x2`" are shown under "`$lr.betas`." (These are printed regardless of the item parameterizations requested.) Again, the intercept of the regression model and the conditional variance of the ability distribution on "`x1`" and "`x2`" are fixed as 0 and 1, respectively, for the model identification purposes. "`x1`" is a dummy variable for the two groups. Controlling the effect of "`x2`," the group difference between the group coded as 1 and the group coded as 0 is estimated to be 0.219.

The following box shows the same model estimation with a different model identification constraint. The first item slope parameter fixed as 1, which replaces the constraint of the population variance equal to 1. For this purpose a syntax to impose this new constraint should be imported into the "`mirt()`" model estimation option.

```
spec<-'F = 1-20
   START = (1, a1, 1)
   FIXED = (1, a1)
    FREE = (GROUP, COV_11)' #4.4_9
mod.lr<-mirt(ddd, model=spec, itemtype= "2PL", covdata=opred,
formula = ~x1+x2) #4.4_10
mod.lr #4.4_11
coef(mod.lr, simplify=T) #4.4_12
```

#4.4_9. The uni-factor structure is first defined, and the first item slope parameter is first set to 1 (using "`START`") and then fixed by the first three lines. The last "`FREE`" argument frees the population θ variance as a model parameter.

#4.4_10. The 2PLM latent regression with the new constraint shown earliere-arlier is estimated with the "`model=spec`."

#4.4_11. The model run convergence check is shown. No second-order test check is printed due to the lack of the information matrix estimation.

#4.4_12. The point estimates of the model are shown.

```
> coef(mod.lr, simplify=T)
$'items'
        a1      d  g  u
It1  1.000  1.120  0  1
It2  1.121 -1.551  0  1
...
It19 0.766  0.454  0  1
It20 1.041 -1.378  0  1

$means
F
0

$cov
    F
F 1.1

$lr.betas
                F
(Intercept) 0.000
x1  0.229
x2  0.773
```

The estimated population variance is 1.1. The estimated coefficients for "`x1`" and "`x2`" are also different from the previous estimation with the default identification constraints. The coefficients (0.229 and 0.773) in the current model are higher than those in the previous estimation (0.219 and 0.737, see #4.4_8).

The hypothesis where the population coefficients of "`x1`" and "`x2`" in the latent regression are 0 can be tested using the likelihood test comparing the regular 2PLM

without including the latent regression component and the 2PLM latent regression model as follows.

```
mod.2pl<-mirt(ddd, model=spec, itemtype= "2PL") #4.4_13
anova(mod.2pl, mod.lr) #4.4_14
```

#4.4_13. The regular 2PLM is estimated using the same "`spec`" constraints used for the prior 2PLM latent regression and excluding "`covdata=opred, formula = ~x1+x2`."

#4.4_14. Information criteria and the likelihood ratio test results are shown.

```
> anova(mod.2pl, mod.lr)
Model 1: mirt(data = ddd, model = spec, itemtype = "2PL")
Model 2: mirt(data = ddd, model = spec, itemtype = "2PL", covdata = opred,
 formula = ~x1 + x2)
      AIC     AICc    SABIC       HQ      BIC    logLik      X2  df   p
1 29709.11 29711.36 29794.57 29788.29 29921.64 -14814.56     NaN NaN NaN
2 29151.36 29153.84 29241.09 29234.49 29374.52 -14533.68 561.753   2   0
```

AIC, AICc, SABIC, and BIC prefer the latent regression model as a better model-data fitting case. The likelihood ratio test *p*-value is virtually 0 (shown under the column "p"), indicating that at least one coefficient is statistically significantly different from 0 in the population.

When individual person latent trait or ability score estimation is desired, "`fscores()`" is used.

```
fscores(mod.lr)
```

The Rasch model latent regression can be estimated by replacing the "`2PL`" in the "`itemtype`" option with "`Rasch`."

```
mod.lr.rasch<-mirt(ddd, 1, itemtype='Rasch',
covdata=opred, formula = ~x1+x2)
```

In addition to item difficulty or intercept parameters with the slope (or discrimination) fixed to 1, latent regression coefficients are also estimated, as in the 2PLM latent regression case.

Using the "`mixedmirt()`" function in the "mirt" package

The syntax for the 2PLM latent regression with the "`mixedmirt()`" has an additional argument, the "`fixed`" option, which is for specifying the fixed and/or random effects for items. The output from the "`mixedmirt()`" takes a similar format to that of "`mirt()`"; thus, the estimation results here are presented only

when necessary. The output from the "`mixedmirt()`" shows not only the point estimates but also their standard errors. The "`simplify=T`" option when requesting the coefficients, which shows only the point estimates, does not work in the "`mixedmirt()`" case. Here is the syntax for executing the model.

```
mod.lr.2pl<-mixedmirt(ddd,model=1, itemtype = '2PL',
covdata=opred,, fixed = ~0 + items,
lr.fixed = ~x1+x2) #4.4_15
mod.lr.2pl #4.4_16
coef(mod.lr.2pl, printSE=T) #4.4_17
```

#4.4_15. The "`mixedmirt()`" fits the 2PLM latent regression. The "fixed" option is used to specify the fixed effects modeling for items, i.e., item parameters specified as fixed effects by typing "`fixed = ~0 + items`." Then "`lr.fixed`" specifies that the random effect for the person (this was "`formula`" in "`mirt()`"), i.e., person parameter θ is regressed upon "x1" and "x2."

#4.4_16. The model convergence check is shown. No particular issue found in this application. Note that MHRM is used for the model estimation. Also, the information matrix is estimated in using "`mixedmirt()`" function. The model estimation takes a longer time than when "`mirt()`" is used for this latent regression modeling. However, MHRM is a more efficient choice for a high-dimensional latent regression modeling or more complex mixed-effects IRT modeling.

#4.4_17. The point estimates of the model parameters and their standard errors are shown. The estimated coefficients (and standard errors) for "`x1`" and "`x2`" from this "`mixedmirt()`" are 0.230 (0.055) and 0.735 (0.031), respectively. Again, the default model identification constraints were used. The coefficients for "x1" and "x2" in the previous latent regression with the default model identification were 0.219 and 0.737.

The same latent regression with the alternative model identification constraints (where the first item slope is fixed as 1 instead of using the population variance fixed as 1) is estimated with the previously created model specification object "`spec`" (#4.4_9) and the "`mixedmirt()`" function as follows.

```
mod.lr.2pl<-mixedmirt(ddd,model=spec, itemtype = "2PL",
covdata=opred,, fixed = ~0 + items,
lr.fixed = ~x1+x2) #4.4_18
mod.lr.2pl #4.4_19
coef(mod.lr.2pl, printSE=T) #4.4_20
```

#4.4_18. The constraint of the fixed slope in the first item is implemented using the previously created object "`spec`" (#4.4_9). The rest of the options are the same as the latent regression with the default model identification constraints (#4.4_15).

#4.4_19. Convergence check is shown. No particular issue is found.

#4.4_20. The point estimates and their standard errors are displayed. The slope coefficients (and standard errors) for "x1" and "x2" are 0.222 (0.163) and 0.764 (0.223). The population variance was estimated to be 1.076 (0.557). A noticeable difference from the default constraint approach is the large increase in the standard errors. The computed log-likelihood values were very close: −14533.12 for the previous default constraint approach and −14533.94 for the current first item slope constraint approach. However, the standard errors for the item parameters, population, and regression parameters were noticeably different. The fixed item slope approach estimation method resulted in a large increase of the standard errors in general for all model parameters.

Given that the model parameterization could affect the model estimation, the information matrix computed in the fixed item slope approach appears to be overestimating the uncertainty related to the parameters (e.g., see the aforementioned population variance estimate of 1.076, with its standard error equal to 0.557). To improve the estimation of the information matrix in the fixed item slope approach, the use of a different type of standard error estimation procedure where an increased number of draws in the stochastic process of computing the information matrix in MHRM with the specification of the use of the fixed number of draws may be tried, as shown here. (See section 5.7 for more details of MHRM and the use of "FMHRM.")

```
mod.lr.v2 <-mixedmirt(ddd,model=spec, itemtype = "2PL",
covdata=opred,
fixed = ~0 + items, lr.fixed = ~x1+x2, SE.type = 'FMHRM',
technical=list(MHRM_SE_draws=10000))

coef(mod.lr.v2, printSE=T)
```

The model estimation takes a longer time for this approach due to the use of "SE.type = FMHRM" and increasing number of fixed draws to 10,000 using the "technical" argument from the default of 2000. The author's computer took about five minutes to run this model. The output of this implementation is shown here. Users can ignore "-999," "999," and "NA" for "g" and "u" in the output because the 2PLM is used.

```
> coef(mod.lr.2pl.a1f.v2, printSE=T)
$'It1'
     a1     d    g   u
par   1 1.122 -999 999
SE   NA 0.083   NA  NA
```

```
$It2
       a1      d    g   u
par 1.134 -1.544 -999 999
SE  0.147  0.096   NA  NA
...
$It20
       a1      d    g   u
par 1.057 -1.373 -999 999
SE  0.135  0.090   NA  NA

$GroupPars
    MEAN_1 COV_11
par      0  1.076
SE      NA  0.214

$lr.betas
 F_(Intercept)   F_x1  F_x2
par          0  0.222 0.764
SE          NA  0.061 0.081
```

The point estimates are the same as before. However, their standard errors are smaller for all model parameter estimates when "`FMHRM`" is used for estimating the standard error, and the number of draws was increased. The standard error of the ability population distribution variance conditional on "`x1`" and "`x2`" is estimated to be 0.214, which is deemed to be more reasonable than the previous estimate (0.557) from the result from the default setting run. The estimated slope coefficients' standard errors for "`x1`" and "`x2`" are 0.061 and 0.081, which are also smaller than the previous standard error estimates 0.163 and 0.223 from the default setting run.

The Rasch latent regression with the MHRM estimation method can also be implemented if the "`itemtype`" argument is replaced with "`Rasch`."

```
mod.lr.rasch<-mixedmirt(ddd,model=1, itemtype = 'Rasch',
covdata=opred,, fixed = ~0 + items,lr.fixed = ~x1+x2)
```

References

Bock, R. D., & Zimowski, M. F. (1997). Multiple group IRT. In W. J. van der Linden & R. K. Hambleton (Eds.), *Handbook of modern item response theory* (pp. 433–448). New York, NY: Springer-Verlag.

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

Chalmers, R. P. (2015). Extended mixed-effects item response models with the MH-RM algorithm. *Journal of Educational Measurement*, *52*, 200–222.

De Boeck, P., & Wilson, M. (Eds.) (2004). *Explanatory item response models: A generalized linear and nonlinear approach*. New York, NY: Springer.

Holland, P. W., & Thayer, D. T. (1988). Differential item performance and the Mantel Haenszel procedure. In H. Wainer & H. I. Braun (Eds.), *Test validity* (pp. 129–145). Hillsdale, NJ: Lawrence Erlbaum Associates.

Holland, P. W., & Wainer, H. (Eds.) (1993). *Differential item functioning*. Hillsdale, NJ: Lawrence Erbaum Associate.

Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd edition). New York, NY: Springer.

Paek, I. (2018). Understanding differential item functioning and item bias in psychological instruments. *Psychology and Psychotherapy*: *Research Study*, *1,* 1–2. doi:10.31031/PPRS.2018.01.000514

5

MULTIDIMENSIONAL IRT FOR SIMPLE STRUCTURE

Multidimensional IRT modeling assumes that more than one single latent trait underlies the total collection of item responses. These models are classified into two structures: between-item and within-item multidimensional modeling (Adams, Wilson, & Wang, 1997). The between-item multidimensionality refers to the case where each item in a test is an indicator for one of several traits, or dimensions, being measured by the test, and one of several ability constructs is required by the test-taker to respond to that item. In factor analysis, the between-item structure is known as simple structure. We use simple structure, or between-item multidimensionality, interchangeably. Within-item multidimensionality occurs when there exists at least one item within the set of test items that is an indicator for more than one trait, or dimension.

If a test has between-item multidimensionality, items can be classified as clusters, each of which measures a single dimension. For example, in a test having a 2-dimesional between-item structure, a cluster of items measures only dimension A, and another cluster of items measures only dimension B; in between-item structure, the dimensions are assumed to be correlated. (Note, a cluster must contain at least two items.)

When some or all items are measuring more than one dimension within a single item, the item response data has the within-item multidimensionality structure. Within-item multidimensionality is a cross-loading item structure, in general. In this chapter, multidimensional model applications for the simple structure are illustrated for dichotomous and polytomous item response data having two or three dimensions. The Rasch, 2PLM, and 3PLM are used to model dichotomous data, and the PCM, GPCM, and GRM are used to model polytomous data. Section 5.7 presents a high-dimensional IRT application to data having eight dimensions. The last section, section 5.8, demonstrates within-item multidimensional IRT modeling.

The "mirt" package is used for the demonstration of multidimensional IRT modeling in this chapter. All applications in this chapter start with reading a data file, assuming that the "mirt" package has been installed and loaded onto the current R session, and that the working directory has been set using the following commands. Readers may refer to #2.1_38, #2.1_39, and #2.1_3 for details on these.

```
install.packages("mirt")
library(mirt)
setwd("c:/mystudy/RIRT/")
```

Readers are referred to Reckase (2009) for more details of multidimensional IRT modeling in general. Note also that, in multidimensional modeling, the slope and intercept parameterization (i.e., the form of $a\theta + d$, where the intercept is a parameter related to item easiness) is widely used because only a single intercept is possible to estimate. Also, the applications of simple structure IRT modeling focus on fitting the intended multidimensional structure, estimating the item parameters and population ability distribution parameters, estimating the individual person ability parameters, and evaluating the overall model fit.

5.1 Multidimensional Rasch model application

The IRF of the Rasch model for simple structure with Q number of dimensions is

$$P\left(X_{ij}=1|\,\theta_{jq}\right)=\frac{\exp\left[\theta_{jq}-b_i\right]}{1+\exp\left[\theta_{jq}-b_i\right]}\,, \tag{5.1_1}$$

where X_{ij} is the jth person's response to the ith item (0 = incorrect/no and 1 = correct/yes), θ_{jq} is the jth person's latent trait or ability for dimension q ($q = 1, 2, \ldots, Q$), and b_i is the ith item difficulty. In the population modeling, the Q-variate normal distribution is assumed as the default in "mirt," where the means of the ability on each dimension are fixed as 0, and the variances and the covariances among the dimensions are freely estimated. Each test-taker has Q number of latent trait or ability estimates, one on each dimension (i.e., $\theta_{j(q=1)}, \theta_{j(q=2)}, \ldots, \theta_{j(q=Q)}$). "mirt" uses the $\theta_{jq} + d_i$ parameterization instead of $\theta_{jq} - b_i$ for the Rasch model. Thus, the item parameter estimates in "mirt" indicate item easiness (d_i), rather than item difficulty (b_i).

```
ddd<-read.table("c5_MIRT_Rasch.dat") #5.1_1
dim(ddd) #5.1_2
```

#5.1_1. A data file "c5_MIRT_Rasch.dat" has a space-delimited format and is read in by the "read.table()" command. The file holds dichotomous responses to 18 items. Item responses are assumed to be from an intentionally 3-dimensional test where items 1 through 6 measure one dimension, items 7

through 12 measure a second dimension, and items 13 through 18 measure a third dimension. The three constructs are assumed to be positively correlated. The default variable or item names are "V1," "V2," through "V18."

#5.1_2. The "`dim()`" command shows the size of the data set. The number of rows, or test-takers, is 610, and the number of columns, or items, is 18. Note, the "`dim()`" command refers only to the number of rows and columns of the data set. It does not provide any information as to the theoretical number of dimensional constructs being measured.

```
spec<-' F1=1-6
        F2=7-12
        F3=13-18
        COV=F1*F2*F3 ' #5.1_3
mod.md<-mirt(ddd, model=spec, itemtype="Rasch", SE=T) #5.1_4
mod.md #5.1_5
extract.mirt(mod.md, what="time") #5.1_6
coef(mod.md, simplify=T) #5.1_7
coef(mod.md, printSE=T) #5.1_8
```

#5.1_3. The 3-dimensional structure is specified using four lines of syntax. Dimension 1 ("`F1`") is identified with items 1 through 6; dimension 2 ("`F2`") is identified with items 7 through 12; and dimension 3 ("`F3`") is identified with items 13 through 18. Because simple structure is defined, no item corresponds to more than one dimension. The covariances among the three dimensions are freely estimated by specifying "`COV=F1*F2*F3`."

#5.1_4. The 3-dimensional Rasch model is estimated using the "spec" model object created in the previous step and the "`itemtype=Rasch`."

#5.1_5. Convergence check results are shown. No particular issues were detected in this application.

#5.1_6. Readers may have noticed that fitting the 3-dimensional model (#5.1_4) takes a relatively longer time to run than a unidimensional model. This command shows the total time to estimate the model. The author's computer took about 35 seconds to finish.

#5.1_7. The model parameter estimates are shown. The output contains the estimated slope for each item on each dimension ("`a1`," "`a2`," "`a3`"), the intercept ("`d`"), lower asymptote ("`g`"), and upper asymptote ("`u`") parameters.

For the Rasch IRF, all slopes are fixed as 1s on the associated dimension defined in "`spec`" and 0s on the other dimensions; the lower asymptote is fixed to 0, and the upper asymptote is fixed to 1. For the population ability distribution model parameters, three variances (one for each dimension) and three covariances (for all pairs of dimensions) are freely estimated in this 3-dimensional Rasch model, while

the means of the ability distribution on each dimension are fixed at 0 for model identification purposes.

```
> coef(mod.md, simplify=T)
$'items'
   a1 a2 a3      d g u
V1  1  0  0 -0.031 0 1
V2  1  0  0 -0.610 0 1
...
V6  1  0  0 -0.047 0 1
V7  0  1  0  0.943 0 1
V8  0  1  0 -1.128 0 1
...
V12 0  1  0 -0.071 0 1
V13 0  0  1 -0.353 0 1
V14 0  0  1  1.590 0 1
...
V18 0  0  1  0.462 0 1

$means
F1 F2 F3
 0  0  0

$cov
      F1    F2    F3
F1 1.157    NA    NA
F2 0.267 1.102    NA
F3 0.744 0.545 1.204
```

The values under "d" (d_i) are the "mirt" intercept, or easiness, estimates, which are equal to the negative values of the difficulty parameters ($d_i = -b_i$ in the Rasch model). The estimated variances of the ability distributions for the three dimensions are 1.157, 1.102, and 1.204, respectively. These are printed along the diagonal of the "$cov" matrix. The covariance between dimensions is printed on the lower off-diagonal. The correlation between two dimensions, q and q', can be calculated using the covariance and variances: $\text{cor}(q,q') = \text{cov}(q,q') / \sqrt{\text{var}(q) \times \text{var}(q')}$. For example, the correlation between dimensions 1 and 2 is $0.267 / \sqrt{1.157 \times 1.102} = 0.236$.

#5.1_8. The point estimates of the model parameters and their standard errors are shown with "printSE=T." Because the intercept parameter ("d") is simply the negative of the item difficulty, the standard errors for the intercept ("d") and item difficulty $\left(\hat{b}_i\right)$ are equivalent. In the Rasch model, the item slope, pseudo-guessing, and upper asymptote are fixed, so the standard errors are "NA."

```
> coef(mod.md, printSE=T)
$'V1'
     a1 a2 a3      d  logit(g) logit(u)
par   1  0  0 -0.031      -999      999
SE   NA NA NA  0.100        NA       NA

$V2
     a1 a2 a3      d logit(g) logit(u)
par   1  0  0 -0.610     -999      999
SE   NA NA NA  0.103       NA       NA
...

$V18
     a1 a2 a3      d logit(g) logit(u)
par   0  0  1  0.462     -999      999
SE   NA NA NA  0.102       NA       NA

$GroupPars
  MEAN_1 MEAN_2 MEAN_3 COV_11 COV_21 COV_31 COV_22 COV_32 COV_33
par    0      0      0  1.157  0.267  0.744  1.102  0.545  1.204
SE    NA     NA     NA  0.147  0.085  0.096  0.148  0.091  0.154
```

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.
  SE = T) #5.1_9
```

#5.1_9. With the "`fscores()`" command, EAP estimators on each dimension are requested using "`method="EAP"`." Scores are printed for each person when "`full.scores=T`," along with the estimate's standard error ("`full.scores.SE = T`").

```
> fscores(mod.md, method="EAP", full.scores=T, full.scores.SE = T)
              F1           F2           F3     SE_F1     SE_F2     SE_F3
 [1,]  0.692651965  0.374358826 -0.019110250 0.6661797 0.6888058 0.6479194
 [2,]  0.104514834  0.080635709  0.421151404 0.6516691 0.6843960 0.6491556
 [3,]  0.234393354 -0.280686695  0.740704546 0.6544414 0.6839008 0.6563728
 [4,] -0.006191207  0.932886395  0.216023308 0.6506241 0.7049385 0.6474958
 [5,] -1.473032554 -1.376440905 -1.932119964 0.7123553 0.7188878 0.7353033
...
[610,] 0.622479242 -1.544563068 -0.451018141 0.6645307 0.7223765 0.6564959
```

EAP estimates for the three dimensions are shown under "`F1`," "`F2,`" and "`F3`." Their standard errors are under "`SE_F1`," "`SE_F2`," and "`SE_F3`."

```
M2(mod.md) #5.1_10
mod.ud<-mirt(ddd, model=1, itemtype="Rasch", SE=T) #5.1_11
mod.ud #5.1_12
anova(mod.ud, mod.md) #5.1_13
```

#5.1_10. A global model-data fit test, M2, is conducted. See #2.1_53 and #3.1_52 for more descriptions of the M2 test statistic. The *p*-value is 0.95, and we fail to reject the fitted model.

```
> M2(mod.md)
            M2  df         p RMSEA RMSEA_5 RMSEA_95      SRMSR      TLI CFI
stats 119.8336 147 0.9509914     0       0        0 0.03464174 1.023973   1
```

#5.1_11. In order to check the 3-dimensional structure, i.e., a 3-dimensional vs. a unidimensional model, the data are also fit to the unidimensional Rasch model. Since the unidimensional model is nested within the multidimensional model (when the same IRF is used in both), a log-likelihood ratio test can be utilized to test the appropriateness of the multidimensional model over the unidimensional one.

#5.1_12. The unidimensional model estimation convergence is checked. No particular issue is detected in this unidimensional model fit.

#5.1_13. A relative model-data fit between unidimensional (#5.1_11) and 3-dimensional simple structure Rasch models (#5.1_4) is compared by the log-likelihood ratio test and information criteria. The *p*-value of the log-likelihood ratio test is virtually 0, rejecting the reduced, unidimensional model. The values of the descriptive model fit indices, AIC, AICc, BIC, and SABIC, in the multidimensional model ("`mod.md`"), are smaller than those from the unidimensional model ("`mod.ud`"). In conclusion, the 3-dimensional simple structure model is preferred to the unidimensional one.

```
> anova(mod.ud, mod.md)
Model 1: mirt(data = ddd, model = 1, itemtype = "Rasch", SE = T)
Model 2: mirt(data = ddd, model = spec, itemtype = "Rasch", SE = T)
       AIC      AICc    SABIC       HQ      BIC    logLik  X2  df   p
1 13287.56 13288.85 13311.1 13320.18 13371.42 -6624.781 NaN NaN NaN
2 13112.27 13114.32 13142.0 13153.47 13218.19 -6532.135 185.292 5 0
```

5.2 Multidimensional 2PLM application

The IRF of the 2PLM used for simple structure with dichotomous responses, where the number of dimensions is Q is

$$P\left(X_{ij}=1|\,\boldsymbol{\theta}_{jq}\right)=\frac{\exp\left[a_{iq}\boldsymbol{\theta}_{jq}+d_i\right]}{1+\exp\left[a_{iq}\boldsymbol{\theta}_{j}+d_i\right]}, \tag{5.1_1}$$

where X_{ij} is the jth person's response to the ith item (0 = incorrect/no; and 1 = correct/yes), θ_{jq} is the jth person's latent trait or ability for the qth dimension (q = 1, 2, . . . , Q), d_i is the ith item intercept related to item easiness, and a_{iq} is the ith item slope parameter for the qth dimension. Each test-taker has Q number of latent trait

scores ($\theta_{j(q=1)}, \theta_{j(q=2)}, \ldots, \theta_{j(q=Q)}$) estimated with the accommodation of the population dimension correlations (i.e., $\rho_{qq'}$ where $q = 1, 2, \ldots, Q$ and $q' = 1, 2, \ldots, Q$). Because each item is considered to be an indicator for a single dimension (under simple structure), the item slopes for all other dimension except for the qth dimension are 0.

```
ddd<-read.table("c5_MIRT_2PL.dat") #5.2_1
dim(ddd) #5.2_2
```

#5.2_1. "c5_MRT_2PL.dat" is a space-delimited format file that is read into R using "`read.table()`." The file contains dichotomous item responses to 11 items that are assumed to be from an intentionally 2-dimensional test. Items 1 through 6 measure one construct and items 7 through 11 measure the other. The two constructs are assumed to be positively correlated. The default variable (or item) names given to the saved data object "`ddd`" in R are "V1," "V2," . . . , and "V11."

#5.2_2. The size of the data set, i.e., number of rows and columns of the data, not necessarily the number of underlying latent trait constructs being modeled, of "`ddd`" are 1100 rows, or test-takers, and 11 columns, or items. See #5.1_2.

#5.2_3. A three-line syntax for the model specification is written and stored under "`spec`." The two lines regarding "`F1`" and "`F2`" define the two dimensions, where items 1 through items 6 load onto the first dimension, and items 7 through 11 load onto the second dimension. The last line states that the covariance of the two dimensions is a freely estimated parameter. For model identification, the mean and the variance of the ability distribution of each dimension are set to 0 and 1, respectively. Thus, the free covariance here is equal to the correlation between the two dimensions.

```
spec<-'F1=1-6
       F2=7-11
       COV=F1*F2' #5.2_3
mod.md<-mirt(ddd, model=spec, itemtype="2PL", SE=T)
  #5.2_4
mod.md #5.2_5
coef(mod.md, simplify=T) #5.2_6
coef(mod.md, printSE=T) #5.2_7
```

#5.2_4. The 2-dimensional, simple structure model is estimated using the "`spec`" model definition created in the previous step (#5.2_3) according to the 2PLM (using "`itemtype="2PL"`"). The results are saved as the object "`mod.md`."

#5.2_5. Convergence results are shown. With the current data, no problem was flagged.

#5.2_6. The estimated item parameters and the population ability distribution covariance (equal to the correlation) estimate are printed. The slope estimates for items loading on the first dimension (i.e., items 1 through 6) are shown under the column "a1," and those for items loading on the second dimension (items 7 through 11) are shown under "a2." The slope estimates for the dimension not being measured by each item is equal to 0.000. The intercept parameter estimates are shown under the column "d."

```
> coef(mod.md, simplify=T)
$'items'
       a1    a2      d g u
V1  1.578 0.000  1.483 0 1
V2  0.926 0.000  1.015 0 1
...
V6  1.077 0.000 -1.525 0 1
V7  0.000 1.978  1.744 0 1
V8  0.000 1.424  0.843 0 1
...
V11 0.000 1.976 -1.345 0 1

$means
F1 F2
 0  0

$cov
      F1 F2
F1 1.000 NA
F2 0.527  1
```

The estimated covariance between the two dimensions is 0.527, which is equivalent to the correlation estimate due to the unit variance for each dimension used for model identification.

The estimated slope values indicate the direction and the strength of association between the item and the dimension. The intercept estimate is related to item difficulty, but due to the use of a positive intercept ($+d_i$), a higher intercept value corresponds to an easier item. In the simple structure dichotomous models, item difficulty can be straightforwardly calculated by $b_{iq} = -d_i/a_{iq}$, which is the ith item difficulty on the qth dimension (e.g., for item 2 measuring the dimension 1, $-1.015/0.926 = -1.096$). In the 2PLM, the pseudo-guessing parameter is fixed to 0, and the upper asymptote is fixed to 1.

When the unidimensional 2PLM item parameters were printed, the "IRTpars=T" was used to print the IRT parameterization of the item parameters. If "IRTpars=T" is included in the "coef()" for a multidimensional model,

an error is printed indicating "traditional parameterization is only available for unidimensional models."

#5.2_7. Including "`printSE=T`" in "`coef()`" prints both the model parameter point estimates and their standard errors.

```
> coef(mod.md, printSE=T)
$'V1'
       a1  a2      d  logit(g)  logit(u)
par 1.578   0  1.483     -999        99
SE  0.168  NA  0.120       NA        NA
...
$V11
     a1     a2      d  logit(g)  logit(u)
par   0  1.976 -1.345     -999       999
SE   NA  0.238  0.140       NA        NA

$GroupPars
 MEAN_1 MEAN_2 COV_11 COV_21 COV_22
par   0      0      1  0.527      1
SE   NA     NA     NA  0.040     NA
```

#5.2_8. The person latent trait or ability estimates are obtained using the same "`fscores()`" command that was used in the unidimensional model applications with the "mirt" package.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.
  SE = T) #5.2_8
```

The EAP estimator was requested (using "`method="EAP"`") for each test-taker (with "`method="EAP"`"). Using "`full.scores=F`" produces a different format of the results where unique item response patterns and their associated EAP estimates and standard errors are printed.

```
> fscores(mod.md, method="EAP", full.scores=T, full.
  scores.SE = T)
                    F1            F2      SE_F1      SE_F2
   [1,]  4.354824e-01  0.139184725 0.5338155 0.5517143
   [2,] -3.266189e-01  0.697851497 0.5333947 0.5772014
   [3,] -1.872017e-01  0.354403437 0.5271713 0.5580011
   [4,]  5.319757e-01  1.254826148 0.5433742 0.6309118
...
[1098,] -7.533948e-03 -0.467778473 0.5235651 0.5518395
[1099,] -6.184837e-01 -1.376740980 0.5538297 0.6176805
[1100,] -4.712196e-01  0.664484453 0.5412461 0.5752827
```

Because there are two dimensions, each person has two ability (EAP) estimates – one for the first dimension and one for the second dimension. Their associated standard errors are shown under "SE_F1" and "SE_F2."

```
M2(mod.md) #5.2_9
mod.ud<-mirt(ddd, model=1, itemtype="2PL", SE=T) #5.2_10
mod.ud #5.2_11
anova(mod.ud, mod.md) #5.2_12
```

#5.2_9. A global model-data fit test, M2, is conducted. See #2.1_53 and #3.1_52 for more descriptions of M2.

```
> M2(mod.md)
          M2 df         p     RMSEA RMSEA_5   RMSEA_95      SRMSR       TLI      CFI
stats 48.1068 43 0.2737334 0.0103954       0 0.02358614 0.02419574 0.9969235 0.9975948
```

The p-value of the test is 0.274, failing to reject the fitted model under $\alpha = 0.05$.

#5.2_10 and #5.2_11. To evaluate the use of the 2-dimensional model, a unidimensional 2PLM is fitted to data, and the convergence is checked.

#5.2_12. A relative model-data fit comparison is conducted between the unidimensional 2PLM (#5.2_9) and the 2-dimensional 2PLM (#5.2_4).

```
> anova(mod.ud, mod.md)
Model 1: mirt(data = ddd, model = 1, SE = T, itemtpe = "2PL")
Model 2: mirt(data = ddd, model = spec, itemtype = "2PL", SE = T)
       AIC     AICc    SABIC      HQ      BIC     logLik  X2  df   p
1 14252.27 14253.21 14292.46 14293.91 14362.34 -7104.136 NaN NaN NaN
2 14033.41 14034.43 14075.42 14076.94 14148.48 -6993.704 220.865 1 0
```

The values of the descriptive model fit indices, AIC, AICc, BIC, and SABIC, in the multidimensional model ("mod.md") are smaller than those from the unidimensional model ("mod.ud"). In conclusion, the 2-dimensional simple structure model is preferred to the unidimensional one. The log-likelihood ratio test p-value is virtually 0 (shown under the last column "p"), rejecting the unidimensional model and supporting the same conclusion.

5.3 Multidimensional 3PLM application

The 3PLM IRF for simple structure with Q number of dimensions is

$$P\left(X_{ij}=1|\,\theta_{jq}\right)=g_i+\left(1-g_i\right)\frac{\exp\left[a_{iq}\theta_{jq}+d_i\right]}{1+\exp\left[a_{iq}\theta_{jq}+d_i\right]}. \tag{5.3_1}$$

g_i is the ith item pseudo-guessing parameter. All the other notations are the same as those in the 2PLM for simple structure, see Equation 5.2_1. As in the 2PLM for simple structure, the slope and intercept form ($a\theta + d$) is used.

```
ddd<-read.table("c5_MIRT_3PL.dat") #5.3_1
dim(ddd) #5.3_2
```

#5.3_1. A dichotomous response data file "c5_MIRT_3PL.dat" is read and stored as "ddd." Twenty-one dichotomous item responses are assumed to be from an intentionally 3-dimensional test where items 1 through 7 measure the first dimension, items 8 through 14 measure the second, and items 15 through 21 measure the third. The three dimensions, or constructs, are assumed to be correlated. The default variable, or item, names are "V1" through "V21."

#5.3_2. The size of the data object "ddd" is shown. It has 2500 rows, or test-takers, and 21 columns, or items.

```
spec<- 'F1=1-7
        F2=8-14
        F3=15-21
        PRIOR = (1-21, g, norm, -1.7, 2)
        COV=F1*F2*F3 ' #5.3_3
mod.md<-mirt(ddd, model=spec, itemtype="3PL", SE=T)
  #5.3_4
mod.md #5.3_5
extract.mirt(mod.md, what="time") #5.3_6
coef(mod.md, simplify=T) #5.3_7
coef(mod.md, printSE=T) #5.3_8
```

#5.3_3. A 3-dimensional, simple structure with the free person ability population covariance parameters is specified. Items 1 through 7, items 8 through 14, and items 15 through 21 measure dimensions 1, 2, and 3, respectively. The normal distribution prior is used for the logit of the pseudo-guessing parameter (g_i), with its mean equal to −1.7 and variance equal to 2 (see #2.4_8). The use of the mean of −1.7 sets the guessing parameter values to be around 0.15. The covariances between pairs of ability distributions on each dimension are freely estimated.

#5.3_4. The 3-dimensional 3PLM for simple structure is estimated using the "model=spec" created in #5.3_3 and "itemtype="3PL"."

#5.3_5. The convergence check results are printed. No particular issues were detected for this application.

#5.3_6. The model estimation time is separately extracted by the command "extract.mirt()." In general, a multidimensional model estimation is likely to take a longer time to calibrate than a unidimensional model estimation (assuming the same data and IRF are used). The time for the model

estimation depends on the number of items, sample size, model complexity (e.g., the number of parameters for an IRF), and the number of dimensions. With the author's personal computer, fitting the 3-dimensional 3PLM for simple structure to data with a sample size of 2500 and 21 items took about 94 seconds. This is much longer compared to the time for the unidimensional 3PLM estimation fitted to the same data (3.1 seconds).

#5.3_7. The point estimates of the model parameters are printed.

```
> coef(mod.md, -simplify=T)
$'items'
        a1    a2    a3      d     g u
V1   1.644 0.000 0.000  1.470 0.084 1
V2   1.334 0.000 0.000  0.529 0.309 1
...
V7   1.751 0.000 0.000 -1.242 0.244 1
V8   0.000 1.467 0.000 -0.587 0.138 1
V9   0.000 1.440 0.000  1.539 0.072 1
...
V14  0.000 2.367 0.000  2.221 0.280 1
V15  0.000 0.000 0.781  0.364 0.083 1
V16  0.000 0.000 2.265  1.390 0.112 1
...
V21  0.000 0.000 1.228 -1.919 0.183 1

$means
F1 F2 F3
 0  0  0

$cov
      F1    F2 F3
F1 1.000    NA NA
F2 0.409 1.000 NA
F3 0.715 0.523  1
```

The item slope estimates are shown under "`a1`," "`a2`," and "`a3`." The intercept parameter and the pseudo-guessing parameter estimates are under "`d`" and "`g`," respectively. Again, the intercept parameter here takes the form of "$+d_i$," which is related to item easiness. For the item difficulty parameterization, $b_i = -d_i/a_i$ may be used.

The estimated ability distribution covariances between dimensions, which are correlations in this case due to the unit variance in each dimension, are 0.409 between dimensions 1 and 2, 0.715 between dimensions 1 and 3, and 0.523 between dimensions 2 and 3.

#5.3_8. Both point estimates and their standard errors are printed for the original parameterization used in "mirt," including the logit of g_i. This logit of g_i is estimated and converted into the pseudo guessing parameter, g_i, which was

provided in #5.3_7. No standard error request is currently available for the g_i parameterization in the multidimensional model estimation in "mirt."

```
> coef(mod.md, printSE=T)
$'V1'
        a1  a2  a3     d  logit(g) logit(u)
par  1.644   0   0  1.47    -2.388      999
SE   0.185  NA  NA  0.19     1.409       NA
$V2
        a1  a2  a3      d logit(g) logit(u)
par  1.334   0   0  0.529   -0.803      999
SE   0.320  NA  NA  0.436    0.699       NA
...
$V21
     a1  a2     a3      d logit(g) logit(u)
par   0   0  1.228 -1.919   -1.496      999
SE   NA  NA  0.324  0.496    0.265       NA

$GroupPars
 MEAN_1 MEAN_2 MEAN_3 COV_11 COV_21 COV_31 COV_22 COV_32 COV_33
par   0      0      0      1  0.409  0.715      1  0.523      1
SE   NA     NA     NA     NA  0.029  0.025     NA  0.029     NA
```

#5.3_9. The person latent trait or ability score estimates via the EAP estimator are requested. Each person has three EAP estimates, one for each of the three dimensions.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.
  SE = T) #5.3_9
```

The EAP estimates are shown under "F1," "F2," and "F3." Their standard errors are shown under "SE_F1," "SE_F2," and "SE_F3."

```
> fscores(mod.md, method="EAP", full.scores=T, full.scores.SE = T)
                F1           F2           F3     SE_F1     SE_F2     SE_F3
   [1,] -0.5318807894 -0.7116820169 -1.1793071250 0.5668761 0.6674024 0.5628653
   [2,] -0.1029933977 -1.4867364591 -0.1173057025 0.5105175 0.5661448 0.5637399
...
[2499,] -1.0748579414 -0.6188002350 -1.0636633095 0.5577289 0.5539886 0.5818998
[2500,] -0.9531282728 -0.4208805560 -0.2624443812 0.5464916 0.6390877 0.5663955
```

```
M2(mod.md) #5.3_10
spec<-'F1=1-21
PRIOR = (1-21, g, norm, -1.7, 2)' #5.3_11
mod.ud<-mirt(ddd, model=spec, itemtype="3PL", SE=T) #5.3_12
mod.ud #5.3_13
anova(mod.ud, mod.md) #5.3_14
```

#5.3_10. An absolute model-data fit test, M2, is conducted. See #2.1_53 and #3.1_52 for more details on M2. The *p*-value of the test M2 is 0.278. The fitted model is supported under $\alpha = 0.05$.

```
> M2(mod.md)
           M2  df         p       RMSEA RMSEA_5   RMSEA_95      SRMSR      TLI       CFI
stats 175.2448 165 0.2778583 0.004984564       0 0.01057991 0.01665368 0.998374 0.9987224
```

#5.3_11 and #5.3_12. For a relative model fit comparison of the dimensional structure, the unidimensional 3PL model is fitted to the same data with the same specification of the prior used in the multidimensional case.

#5.3_13. The unidimensional model estimation convergence check is shown. No issue was detected in the unidimensional model estimation.

#5.3_14. The unidimensional 3PLM (#5.3_12) and the 3-dimensional, simple structure 3PLM (#5.3_4) are compared using the AIC, AICc, SABIC, BIC, and DIC descriptive model indices. These indices indicate that the multidimensional model is preferred over the unidimensional model. The final value in the optimization process is shown under "`logPost`." The log-posterior, "`logPost`," is the summation of two sources: the contribution by the log-likelihood, "`logLik`," and the contribution by the prior, which is not shown in this output. As the sample size increases, the contribution by the "`logLik`" increases, dominating the optimization function value, "`logPost`." AIC, AICc, SABIC, and BIC are based on the likelihood value, "`logLik`" only, and penalize the model complexity.

DIC is a Bayesian information criterion analogous to AIC and is smaller for the multidimensional model, indicating the multidimensional model is preferred. Bayes factor is a model comparison method used in Bayesian analysis. It represents a ratio of the posterior odds for two comparing models (unidimensional model with multidimensional model here) to the prior odds for the two models being compared. A larger value of the Bayes factor indicates a stronger support for the reduced, unidimensional model. A small value (e.g., less than 1) supports the multidimensional model in our application. The value of the Bayes factor is virtually 0, supporting the use of the multidimensional model.

```
> anova(mod.ud, mod.md)
Model 1: mirt(data = ddd, model = spec, itemtype = "3PL", SE = T)
Model 2: mirt(data = ddd, model = spec, itemtype = "3PL", SE = T)
       AIC     AICc    SABIC       HQ      BIC      DIC    logLik   logPost Bayes_Factor
1 63542.03 63545.34 63708.78 63675.24 63908.94 63614.07 -31708.01 -31744.04           NA
2 62730.71 62734.35 62905.40 62870.26 63115.10 62800.59 -31299.36 -31334.29            0
```

5.4 Multidimensional partial credit model (PCM) application

The category response function (CRF) of the multidimensional PCM for the simple structure with Q dimension ($q = 1, 2, \ldots, Q$) has the following form:

$$P\left(X_{ikj} = k|\, \theta_{jq}\right) = \frac{\exp\sum_{h=0}^{k}\left(\theta_{jq} - b_{ih}\right)}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left(\theta_{jq} - b_{ih}\right)}, \qquad (5.4\_1)$$

where X_{ikj} is the jth person's response in category k to the ith item, θ_{jq} is the jth person's latent trait or ability for the qth dimension, and b_{ih} is the threshold parameter for the kth category of item i, when $h = k$. The parameterization in "mirt" is $\theta_{jq} + b_{ik}$, rather than $\theta_{jq} - b_{ik}$.

The multidimensional PCM is estimated by imposing constraints on the multidimensional GPCM. By constraining the slope of items on specific dimensions to 1 or 0, the GPCM is reduced to the PCM. In "mirt," the slope-intercept parameterization is used, i.e., $a_{iq}\theta_{jq} + d^{\star}_{ik}$. However, the $d^{\star}_{ik}$ parameter is not the usual intercept parameter, so $d_{ik} \neq -a_{iq}b_{ik}$ in the estimation of GPCM. Instead, the intercept parameter is defined as $d^{\star}_{ik} = \sum_{h=0}^{k} d_{ik}$.

The authors did not find an effective implementation of the multidimensional version of RSM, which is a further constraint of the PCM, in the current version of the "mirt." The "`rsm`" option in "mirt" is only valid for the unidimensional model case. (See section 3.2.) This chapter provides the application of the multidimensional PCM.

```
ddd<-read.table("c5_MIRT_PCM.dat") #5.4_1
dim(ddd) #5.4_2
apply(ddd,2,table) #5.4_3
```

#5.4_1. A response data file, which has space-delimited format, is read into R using the "`read.table()`" command. Polytomous item responses to nine items are assumed to be from an intentionally 3-dimensional test, where items 1 through 3 measure the first dimension, items 4 through 6 measure the second dimension, and items 7 through 9 measure the third. Responses were measured on a four-point scale, with the options 0, 1, 2, or 3. The three dimensions are assumed to be correlated. The default variable names of "V1," "V2," . . . , "V9" are given to the items.

#5.4_2. The size of the data set is printed. The number of rows, or test-takers, is 880, and the number of columns, or item variables is nine.

#5.4_3. Frequency analysis is conducted for each item. The "`apply()`" function shows the response categories (four categories here) for each item and the frequency responses of each. There were no null, or unused, categories in this data set. In addition, the "`summary()`" command may be used, by typing "`summary(ddd)`," which provides descriptive statistics (min., max., median, mean, etc.) for each items' responses.

```
spec <- '
       F1 = 1-3
       F2 = 4-6
       F3 = 7-9
       START = (1-3, a1, 1.0)
       START = (4-6, a2, 1.0)
       START = (7-9, a3, 1.0)
       FIXED = (1-3, a1)
       FIXED = (4-6, a2)
```

```
        FIXED = (7-9, a3)
        FREE = (GROUP, COV_11)
        FREE = (GROUP, COV_22)
        FREE = (GROUP, COV_33)
        COV = F1*F2*F3 ' #5.4_4
mod.md.pcm <- mirt(ddd, model=spec, itemtype='gpcm', SE=T)
#5.4_5
mod.md.pcm #5.4_6
extract.mirt(mod.md.pcm, what="time") #5.4_7
coef(mod.md.pcm, simplify=T) #5.4_8
coef(mod.md.pcm, printSE=T) #5.4_9
```

#5.4_4. A model specification, "`spec,`" is created to define the multidimensional item loading structure and the parameter constraints to specify the multidimensional PCM from the multidimensional GPCM (which is how the PCM is fit in "mirt"). The first three lines starting with "`F1`," "`F2`," and "`F3`" define the item loading structure. Items 1 through 3 load on the first dimension, "`F1`"; items 4 through 6 load on the second dimension, "`F2`"; and items 7 through 9 load on the third dimension, "`F3`." To fix the appropriate slope parameter equal to 1 as defined in the PCM, i.e., $a_{i1} = 1$ for items 1 through 3, $a_{i2} = 1$ for items 4 through 6, and $a_{i3} = 1$ for items 7 through 9, the "`START`" and "`FIXED`" commands are used. The "`START`" definition contains the items, the parameter, and the starting value; the "`FIXED`" definition contains the items and the parameter. For instance, items 4 through 6 are given 1.0 for their slopes for the second dimension ("`START=(4-6, a2, 1.0)`"), and they are fixed by the command "`FIXED (4-6, a2)`."

The "`FREE`" command relaxes the default constraint imposed on the population ability distribution variances, making the variances of the ability distribution of the three dimensions free parameters. The means of the three dimensions are constrained to be 0 by default. Finally, the covariances of the ability distribution between dimensions, whose default values are 0 (which implies uncorrelated dimensions), are made to be free parameters by the command "`COV.`"

#5.4_5. The multidimensional PCM is estimated with the "`mirt()`" function according to "`spec`" model definition from #5.4_4 and "`itemtype='gpcm'`."

#5.4_6. Convergence check results are shown. No issue was detected in this application.

#5.4_7. The run time for the model estimation is printed. The author's computer took about 42 seconds to finish the estimation.

#5.4_8. The point estimates of the multidimensional PCM parameters following the parameterization used in "`mirt`" are printed.

```
> coef(mod.md.pcm, simplify=T)
$'items'
   a1 a2 a3 ak0 ak1 ak2 ak3 d0     d1     d2     d3
V1  1  0  0   0   1   2   3  0  0.602 -0.147 -1.339
V2  1  0  0   0   1   2   3  0 -0.870 -2.383 -3.905
...
V8  0  0  1   0   1   2   3  0 -0.172 -0.967 -2.611
V9  0  0  1   0   1   2   3  0  1.387  1.870  1.956

$means
F1 F2 F3
 0  0  0

$cov
      F1    F2    F3
F1 1.095    NA    NA
F2 0.542 0.933    NA
F3 0.607 0.488 0.938
```

As mentioned, "`mirt()`" uses the $a_{iq}\theta_{jq} + d^{\star}_{ik}$ parameterization, where $d^{\star}_{ik} = \sum_{h=1}^{k} d_{ih}$, and d_{ik} is the regular item intercept parameter equal to $-a_{iq}b_{ik}$. Because $a_{iq} = 1$ for the qth dimension, where the ith item load in this PCM, $d^{\star}_{ik} = -\sum_{h=0}^{k} b_{ik}$ here. These intercept parameters correspond to the category response options for each item (and are not specific to a dimension). "`ak1`," "`ak2`," and "`ak3`" are the scoring function parameters, and they are fixed as 0, 1, 2, and 3 for the PCM. See the GPCM's alternative parameterization form shown in Equation 3.3_3, where "`k`" corresponds to the value given to the scoring function parameter for the GPCM. The item slope values shown in "`a1`," "`a2`," and "`a3`" are fixed at either 1 or 0 for this 3-dimensional PCM.

The estimated variances and covariances for three latent trait dimensions are shown under "`$cov`"; the variances are along the diagonal, and the covariances are the off-diagonal elements. With the estimated population covariances and variances, a correlation between two dimensions can be calculated by $\text{cor}(q,q') = \text{cov}(q,q') / \sqrt{\text{var}(q) \times \text{var}(q')}$. For example, the correlation between dimensions 1 and 2 is $0.542 / \sqrt{1.095 \times 0.933} = 0.536$.

#5.4_9. The point estimates and their standard errors both are shown. To convert the "mirt" PCM's $d^{\star}_{ik}$ into the regular intercept parameter $d_{ik} = -a_{iq}b_{ik}$, ($d_{ik} = -b_{ik}$ in this PCM application because of $a_{iq} = 1$), the following steps may be taken.

```
dd.mirt<-coef(mod.md.pcm, simplify=T)$'items'[,9:11]
  #5.4_10
d1<- dd.mirt[,1] #5.4_11
d2<- dd.mirt[,2]-(dd.mirt[,1]) #5.4_12
d3<- dd.mirt[,3]-(dd.mirt[,2]) #5.4_13
cbind(d1,d2,d3) #5.4_14
-cbind(d1,d2,d3) #5.4_15
```

#5.4_10. First, the intercept parameter $(d_{ik}^{\star})$ values are extracted from columns 9, 10, and 11 and saved as an object "`dd.mirt`."

#5.4_11, #5.4_12, and #5.4_13. The values of the regular intercept, d_{ik}, are calculated. The first intercept, "`d1`," is equal to $d_{i1}^{\star}$. The second intercept, "`d2`," is equal to $d_{i2}^{\star} - d_{i1}^{\star}$. The third intercept, "`d3`," is equal to $d_{i3}^{\star} - d_{i2}^{\star}$.

#5.4_13. The regular intercept values ("`d1`," "`d1`," and "`d3`") are combined and printed.

#5.4_15. The regular intercept is combined and converted into the value of b_{ik} in Equation 5.4_1, by taking the negative of each d_i.

```
> -cbind(d1,d2,d3)
            d1          d2           d3
V1 -0.6016920  0.7484118  1.19265197
V2  0.8701261  1.5131043  1.52158295
...
V9 -1.3869479 -0.4830686 -0.08583667
```

```
fscores(mod.md.pcm, method="EAP", full.scores=T, full.
  scores.SE = T) #5.4_16
```

#5.4_16. Person ability (or latent trait) scores are estimated using the EAP estimator. Each person has three estimates, one for each of the three dimensions, which are shown under "`F1`," "`F2`," and "`F3`." Their standard errors are shown under "`SE_F1`," "`SE_F2`," and "`SE_F3`."

```
> fscores(mod.md.pcm, method="EAP", full.scores=T, full.scores.SE = T)
                 F1            F2           F3     SE_F1     SE_F2     SE_F3
  [1,]  0.1771789886 -1.237924e-01 0.0301591282 0.5526513 0.4938639 0.5179421
  [2,] -0.0663952992 -1.292295e-01 0.2377226468 0.5615536 0.4948054 0.5297527
...
[880,]  0.0042891252 -8.588194e-02 0.5282281279 0.5592364 0.4926940 0.5465459
```

```
M2(mod.md.pcm) #5.4_17
```

#5.4_17. The overall model-data fit test M2 is conducted. The default test for polytomous response data is "M2⋆" in "mirt," which reduces to "M2" for dichotomous data. Users can get the same results by using "`M2(mod.md, type= "M2*")`." When the degrees of freedom cannot be calculated, then the "C2" statistic can be invoked. In this application, the p-value of the M2⋆ is 0.57, failing to reject the fitted model. See #3.1_52 for more descriptions of the overall model fit test M2⋆ for polytomous data.

```
> M2(mod.md.pcm)
           M2 df         p RMSEA RMSEA_5   RMSEA_95      SRMSR       TLI CFI
stats 10.47232 12 0.5745975     0       0 0.03069035 0.02416294 1.002873   1
```

5.5 Multidimensional generalized partial credit model (GPCM) application

In the previously discussed PCM, the item slope was fixed to 1 for the associated dimension. In the GPCM, the item slope is a free parameter. In the simple structure multidimensional GPCM, the slope of an item is a freely estimated, non-zero parameter on one dimension and 0 on all other dimensions. For the ith item in category $k \in \{0,1,2,\ldots,m_i\}$, the CRF of the GPCM for the Q-dimensional simple structure is

$$P\left(X_{ikj} = k|\,\theta_{jq}\right) = \frac{\exp\sum_{h=0}^{k}\left[a_{iq}\left(\theta_{jq} - b_{ih}\right)\right]}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left[a_{iq}\left(\theta_{jq} - b_{ih}\right)\right]}, \tag{5.5_1}$$

where X_{ikj} is the jth person's response in category k to the ith item, a_{iq} is the ith item slope for the qth dimension, θ_{jq} is the jth person's latent trait or ability score on dimension q, and b_{ih} is the threshold parameter for the ith item in category k, when $h = k$.

In the MML estimation of the GPCM in "mirt," in addition to fixing the means of the ability distributions for each dimension as 0, the variances of the ability distributions for each dimension are fixed as 1 for model identification, that is, $\theta \sim N(0,1)$ within each dimension. The popular slope and intercept form parameterization is

$$P\left(X_{ikj} = k|\,\theta_{jq}\right) = \frac{\exp\sum_{h=0}^{k}\left(a_{iq}\theta_{jq} + d_{ih}\right)}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left(a_{iq}\theta_{jq} + d_{ih}\right)}, \tag{5.5_2}$$

where $d_{ik} = -a_{iq}b_{ik}$. The actual GPCM model implemented in "mirt" is

$$P\left(X_{ikj} = k|\,\theta_{jq}\right) = \frac{\exp\sum_{h=0}^{k}\left(a_{iq}\theta_{jq} + d^{\star}_{ih}\right)}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left(a_{iq}\theta_{jq} + d^{\star}_{ih}\right)}, \tag{5.5_3}$$

where $d^{\star}_{ik} = \sum_{h=0}^{k} d_{ik}$ when $h = k$. Thus, usual intercept form can be calculated as $d_{ik} = d^{\star}_{ik} - d^{\star}_{i(k-1)}$ when $h = k$. This is the same parameterization form as described for the PCM in section 5.4.

```
ddd<-read.table("c5_MIRT_GPCM.dat") #5.5_1
dim(ddd) #5.5_2
apply(ddd,2,table) #5.5_3
```

#5.5_1. A space-delimited format item response data file is read into R and stored in the name of "`ddd`" by "`read.table()`." Polytomous item responses to 12 items are assumed to be from an intentionally 3-dimensional test where items 1 through 4 measure the first dimension, items 5 through 8 measure the second dimension, and items 9 through 12 measure the third. Responses were measured on a four-point scale, with the options 0, 1, 2, or 3. The three constructs are assumed to be correlated.

#5.5_2. The size of the R data object "ddd" is shown. This data set has 900 rows, or test-takers, and 12 columns, or items, and is stored as the data object "ddd."

#5.5_3. Frequency analysis of items is conducted. The result shows that each item has four categories scored 0, 1, 2, and 3, and that there were no null, or unused, categories.

```
spec <- '
F1 = 1-4
F2 = 5-8
F3 = 9-12
COV = F1*F2*F3' #5.5_3
mod.md <- mirt(ddd, model=spec, itemtype='gpcm', SE=T)
#5.5_4
mod.md #5.5_5
coef(mod.md, simplify=T) #5.5_6
coef(mod.md, printSE=T) #5.5_7
```

#5.5_3. The 3-dimensional, simple structure is specified. The first three lines starting with "F1," "F2," and "F3" define the item loading structure. Items 1 through 4 load on the first dimension, "F1"; items 5 through 8 load on the second dimension, "F2"; and items 9 through 12 load on the third dimension, "F3." "COV=F1*F2*F3" allows the covariances of three latent traits in the population to be free parameters. For model identification, the means and the variances of the ability distribution of the three dimensions are set as 0s and 1s, respectively, assuming they follow a trivariate normal distribution.

#5.5_4. The 3-dimensional GPCM for simple structure is estimated using the "model=spec" and "itemtype='gpcm'."

#5.5_5. Convergence checks are shown. No issue is detected in this application.

#5.5_6. The model parameter estimates are printed. The item slope estimates are shown under "a1," "a2," and "a3." The values under "d1," "d2," and "d3" are, again, not the usual intercept parameter values. They are the values of $d_{ik}^{\star}$ that correspond to the categories of each item (not to the dimension).

```
> coef(mod.md, simplify=T)
$'items'
       a1    a2    a3 ak0 ak1 ak2 ak3 d0     d1     d2     d3
V1  1.550 0.000 0.000   0   1   2   3  0  0.538 -0.271 -1.089
V2  0.884 0.000 0.000   0   1   2   3  0 -1.103 -2.372 -3.817
...
V7  0.000 1.645 0.000   0   1   2   3  0  0.286  0.340 -0.038
V8  0.000 1.730 0.000   0   1   2   3  0 -0.270 -1.501 -3.234
...
```

```
V11  0.000  0.000  1.821  0  1  2  3  0  1.675  2.480  2.387
V12  0.000  0.000  1.975  0  1  2  3  0  0.511 -0.162 -1.421

$means
F1 F2 F3
 0  0  0

$cov
      F1    F2  F3
F1 1.000    NA  NA
F2 0.609 1.000  NA
F3 0.434 0.505   1
```

As mentioned in the multidimensional PCM application, the item slope estimates are shown in "a1," "a2," and "a3." The values of "ak1," "ak2," and "ak3" are the values of the scoring function parameters used for GPCM. The output for the multidimensional version of GPCM follows an alternative parameterization that involves the use of the scoring function parameterization. See Equation 3.3_3, where "k" corresponds to the value of a scoring function parameter for the category *k*.

The estimated covariances for the three latent trait dimensions are equal to the correlation values because of the fixed unit variance model identification. The correlation between dimensions 2 and 3, for example, is estimated to be 0.505 in this application.

To convert the values of the $d_{ik}^{\star}$ (Equation 5.5_3) estimated in "mirt" into the usual intercept d_{ik} (Equation 5.5_2), the steps that follow may be taken. See also #5.4_10 through #5.4_14 for details.

```
dd.mirt<-coef(mod.md, simplify=T)$'items'[,9:11]
d1<- dd.mirt[,1]
d2<-dd.mirt[,2]-(dd.mirt[,1])
d3<- dd.mirt[,3]-(dd.mirt[,2])
dd.estim<-cbind(d1,d2,d3)
round(dd.estim, 3)
```

The "dd.mirt" contains the extracted values of $d_{ik}^{\star}$. These are converted into the usual intercept values of d_{ik}, "d1," "d2," and "d3." The intercept values are combined together in the R object "dd.estim." Lastly the content of the "dd.estim" is printed and rounded to three decimal places, which is shown here.

```
> round(dd.estim, 3)
       d1     d2     d3
V1  0.538 -0.808 -0.819
V2 -1.103 -1.270 -1.444
...
V11 1.675  0.805 -0.092
V12 0.511 -0.673 -1.258
```

The values of the usual intercept d_{ik} can be transformed into the estimates of the item category threshold, b_{ik}, presented in the parameterizations from Equation 5.5_1.

```
aa.estim<-coef(mod.md, simplify=T)$'items'[,1:3]
bb.estim<-data.frame (rbind(
-dd.estim[1:4,]/aa.estim[1:4,1],
-dd.estim[5:8,]/aa.estim[5:8,2],
-dd.estim[9:12,]/aa.estim[9:12,3]) )
names(bb.estim)<-c("b1","b2","b3")
round(bb.estim, 3)
```

The item slope estimates are extracted and stored under "`aa.estim`." Using the relationship, $b_{ik} = -d_{ik}/a_{iq}$, the values of b_{ik} are calculated by dividing the previously obtained "`dd.estim`," which has the values of d_{ik}, by the slope estimates in "`aa.estim`," and then negating that value. This is repeated three times for the subsets of items within each dimension. The estimates of b_{ik} are combined and stored in and object called "`bb.estim`." The column names of "`bb.estim`" are defined as "`b1`," "`b2`," and "`b3`." Lastly, its contents are printed, rounding to three decimals.

```
> round(bb.estim, 3)
        b1     b2    b3
V1  -0.347  0.522 0.528
V2   1.247  1.436 1.633
...
V11 -0.920 -0.442 0.051
V12 -0.259  0.341 0.637
```

#5.5_7. The command prints the original model parameter estimates and their standard errors together. The "`SE`" of the "`d1`," "`d2`," and "`d3`" are the standard errors of the $d_{ik}^{\star}$ (Equation 5.5_3). "`NA`" is printed for the standard error when parameters are not freely estimated.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.
  SE = T) #5.5_8
```

#5.5_8. The EAP estimator is used to estimate person latent trait or ability scores on each dimension. The output for this command includes the three EAP estimates for each person ("`F1`," "`F2`," and "`F3`") and their standard errors ("`SE_F1`," "`SE_F2`," and "`SE_F3`").

```
M2(mod.md) #5.5_9
mod.ud<-mirt(ddd, model=1, itemtype="graded", SE=T) #5.5_10
mod.ud #5.5_11
anova(mod.ud, mod.md) #5.5_12
```

#5.5_9. The "M2*" global model-data fit test, which is the default in "mirt" for polytomous data, is conducted. The *p*-value of the "M2*" test in this application is 0.3167, and the model-data fit is supported. See #3.1_52 for more details of an overall model test M2* and more.

#5.5_10. The unidimensional GPCM is estimated for a relative model comparison and evaluation of the usefulness of the multidimensional structure.

#5.5_11. Convergence check results of the unidimensional model are shown. The unidimensional GPCM fit to the data did not show any particular issues in the estimation.

#5.5_12. The unidimensional GPCM ("`mod.ud`," #5.5_10) and the multidimensional GPCM ("`mod.md`," #5.5_4) are compared using the log-likelihood ratio test and information criteria.

```
> anova(mod.ud, mod.md)
Model 1: mirt(data = ddd, model = 1, itemtype = "graded", SE = T)
Model 2: mirt(data = ddd, model = spec, itemtype = "gpcm", SE = T)
       AIC     AICc    SABIC       HQ      BIC    logLik       X2  df    p
1 25649.76 25655.29 25727.84 25737.82 25880.28 -12776.88      NaN NaN  NaN
2 24626.65 24632.90 24709.60 24720.21 24871.57 -12262.32 1029.118   3    0
```

AIC, AICc, SABIC, and BIC prefer the multidimensional GPCM to the unidimensional model. The log-likelihood ratio test *p*-value is nearly 0 (in the column "`p`"), rejecting the unidimensional GPCM.

Another interesting relative model comparison may be between the multidimensional GPCM and the common-slope multidimensional GPCM, which is equivalent to the multidimensional PCM in terms of the model-data fit. (The previous application of the PCM further restricted the slope of an item to be 1, but here it is not necessarily fixed to 1.)

```
spec <- ' F1 = 1-4
          F2 = 5-8
          F3 = 9-12
          CONSTRAIN = (1-4, a1)
          CONSTRAIN = (5-8, a2)
          CONSTRAIN = (9-12, a3)
          COV = F1*F2*F3 ' #5.5_13
mod.md.ca <- mirt(ddd, model=spec, itemtype='gpcm', SE=T)
  #5.5_14
mod.md.ca #5.5_15
anova(mod.md.ca, mod.md) #5.5_16
```

#5.5_13. An R object "`spec`" is created to specify the item loading structure and to impose the equality constraint across the item slope parameters in each dimension using "`CONSTRAIN`." "`COV=F1*F2*F3`" allows the covariances between the three dimensions to be free parameters. The model

identification is achieved by setting the means and the variances of the three latent trait dimensions as 0s and 1s, respectively (this is done by default in "mirt," so no model definition is included).

#5.5_14. The common-slope multidimensional GPCM is estimated using the "`spec`." It is stored as an object named "`mod.md.ca`."

#5.5_15. Estimation convergence results are shown. No issue was detected in this application. The use of "`coef(mod.md.ca, simplify=T)`" shows the model parameter estimation results. The estimated common slopes for the three dimensions are 1.147, 1.219, and 1.4, respectively.

#5.5_16. The common-slope GPCM (#5.5_14) and the full GPCM (5.5_4) are compared using information criteria and the log-likelihood ratio test.

```
> anova(mod.md.ca, mod.md)
Model 1: mirt(data = ddd, model = spec, itemtype = "gpcm", SE = T)
Model 2: mirt(data = ddd, model = spec, itemtype = "gpcm", SE = T)
       AIC     AICc S   ABIC       HQ      BIC    logLik     X2  df p
1 24703.26 24707.47 24771.57 24780.31 24904.96 -12309.63    NaN NaN NaN
2 24626.65 24632.90 24709.60 24720.21 24871.57 -12262.32  94.61   9   0
```

The *p*-value of the log-likelihood ratio test is nearly 0 (shown in the column "`p`"), rejecting the common-slope GPCM. The values of AIC, AICc, SABIC, and BIC are smaller for the full GPCM than for the common-slope GPCM, indicating the former is preferred.

5.6 Multidimensional graded response model (GRM) application

The GRM is for polytomous, ordinal item response data. The GRM uses a cumulative response probability function to model a response in category k or higher ($k \in \{0,1,2,\ldots,m_i\}$) in the ith item. For more details of the GRM in the unidimensional setting, refer to section 3.4. The cumulative response probability function of responding in category k or higher to ith item according to the GRM for simple structure with Q dimensions ($q = 1, 2, \ldots, Q$) is

$$P\left(X_{ikj} \geq k | \theta_j\right) = \frac{\exp\left[a_{iq}\theta_{jq} + d_{ik}\right]}{1 + \exp\left[a_{iq}\theta_{jq} + d_{ik}\right]}, \qquad (5.6\_1)$$

where X_{ikj} is the jth person's response in category k or higher to the ith item, a_{iq} is the ith item slope parameter for the qth dimension, θ_{jq} is the jth person's latent trait or ability score for the qth dimension, and d_{ik} is the intercept parameter that is related to the category boundary parameter of the ith item for category k or higher, i.e., $d_{ik} = -a_{iq}b_{ik}$, where b_{ik} is the category boundary parameter. Note again that the slope and intercept parameterization is common in IRT multidimensional modeling.

```
ddd<-read.table("c5_MIRT_GRM.dat") #5.6_1
dim(ddd) #5.6_2
apply(ddd,2,table) #5.6_3
```

#5.6_1. The data file "c5_MIRT_GRM.dat" is read into R and stored as "ddd." The data file has a space-delimited format, where the column variables are item responses. The polytomous item responses to 18 items are assumed to be from an intentionally 3-dimensional test, where items 1 through 6 measure the first dimension, items 7 through 12 measure the second dimension, and items 13 through 18 measure the third. Responses were measured on a four-point scale, with the options 0, 1, 2, or 3. The three dimensions are assumed to be correlated. The default variable (item) names are "V1," "V2," . . . , "V18."

#5.6_2. The size of the data set shows the number of rows, or test-takers, is 800, and the number of columns, or items, is 18.

#5.6_3. Frequency analysis of all items is conducted. The results show that each item has four categories scored 0, 1, 2, and 3, and that there were no null, or unused, categories. If the frequency analysis of a single item, for instance, item 1, is desired, "`table(ddd$V1)`" can be used.

```
spec <- ' F1 = 1-6
          F2 = 7-12
          F3 = 13-18
          COV = F1*F2*F3' #5.6_4
mod.md <- mirt(ddd, model=spec, itemtype='graded', SE=T)
  #5.6_5
mod.md #5.6_6
extract.mirt(mod.md, what="time") #5.6_7
coef(mod.md, simplify=T) #5.6_8
coef(mod.md, printSE=T) #5.6_9
```

#5.6_4. A 3-dimensional simple structure model is specified. Items 1 through 6 load on dimension 1, items 7 through 12 load on dimension 2, and items 13 through 18 load on dimension 3. "`COV = F1*F2*F3`" allows the correlations (or covariances) between the three dimensions to be freely estimated. When utilizing the GRM in the "mirt" package, the variance of the ability distribution on each dimension is fixed to 1, so the correlation between dimensions is equal to the covariance.

#5.6_5. The GRM with the 3-dimesional simple structure is estimated using the "`model=spec`" and "`itemtype='graded'`."

#5.6_6. Convergence check results are shown. No particular issue was present in this application.

#5.6_7. The time spent for the model estimation is printed. For this 3-dimesional GRM, about 68 seconds were taken with the author's personal computer for this application. Depending on a user's computer, the data size (the number of items and the sample size), and the model complexity (e.g., the number of dimensions), the estimation time will vary.

#5.6_8. The estimates of the GRM item parameters and the covariances between the three dimensions' ability distributions are shown. The model identification constraint for this simple structure IRT modeling were to set the population ability distribution mean and variance for each dimension to be 0 and 1, respectively. Thus, the covariance estimates are equivalent to correlation estimates. As with other multidimensional models, the IRT parameterizations of the item parameters cannot be printed directly using "`IRTparms=T`."

```
> coef(mod.md, IRTparms=T, simplify=T)
$'items'
        a1    a2    a3     d1     d2     d3
V1   1.706 0.000 0.000  1.376  1.065  0.148
V2   0.916 0.000 0.000  0.064 -0.753 -1.319
...
V6   2.422 0.000 0.000  1.188 -0.379 -0.913
V7   0.000 1.467 0.000  0.516 -0.018 -1.038
V8   0.000 1.375 0.000  1.516  1.039  0.097
...
V12  0.000 1.958 0.000  1.157  0.824 -0.521
V13  0.000 0.000 2.988  0.503 -0.913 -1.385
V14  0.000 0.000 2.128 -1.131 -1.535 -1.946
...
V18  0.000 0.000 1.414  0.456  0.002 -0.409

$means
F1 F2 F3
 0  0  0

$cov
      F1    F2 F3
F1 1.000    NA NA
F2 0.660 1.000 NA
F3 0.532 0.464  1
```

The slope estimates are shown under "`a1`" for items loading on dimension 1 (i.e., "`V1`" through "`V6`"), "`a2`" for items loading on dimension 2 (i.e., "`V7`" through "`V12`"), and "`a3`" for items loading on dimension 3 (i.e., "`V13`" through "`V18`"). There are four categories, so three category boundary intercept parameters (d_{ik}) are estimated for each item (note, these are not estimated for each of the three dimensions, and there is one set of intercept parameters for each item).

In the unidimensional GRM it was mentioned that the category boundary parameter estimates are always ordered in an increasing manner; the opposite is true for the boundary intercept parameter estimates in the multidimensional setting, i.e., $d_{i1} \geq d_{i2} \geq \ldots \geq d_{im_i}$. This is because the multidimensional modeling of the GRM utilizes the positive intercept form. To be consistent with the unidimensional modeling, users may take the negative values of the d_{ik} estimates (i.e, $-1 \times \hat{d}_{ik}$) as the GRM intercept parameters. The estimated correlations among the three dimensions are shown under "`$cov`." Dimensions 1 and 2 shows the highest correlation, $r_{12} = 0.660$.

#5.6_9. The point estimates of the model parameters and their standard errors are requested. See #5.1_7 as well.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.SE = T) #5.6_10
```

#5.6_10. The EAP person latent trait or ability, score estimates are estimated and printed using the "`fscores()`" command. Because there are three dimensions, each person has three ability estimates (and corresponding standard errors), one for each dimension. The ability estimates are shown under the column names "`F1`," "`F2`," and "`F3`." See #5.1_8 as well.

```
M2(mod.md) #5.6_11
mod.ud<-mirt(ddd, model=1, itemtype="graded", SE=T) #5.6_12
mod.ud #5.6_13
anova(mod.ud, mod.md) #5.6_14
```

#5.6_11. An absolute model-data fit test, "M2⋆," is conducted for the polytomously scored item response data set. See #3.1_52 for more details of the overall model test M2⋆ and more. The p-value of the "M2⋆" in this application is 0.728, and the fitted model is supported under $\alpha = 0.05$.

#5.6_12, #5.6_13, and #5.6_14. If model comparisons between the unidimensional and 3-dimension simple structures is desired, the log-likelihood ratio test and information criteria, such as AIC, AICc, BIC, and SABIC, can be utilized. The unidimensional GRM (#5.2_12) is fitted (and the convergence results are printed with #5.2_13). Then the unidimensional GRM is compared with the multidimensional GRM (defined in #5.6_5), using the "`anova()`" command (#5.2_14).

All information criteria values are smaller for the 3-dimensional GRM, indicating the multidimensional model is preferred to the unidimensional model. Also, the log-likelihood ratio test p-value was virtually 0, indicating the rejection of the unidimensional model under $\alpha = 0.05$. When data are virtually unidimensional (i.e., the dimension correlation is very high), fitting a simple structure multidimensional model is an overfitted model and could cause model estimation convergence issues.

```
> anova(mod.ud, mod.md)
Model 1: mirt(data = ddd, model = 1, SE = T, itemtype = "graded")
Model 2: mirt(data = ddd, model = spec, itemtype = "graded", SE = T)
       AIC     AICc    SABIC       HQ      BIC    logLik      X2  df   p
1 31270.99 31285.45 31379.64 31400.56 31608.28 -15563.50     NaN NaN NaN
2 30460.35 30476.10 30573.53 30595.32 30811.70 -15155.18 816.639   3   0
```

Another model comparison of interest may be to compare the multidimensional GRM with the common slope multidimensional GRM. In the common

slope multidimensional model, the slope parameters of all item clusters within each dimension are constrained to be equal (i.e., $a_{iq} = a_q$ in Equation 5.4_1). For this situation, items 1 through 6 are constrained to have a common slope parameter, items 7 through 12 are constrained to have a common slope parameter, and items 13 through 18 are constrained to also have a common slope parameter. The intercept parameters of items are free to vary as before. In the full multidimensional GRM, 18 slope parameters are estimated, while only three slope parameters are estimated in the common slope multidimensional GRM for simple structure.

```
spec<- ' F1 = 1-6
         F2 = 7-12
         F3 = 13-18
         CONSTRAIN = (1-6, a1)
         CONSTRAIN = (7-12, a2)
         CONSTRAIN = (13-18, a3)
         COV = F1*F2*F3 ' #5.6_15
mod.md.ca<-mirt(ddd, model=spec, itemtype="graded", SE=T)
  #5.6_16
mod.md.ca #5.6_17
anova(mod.md.ca, mod.md) #5.6_18
```

#5.6_15. The 3-dimensional simple structure model with the equal slope constraints are specified. The covariance between the ability distributions of each dimension are free to vary.

#5.6_16. The common slope, simple structure, 3-dimensional GRM is estimated using the syntax "`spec`" created by #5.6_15.

#5.6_17. Convergence check results are printed. No particular issue was detected for this application.

#5.6_18. The common slope GRM (#5.6_16) and the full GRM (#5.6_5) are compared using the log-likelihood ratio test and information criteria.

```
> anova(mod.md.ca, mod.md)
Model 1: mirt(data = ddd, model = spec, itemtype = "graded", SE = T)
Model 2: mirt(data = ddd, model = spec, itemtype = "graded", SE = T)
       AIC     AICc    SABIC       HQ      BIC    logLik      X2  df   p
1 30675.33 30685.24 30765.88 30783.31 30956.41 -15277.67     NaN NaN NaN
2 30460.35 30476.10 30573.53 30595.32 30811.70 -15155.18 244.983  15   0
```

The *p*-value of the log-likelihood ratio test is virtually 0, which is shown in the second row and the last column under "p." The smaller model, which is the common slope GRM, is rejected under $\alpha = 0.05$. The values of AIC, AICc, SABIC, and BIC are all smaller for the full GRM than they are for the common slope GRM by sizable differences, indicating the full GRM fits the data better than the common slope GRM.

5.7 IRT modeling with high dimensionality

The complexity and the time of estimating an IRT model depends on several factors including the number of items, sample size, the number of parameters, parameterization of a model, and the number of dimensions. When an IRT model is multidimensional, the number of dimensions becomes a major factor affecting the complications associated with estimating the model. As the number of dimensions in an IRT model increases, the full information estimation typically employed in IRT analyses becomes more and more inefficient because of the numerical integration involved in the approximation of the Q-dimensional space in popular MML EM approaches. With the use of all default settings embedded in the default MML EM estimation method in "mirt," the estimation of an IRT model whose dimension is greater than three or four is not efficient, and it becomes practically intolerable or virtually impossible to estimate.

Cai (2010) introduced the Metropolis-Hastings Robins-Monro (MHRM) method for high-dimensional IRT model estimation, which avoids the numerical integration approach. Another alternative is the adaptive quadrature method (Schilling & Bock, 2005). In addition to the MHRM, "mirt" is equipped with the stochastic EM, Monte Carlo EM, and quasi-Monte Carlo EM estimation methods, which are also potential candidates for a high-dimensional IRT model estimation. Although some studies exist regarding these less commonly used estimation methods (e.g., Lin & Paek, 2016), further investigations of these optional methods in "mirt" are necessary for routine analysis in practice.

In this section, an application of the MHRM for a high-dimensional IRT model is done with 8-dimensional data using the 2PLM for dichotomous responses. The more traditional MML EM approach for this high-dimensional modeling is virtually impossible. However, the point estimates of the model parameters can be effectively and efficiently obtained using the MHRM option in "mirt."

```
ddd<-read.table("c5_MIRT_highdim.dat") #5.7_1
dim(ddd) #5.7_2
```

#5.7_1. The data file "c5_MIRT_highdim.data" has a space-delimited format, which can be read into R with ease using the "`read.table()`" command. The dichotomous item responses to 40 items are assumed to be from an 8-dimensional test with the following loading structure, where five items load on each dimension: items 1–5 load on dimension 1; items 6–10 load on dimension 2; items 11–15 load on dimension 3 . . . ; items 31–35 load on dimension 7; and items 36–40 load on dimension 8. The eight dimensions are assumed to be correlated.

#5.7_2. The size of the data set is provided. The number of rows, or test-takers, is 2000, and the number of columns, or items, is 40.

```
spec <- ' F1 = 1-5
          F2 = 6-10
          F3 = 11-15
          F4 = 16-20
          F5 = 21-25
          F6 = 26-30
          F7 = 31-35
          F8 = 36-40
          COV = F1*F2*F3*F4*F5*F6*F7*F8 ' #5.7_3
mod.md.without.se <- mirt(ddd, spec, itemtype='2PL',
method = 'MHRM') #5.7_4
mod.md.without.se #5.7_5
extract.mirt(mod.md.without.se, what='time') #5.7_6
coef(mod.md.without.se, simplify=T) #5.7_7
```

#5.7_3. The 8-dimensional item loading structure is specified. Also, the 8(8 – 1)/2 = 28 covariances between the eight dimensions are specified as free parameters. The means and the variances of the ability distributions on each dimension are fixed as 0s and 1s, respectively, for model identification by default (no additional specifications are needed for this).

#5.7_4. The 8-dimensional 2PLM for simple structure is estimated using the "`spec`," defined in #5.7_3, and the "`method = 'MHRM'`" is used. Here "`SE=T`" is not specified in the "`mirt()`" function, which means the error covariance matrix estimation is not requested.

#5.7_5. The convergence check is shown. Because "`SE=T`" is not requested, there is no second-order test check available in this case.

#5.7_6. The time of the model estimation is extracted. Typically, estimating only the point estimates of model parameters reduces the run time of the estimation; however, due to the high-dimensional structure, this estimation has a longer duration.

```
> extract.mirt(mod.md.without.se, what='time')
  TOTAL: Data MH_draws Mstep   SE  Post
  120.18 0.14    23.11 33.89 0.00 62.44
```

The time for the model calibration to finish with the MHRM estimation method was about 120 seconds (or two minutes) using the authors computer. The time for "`SE`" is 0 because the option for the standard error was not used in this run.

#5.7_7. The point estimates of the model parameters are shown. Again, because the option for calculating the error covariance was not specified, "`coef(mod.md.without.se, printSE=T)`" does not print the standard errors, even though "`printSE=T`" is included in the function. The item slope estimates for each dimension are show shown under "`a1`,"

"a2," ..., "a8." The intercept estimates $\left(\hat{d}_i\right)$, which are related to item easiness, are shown under "d." For the eight dimensions, 28 unique covariances of the population ability latent traits are estimated and shown under "$cov." They are equivalent to the correlations since the variances of the ability distributions are fixed as 1 (shown on the diagonal).

```
> coef(mod.md.without.se,simplify=T)
$'items'
       a1    a2    a3    a4    a5    a6    a7    a8      d  g  u
V1  1.784 0.000 0.000 0.000 0.000 0.000 0.000 0.000  1.378  0  1
V2  0.829 0.000 0.000 0.000 0.000 0.000 0.000 0.000  1.005  0  1
V3  1.192 0.000 0.000 0.000 0.000 0.000 0.000 0.000  0.049  0  1
V4  1.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000  0.023  0  1
V5  1.742 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -0.869  0  1
V6  0.000 1.307 0.000 0.000 0.000 0.000 0.000 0.000  1.604  0  1
V7  0.000 1.146 0.000 0.000 0.000 0.000 0.000 0.000  0.644  0  1
V8  0.000 1.748 0.000 0.000 0.000 0.000 0.000 0.000 -0.018  0  1
V9  0.000 1.807 0.000 0.000 0.000 0.000 0.000 0.000 -0.415  0  1
V10 0.000 0.861 0.000 0.000 0.000 0.000 0.000 0.000 -1.461  0  1
...
V36 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.441 -1.382  0  1
V37 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.828 -0.040  0  1
V38 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.685 -0.599  0  1
V39 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.850  0.854  0  1
V40 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.141  1.539  0  1

$means
F1 F2 F3 F4 F5 F6 F7 F8
 0  0  0  0  0  0  0  0

$cov
      F1    F2    F3    F4    F5    F6    F7  F8
F1 1.000    NA    NA    NA    NA    NA    NA  NA
F2 0.655 1.000    NA    NA    NA    NA    NA  NA
F3 0.449 0.522 1.000    NA    NA    NA    NA  NA
F4 0.567 0.593 0.426 1.000    NA    NA    NA  NA
F5 0.578 0.634 0.482 0.622 1.000    NA    NA  NA
F6 0.426 0.513 0.599 0.391 0.473 1.000    NA  NA
F7 0.535 0.643 0.423 0.598 0.614 0.432 1.000  NA
F8 0.589 0.590 0.500 0.570 0.607 0.486 0.568   1
```

The estimated correlations among the eight latent traits ranged between 0.23 and 0.655.

When the error covariance calculation is requested by specifying "SE=T" with the MHRM estimation in "mirt()," the standard error method is changed into "MHRM" where the information matrix is obtained via a stochastic process. From the author's experience with the use of "SE=T" in the MHRM estimation, the second-order test as part of the convergence check with simulated data sets resulted in failure for almost all applications. This does not necessarily mean the point estimates produced by the MHRM method are not good. The point estimates of the "mirt()" function with the MHRM method recovered their true values (for simple structure multidimensionality) reasonably well and showed comparative performance to a

commercial IRT program, flexMIRT (Cai, 2013; Houts & Cai, 2016), in a simulation study done by Lin and Paek (2016). The tendency of failure of passing the second-order test in "`mirt()`" seems to be due more to the difficulty of approximating the information matrix by the stochastic process employed in the "mirt" program.

```
mod.md.se <- mirt(ddd, spec, itemtype='2PL',
  method = 'MHRM', SE=T) #5.7_8
mod.md.se #5.7_9
extract.mirt(mod.md.se, what="converged") #5.7_10
extract.mirt(mod.md.se, what="secondordertest") #5.7_11
```

#5.7_8. The 8-dimensional model is fit to the same data, and the error covariance calculation is requested by specifying "`SE=T`."

#5.7_9. Estimation convergence check results. The printed content contains the information matrix calculation and the second-order test result. In this case, the following message is shown.

```
. . .
Information matrix estimated with method: MHRM
. . .
Second-order test: model is not a maximum, or the
  information matrix is too inaccurate
```

The current application did not pass the second-order test (under the default setting used in the estimation of the 8-dimensional model). Again, this is consistent with the author's experience in using the MHRM with the "`SE=T`" specification. It is likely due to the difficulty of approximating the information matrix via the current stochastic process in "mirt."

#5.7_10 and $5.7_11. Separate extraction of the two convergence checks are shown. The first "`what="converged"`" check shows "`TRUE`," while the second check, "`what="converged",`" prints "`FALSE.`"

```
> extract.mirt(mod.md.se, what="converged")
[1] TRUE
> extract.mirt(mod.md.se, what="secondordertest")
[1] FALSE
```

Precisely speaking, the first-order convergence check verifies that the gradient of the objective function is 0 at the proposed solution. The first convergence check in "mirt" from "`what="converged"`" is not the same as this, but checking if the estimation process is run without any particular issues and stopped when the specified convergence criteria (e.g., estimate difference less than 0.001) are met. Because the first-order test is only a necessary condition for a solution to achieve a local maximum, the second-order test can be conducted to ensure that the objective function near the proposed solution is locally concave.

The second-order test uses the information matrix or Hessian matrix. In the MHRM method, the information matrix is approximated by a stochastic process. The obtained solutions may fail to pass the second-order test because they are poor estimates themselves, because the information matrix in this case is a poor approximation, or because both estimates and the approximation of the information matrix are poor. It seems that in using the MHRM in the "mirt" program, the difficulty of approximating the information matrix contributes more to the failure of the second-order test.

To increase the chance of passing the second-order test and better approximating the information matrix in the use of MHRM in "mirt," increasing the maximum number of draws for the MHRM information approximation, or increasing the maximum number of fixed draws with the additional argument "`SE.type=FMHRM`" may be tried (P. Chalmers, personal communication, November 18, 2018). The current default value of the maximum number of draws in the approximation of the information matrix is 2000. When the number of dimensions is large, requiring a larger number of draws is likely to provide better results that pass the second-order test. For our application, the maximum number of draws is increased to 10,000 and requested by "`technical = list(MHRM_SE_draws=10000)`" in "`Try1`."

```
Try1<-mirt(ddd, spec, itemtype='2PL', method = 'MHRM',
  SE=T,
technical = list(MHRM_SE_draws=10000))
extract.mirt(Try1, what="converged")
extract.mirt(Try1, what="secondordertest")
```

"`Try2`" increases the maximum number of draws to 10,000 and fixes the number of required draws to this 10,000 draws with "`SE.type = 'FMHRM'`."

```
Try2<-mirt(ddd, spec, itemtype='2PL', method = 'MHRM',
  SE=T,
SE.type = 'FMHRM', technical=list(MHRM_SE_draws=10000))
extract.mirt(Try2, what="converged")
extract.mirt(Try2, what="secondordertest")
```

Note that the maximum number of draws in "`Try1`" and "`Try2`" was changed to 10,000, from the default value of 2000. However, in the first try, "`Try1`" still did not succeed in passing the second-order test.

```
> extract.mirt(Try1, what="converged")
[1] TRUE
> extract.mirt(Try1, what="secondordertest")
[1] FALSE
```

In the second try, "`Try2`" passed the second-order test due to a better approximation of the information matrix.

```
> extract.mirt(Try2, what="converged")
[1] TRUE
> extract.mirt(Try2, what="secondordertest")
[1] TRUE
```

Note, an additional combination may be tried, using "`SE.type = 'FMHRM'`" but excluding "`technical=list(MHRM_SE_draws=10000)`." This fixes the number of draws to the default 2000 in using "`SE.type = 'FMHRM'`." This result still does not pass the second-order test.

The use of these options for increasing the number of draws does not always guarantee success. In using the MHRM estimation method in "mirt," however, one should try using these options when the second-order test result shows failure, which is likely to be the case for most of the applications when the current default options are used to fit a higher-order multidimensional structure.

```
coef(Try2, simplify=T) #5.7_12
coef(Try2, printSE=T) #5.7_13
```

#5.7_12. This prints the point estimates of the model parameters. They are the same as those of the previously fitted 8-dimensional models, i.e., without the SE estimated, with the SE estimated, and with the increased number of draws. These modifications only affect the estimations of the standard errors, not the estimates of the parameters themselves. The coefficients of "`mod.md.without.se`" were printed (see #5.7_7 for "`coef(mod.md.without.se, simplify=T)`"), but the same function can be utilized with any of the "mirt" objects.

#5.7_13. The point estimates and their standard errors are printed.

```
> coef(Try2, printSE=T)
$'V1'
       a1 a2 a3 a4 a5 a6 a7 a8     d logit(g) logit(u)
par 1.784  0  0  0  0  0  0  0 1.378     -999      999
SE  0.167 NA NA NA NA NA NA NA 0.099       NA       NA

$V2
       a1 a2 a3 a4 a5 a6 a7 a8     d logit(g) logit(u)
par 0.829  0  0  0  0  0  0  0 1.005     -999      999
SE  0.078 NA NA NA NA NA NA NA 0.065       NA       NA
...
$V40
    a1 a2 a3 a4 a5 a6 a7    a8     d logit(g) logit(u)
par  0  0  0  0  0  0  0 1.141 1.539     -999      999
SE  NA NA NA NA NA NA NA 0.112 0.109       NA       NA

$GroupPars
 MEAN_1 MEAN_2 MEAN_3 ... MEAN_8 COV_11 COV_21 COV_31 ... COV_87 COV_88
par   0      0      0 ...      0      1  0.655  0.449 ...  0.568      1
SE   NA     NA     NA ...     NA     NA  0.062  0.041 ...  0.034     NA
```

The run time comparison of the four approaches shown here ("`mod.md.without.se`" without "`SE=T`" [#5.7_4], with "`SE=T`" [#5.7_8], with "`SE=T`" and the increased maximum possible number of draws for the approximation of the information matrix ["`Try1`"], and with "`SE=T`" and the increased fixed number of draws for the approximation of the information matrix using "`SE.type = 'FMHRM'`" ["`Try2`"]) are printed here. The effect of increasing the maximum number of draws is very clearly shown in the "`FMHRM`" approach. Increasing the maximum number of draws to 10,000 increased the run time of the "`FMHRM`" in "`Try2`" to 28.8 minutes (shown here as 1782.80 seconds), which is a very large increase in the model estimation time compared to the other approaches that have taken between two and four minutes.

```
> extract.mirt(mod.md.without.se, what='time')
 TOTAL:  Data MH_draws Mstep    SE   Post
  120.18 0.14  23.11   33.89  0.00  62.44
> extract.mirt(mod.md.se, what="time") #
 TOTAL:  Data MH_draws Mstep     SE    SE    Post
  208.01 0.14  23.19   34.75  84.88  0.00   63.52
> extract.mirt(Try1, what="time") #
 TOTAL:  Data MH_draws Mstep     SE    SE    Post
  200.26 0.19  22.66   31.39  82.64  0.00   62.14
> extract.mirt(Try2, what="time")
 TOTAL:  Data MH_draws Mstep       SE    SE   Post
 1728.80 0.11  22.87   31.94  1609.95  0.00  60.97
```

Even though the additional specifications take a longer run time, the estimation of a high-dimensional model, as in this case, is still practically feasible in the MHRM estimation method when the MML EM estimation method virtually fails to provide the model parameter estimates and their standard errors due to high dimensionality.

```
Ability <- fscores(mod.md.without.se, method="EAP", full.
  scores=T, full.scores.SE = T, QMC=T) #5.7_14
head(Ability) #5.7_15
```

#5.7_14 and #5.7_15. The person latent trait or ability scores are estimated using the EAP estimator via "`fscores()`." "`QMC=T`," which is an option for the quasi-Monte Carlo integration, was specified in this 8-dimensional modeling to make the computation of the person ability scores more efficient. The results (part of which is shown using "`head(Ability)`") contain the eight latent trait scores for each individual and their standard errors. The first eight columns are for the latent trait scores, titled "`F1`," "`F2`," etc. The remaining columns contain the standard errors, titled "`SE_F1`," "`SE_F2`," etc.

5.8 IRT modeling for within-item multidimensionality

The previous multidimensional structures are all simple structure, where each item loads on a single dimension. When some or all items load on more than a single dimension, it is sometimes called within-item multidimensionality (Adams et al., 1997). Suppose that items on a test for mathematical operations tend to be wordy and lengthy. Those items may require reading comprehension skills in addition to knowledge of mathematical operations. One possibility of fitting a latent trait model to data having these types of items is to use a 2-dimensional, within-item multidimensional IRT model, where every item loads onto the two dimensions of mathematical operations and reading comprehension. Under this situation, items are assumed to measure the target dimension and an extra dimension, both of which underlie the test-takers' responses and are required to correctly answer the item.

In this chapter, a 2-dimensional within-item 2PLM is illustrated. The model identification involves not only the reference point and the scale of each dimension, but also a rotational indeterminacy. This means that, in addition to the (typical) constraints of the means and variances, the distribution of abilities on each dimension is fixed to some constants (e.g., 0 and 1, respectively), and the unique parameter solution requires additional constraints. The constraints adopted in this applications are: (1) the means and variances of the distribution of abilities on each dimension set to 0s and 1s, respectively, in the bivariate ability population distributions; (2) the covariances of the two ability dimension set to 0, i.e., that the dimensions are uncorrelated; and (3) the first item slope parameter for the second dimension set to 0. The currently employed set of constraints is for an orthogonal solution. When a within-item multidimensional model is used, attention should be given to the model identification model constraints to ensure unique model parameter solutions.

The IRF for a Q-dimensional within-item multidimensional structure ($q = 1, 2, \ldots, Q$) with dichotomous responses is

$$P\left(X_{ij} = 1 | \theta_{j1}, \theta_{j2}, \cdots, \theta_{jQ}\right) = \frac{\exp\left[\sum a_{iq}\theta_{jq} + d_i\right]}{1 + \exp\left[\sum a_{iq}\theta_{jq} + d_i\right]}, \quad (5.8\_1)$$

where X_{ij} is the jth person's response to the ith item (0 = incorrect/no; and 1 = correct/yes), θ_{jq} is the jth person's latent trait or ability score for the qth dimension, d_i is the ith item intercept related to item easiness, and a_{iq} is the ith item slope parameter for the qth dimension. The summation in the prior equation is over all $q = 1, 2, \ldots Q$ dimensions. Again, the model identification employed for the 2-dimensional modeling is $(\theta_{j1}, \theta_{j2})' \sim$ *Bivariate* $N(\mu, \Sigma)$ where $\mu = (0, 0)$, Σ is a 2×2 identity matrix, and $a_{i2} = 0$.

IRT modeling is sometimes called ordinal response item factor analysis. IRT uses full information estimation methods, such as the MML, while factor analysis typically uses what are called limited information estimation methods, where a covariance or correlation matrix is the basic input data. We do not pursue the relationship between factor analysis and IRT here, however; readers are referred to, for example, Paek, Cui, Gubes, and Yang (2017) for more details of the relationship between factor analysis and IRT models.

In the Equation 5.8_1, each item has one intercept, d_i, and Q number of item slope parameters, a_{iq}. Reckase (1985, 1991, 2009) proposed a single index of item difficulty and a single index of item discrimination in multidimensional modeling. The single index of the multidimensional item discrimination ($MDISC_i$) is

$$A_i = MDISC_i = \sqrt{\sum_{q=1}^{Q} a_{iq}^2} \tag{5.8_2}$$

The $MDISC_i$, or A_i, represents the ith item's overall multidimensional discrimination. In unidimensional IRT modeling, the discrimination is equal to 0.25 times the slope of the IRF when the IRF becomes steepest. In multidimensional modeling, A_i is similarly defined as the value related to the slope of the multidimensional item response surface in the direction where the item response surface takes the steepest slope. The direction from the origin of the Q-dimensional latent trait space in the steepest direction is quantified by

$$\alpha_{it} = \cos^{-1}\left(a_{it} / \sqrt{\sum_{q=1}^{Q} a_{iq}^2} \right) \tag{5.8_3}$$

α_{it}, the value of the arccosine, is the angle of the ith item from the reference dimension, t, where $t = 1, 2, \ldots, Q$. In this 2-dimensional example, $Q = 2$ ($q = 1$ or 2), one can choose either the first dimension ($t = 1$) or the second dimension ($t = 2$) to act as the reference. The directions of the items can be expressed as angles of items from the first dimension if $t = 1$ is chosen. In an orthogonal coordinate system with $Q = 2$, α_{it} ranges from 0 degree to 90 degrees when $a_{it} > 0$. The smaller the angle, closer to 0 degrees, the more strongly the item is associated with the first dimension. The larger the angle, closer to 90 degrees, the more strongly the item is associated with the second dimension.

The single index of the multidimensional item difficulty is defined as

$$B_i = MDIFF_i = -d_i / A_i, \tag{5.8_4}$$

B_i is obtained by the negative of the item intercept divided by A_i. This is similar to the definition of the unidimensional difficulty, which is the negative of the unidimensional intercept divided by the discrimination parameter ($b_i = -d_i/a_i$). The interpretation of B_i is the same as the unidimensional b_i at the direction specified by α_{it} in Equation 5.8_3. Readers are referred to Reckase (2009) for more details of these multidimensional item parameters.

```
ddd<-read.table("c5_MIRT_within-item.dat") #5.8_1
dim(ddd) #5.8_2
```

#5.8_1. The "c5_MIRT_within-item.dat" file contains dichotomous item responses to 30 items in a space-delimited format. It is read into R and saved under the R object "ddd." The 30 item responses are assumed to be from a 2-dimensional test where each item has a 2-dimensional structure, i.e., each item loads on two dimensions. The default variable (item) labels are "V1," "V2," . . . , "V30."

#5.8_2. The number of rows, or test-takers, is 2018, and the number of columns, or items, is 30.

```
spec <- 'F1 = 1-30
         F2 = 1-30
         START=(1,a2,0)
         FIXED=(1,a2)' #5.8_3
mod.md <- mirt(ddd, spec, itemtype='2PL', SE=T) #5.8_4
mod.md #5.8_5
extract.mirt(mod.md, what="time") #5.8_6
coef(mod.md, simplify=T) #5.8_7
coef(mod.md, printSE=T) #5.8_8
```

#5.8_3. The item loading structure is specified. Every item loads onto each of the two dimensions ("F1" and "F2"). The first item's slope parameter on the second dimension ("a2") is fixed as 0 as part of the model identification using the "START" and "FIXED" definitions.

#5.8_4. The 2-dimensional within-item 2PLM is estimated according to the model "spec" defined by #5.8_3.

#5.8_5. Convergence check results are printed. No particular issues or warnings were found.

#5.8_6. The run time for the model estimation is extracted and printed. The author's computer took about one minute.

#5.8_7. The point estimates of the item slope and the intercept parameters are printed. Again, $a_{12} = 0$, $\mu = (0, 0)$, and Σ is the 2×2 identity matrix for the model identification. The item slope estimates for the first dimension are higher in general than those for the second dimension. The means of the item slope estimates are 1.46 and 0.50 for the first and the second dimensions, respectively.

```
> coef(mod.md, simplify=T)
$'items'
        a1    a2      d  g  u
V1   1.421 0.000  1.430  0  1
V2   0.845 0.189  0.707  0  1
V3   1.044 0.584 -0.068  0  1
...
V29  1.939 0.263  0.855  0  1
V30  1.127 0.387  1.386  0  1

$means
F1 F2
 0  0

$cov
   F1 F2
F1  1 NA
F2  0  1
```

These indicate that the items are more strongly associated with the first dimension than the second. In the context of the mathematical operations test, the ability to perform the math operation influences examinees' responses more strongly than another potential dimension such as reading comprehension.

An alternative approach to estimating this within-item multidimensional model is to use the default setting in "`mirt()`" without creating a syntax for the item loading structure and model identification specifications.

```
mod.md.default<-mirt(ddd, model=2, SE=T)
```

Here, "`model=2`" in in the "`mirt()`" command indicates the estimation of the 2-dimensional within-item model (instead of "`model=1`"). Though "`itemtype= "2PL"`" is not included, the 2PLM is the default model selected for dichotomous data when an item type is not specified. The model identification here is the same as before, except the default item whose slope is fixed to 0 is the last item on the second dimension.

The item parameter estimates can be viewed using "`coef(mod.md.default, simplify=T)`." Under the default setting, the last item slope parameter on the second dimension is fixed as 0 instead of the first item slope on the second dimension. If this default setting estimation is done, all slope estimates on the first dimension are negative (and many slope estimates on the second dimension are also negative). Due to the exploratory nature in this within-item multidimensional modeling (i.e, each item loading onto all dimensions), it is possible that the solutions for the slope parameters in the estimation process to converge in the opposite direction, so that all slope estimates are negative. Suggestions to cope with this kind of situation include the following: selecting different item(s) for the model identification item(s), which could show better behavior and clearer direction, e.g., item 1 loading on dimension 1 very strongly and monotonically increasing in the dimension; using a different set of starting values; using both a different item(s) for fixing its (or their) slope(s) and different starting values from those used in the default setting; and consideration of using reasonable prior distributions for the slope parameters (e.g., lognormal distribution). In this application, item 1 was selected instead of the default choice of the last item as part of the model identification.

#5.8_8. Item parameter point estimates and their standard errors are printed.

The multidimensional item discrimination, MDISC, and the multidimensional item difficulty, MDIFF, are obtained with the following steps.

```
isv<- coef(mod.md, simplify=T)$'item'[,1:3] #5.8_9
angle.degree<- acos (isv[,1]/sqrt(isv[,1]^2+isv[,2]^2))
  *180/pi #5.8_10
cbind(angle.degree, MDISC(mod.md), MDIFF(mod.md)) #5.8_11
```

#5.8_9. The estimates of item parameters (slopes for dimensions 1 and 2 and intercepts) are extracted separately and stored under "`isv`." The first and the second columns in "`isv`" are the item slopes for dimensions 1 and 2, respectively. The third column is the intercept value.

#5.8_10. The direction from the first dimension in terms of degrees in angle is calculated (see Equation 5.8_3). "`acos()`" in R produces the angle in terms of radians. Using $2\pi \times$ radian = 360 degrees (or, as presented, multiplying the result of `acos()` by 180 and dividing by π), the angle α_{it} for the ith item when $t = 1$ is calculated in degrees.

#5.8.11. "`MDISC()`" and "`MDIFF()`" are functions in "mirt" which calculate the MISC and MDIFF values. The degree of the item angle, MDISC, and MDIFF are combined together and shown in the first, the second, and the third columns.

```
> cbind(angle.degree, MDISC(mod.md), MDIFF(mod.md))
   angle.degree                  MDIFF_1
V1     0.000000  1.4214149  -1.00599430
V2    12.631987  0.8660065  -0.81672575
V3    29.213503  1.1957590   0.05684067
...
V29    7.721986  1.9569291  -0.43681456
V30   18.959334  1.1917122  -1.16302182
```

The first item's angle is 0 degrees, implying that it perfectly aligns with the first dimension. This is because it loads only on the first dimension, and its value is fixed as 0 on the second dimension, as part of the model identification. Item 2 has an angle of 12.63 degrees from dimension 1, while item 3 has an angle of 29.2 degrees. Item 2 is more closely aligned to the first dimension than is item 2. An angle of 45 degrees implies that the item measures both dimensions equally. In the Rasch model case, $a_{iq} = 1$, and α_{iq} for all items is $\cos^{-1}\left(1/\sqrt{1^2+1^2}\right) = 45$, which means that items are assumed to measure both dimensions equally.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.
  SE = T) #5.8_12
```

#5.8_12. The person latent trait or ability scores are estimated using the EAP estimator. Each test-taker has two ability scores, one for each dimension. The ability estimate on the first dimension is listed under "`F1`," and the ability estimate on the second dimension is listed under "`F2`." The output also has the standard errors of the EAP estimates (in the columns of "`SE_F1`" and "`SE_F2`" for the standard error of the ability estimate on dimensions 1 and 2, respectively).

References

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

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

Cai, L. (2013). *flexMIRT version 3.51: Flexible multilevel multidimensional item analysis and test scoring* (Computer software). Chapel Hill, NC: Vector Psychometric Group.

Houts, C. R., & Cai, L. (2016). *flexMIRTR user's manual version 3.5: Flexible multilevel multidimensional item analysis and test scoring*. Chapel Hill, NC: Vector Psychometric Group.

Lin, Z., & Paek, I. (2016). *A comparison of recently implemented point and standard error estimation procedures for multidimensional IRT modeling between R package 'mirt' and flexMIRT*. Presented at the 2016 Modern Modeling Methods Conference, Storrs, CT.

Paek, I., Cui, M., Gubes, N. O., & Yang, Y. (2017). Estimation of an IRT model by Mplus for dichotomously scored responses under different estimation methods. *Educational and Psychological Measurement*, *78*, 569–588.

Reckase, M. D. (1985). The difficulty of test items that measure more than one ability. *Applied Psychological Measurement*, *25*, 193–204.

Reckase, M. D. (1991). The discriminating power of items that measure more than one dimension. *Applied Psychological Measurement*, *15*, 361–373.

Reckase, M. D. (2009). *Multidimensional item response theory*. New York, NY: Springer.

Schilling, S., & Bock, R. D. (2005). High-dimensional maximum marginal likelihood item factor analysis by adaptive quadrature. *Psychometrika*, *70*, 533–555.

6

MULTIDIMENSIONAL IRT FOR BIFACTOR STRUCTURE

The bifactor structure is a specific design of within-item multidimensionality. In bifactor modeling, all items load on a general or primary factor, and subsets of items each load on their own specific factor (Gibbons & Hedeker, 1992; Gibbons et al., 2007). Take a mathematics test, for example. Suppose that the test has three subdomains, or constructs, where items within each subdomain measure one of three mathematics domains: algebra, geometry, and trigonometry. However, all items across the three subdomains measure the general or primary factor of mathematics. If one wants to model the general or primary factor as a single construct while modeling each subdomain's specific factor, the bifactor modeling suits the situation.

Another example where the bifactor modeling could be considered is a reading comprehension test. In a reading test, subsets of items are clustered together with a common passage. If a reading test has three passages and each passage has five items, then, the general or primary reading factor is assumed to be associated with all 15 items, and the three subsets of five items, corresponding to items specific to each passage, are associated with one of the three specific factors. Thus, every item loads on the general or primary dimension and on a single specific dimension, which results in a 2-dimensional structure for every item. The general or primary factor and the specific factors are assumed to be uncorrelated. And, the specific factors are assumed to be uncorrelated to each other since, after accounting for their common association with the general or primary factor, the "leftover" association with the specific factor is unrelated across the specific factors. This independence of the factors is part of the model identification constraints in the bifactor modeling. The illustrations of fitting bifactor models in this chapter focus on specifying the correct model and on estimating the item parameter, population parameter, and the individual latent trait scores using the package "mirt." We start with the assumption that "mirt" is installed and loaded onto the current R session and that the working directory has been set. See #2.1_38, #2.1_39, and #2.1_3 for details. Again, the data file for analysis should be in the folder specified in the working directory.

```
install.packages("mirt")
library(mirt)
setwd("c:/mystudy/RIRT/")
```

For the estimation of a bifactor model, "mirt" provides a special function for fitting the bifactor model structure that provides a very efficient estimation algorithm. The "`bfactor()`" function uses a dimension reduction technique, which boosts the speed of the estimation. The evaluation of the integrals in the objective function that has Q number of dimensions is analytically simplified based on the bifactor structure, achieving a great amount of computational efficiency in the use of the dimension reduction technique (see, e.g., Cai (2010) and Rijmen, Vansteelandt, and De Boeck (2008) for more details of the dimension reduction technique). In a 4-dimensional bifactor modeling, where there is one general or primary dimension and three specific dimensions, applying the "`mirt()`" function with the default EM option is very inefficient in the model estimation and may not be practically tolerable for running the model. The use of "`bfactor()`" makes the estimation of the 4-dimensional model more efficient and reduces the model run time. The current version of "`bfactor()`" works for the estimation of the bifactor 2PLM for dichotomous data and the bifactor GRM for polytomous data. It is possible to use the 3PLM IRF when modeling the bifactor structure, where the values of the lower asymptotes are specified by the user and are not freely estimated.

An alternative approach to estimate a high-dimensional bifactor model (e.g., the total number of dimension > 3) is to use MHRM with a provision of the bifactor structure information using the "`mirt()`" function, similar to the technique of section 5.7. Using the "`mirt()`" function to fit a bifactor model requires more specific information from the user for the item loading structure, possibly the covariance structure between dimensions, and the appropriate choice of an estimation method. There exists some evidence that the "`bifactor()`" in "mirt," which uses the dimension reduction technique, produces very comparable performance with the commercial software, flexMIRT (Cai, 2013), for the estimation of a bifactor 2PLM model (Lin & Paek, 2016).

In this chapter, the use of "`bfactor()`" is illustrated for dichotomous data with 2PLM bifactor modeling, for polytomous data with the GRM bifactor modeling, for the testlet model specification and its estimation, and for the two-tier modeling. Examples are also given for fitting the Rasch and 3PLM for dichotomous data and GPCM for polytomous data using the "`mirt()`" function for the bifactor modeling. Readers interested in applying the MHRM or other high-dimensional model estimation methods to a bifactor modeling for other IRT models are referred to Chapter 5, especially section 5.7.

6.1 Bifactor modeling for dichotomous item responses

The IRF of the 3PLM bifactor model for dichotomous responses is

$$P\left(X_{ij}=1|\,\boldsymbol{\theta}_{jp},\boldsymbol{\theta}_{js}\right)=g_i+\left(1-g_i\right)\frac{\exp\left[a_{ip}\boldsymbol{\theta}_{jp}+a_{is}\boldsymbol{\theta}_{js}+d_i\right]}{1+\exp\left[a_{ip}\boldsymbol{\theta}_{jp}+a_{is}\boldsymbol{\theta}_{js}+d_i\right]}, \qquad (6.1\_1)$$

where a_{ip} is the ith item slope for the primary dimension, a_{is} is the ith item slope for the sth specific dimension ($s = 1, 2, \ldots, S$), θ_{jp} is the jth person latent trait score for the primary dimension, θ_{js} is the jth person latent score for the sth specific dimension, d_i is the ith item intercept parameter, and g_i is the ith item pseudo-guessing parameter. When $g_i = 0$, the 2PLM bifactor model is obtained. When $g_i = 0$, $a_{ip} = 1$, and $a_{is} = 1$ on the associated specific dimension ($a_{is} = 0$ on the other specific dimensions), the Rasch style bifactor model is acquired.

When the data are dichotomous, the 2PLM model is assumed in "`bfactor()`." As mentioned earlier, the "bfactor()" function does allow the user to fit the 3PLM IRF by user-fixed values of the lower asymptotes. In the bifactor modeling, each item loads onto two dimensions, a primary dimension and a specific dimension, or possibly only on a primary dimension.

```
ddd<-read.table("c6_Bif_dicho.dat") #6.1_1
dim(ddd) #6.1_2
```

#6.1_1. The 21-item dichotomous item response data file "c6_Bif_2PL.dat," which has space-delimited format, is read into R and stored as the object named "`ddd`." The data are assumed to follow a bifactor structure where all items load on the general, or primary, dimension. There are three specific dimensions, where items 1 through 7 are associated with only the first specific dimension, items 8 through 14 are associated with only the second specific dimension, and items 15 through 21 are associated with only the third specific dimension.

#6.1_2. The size of the data set is printed. There are 899 rows, or test-takers, and 21 columns, or item variables.

2PLM bifactor model application with the "`bfactor()`" function

```
spec<-c(rep(1,7),rep(2,7),rep(3,7)) #6.1_3
mod.md<-bfactor(ddd, model=spec, SE=T) #6.1_4
mod.md #6.1_5
extract.mirt(mod.md, what="time") #6.1_6
coef(mod.md, simplify=T) #6.1_7
coef(mod.md, printSE=T) #6.1_8
```

#6.1_3. Item loading structure for the specific dimensions is specified using numeric values. This is necessary for using the "`bfactor()`" function. This creates a numeric vector of values to indicate which specific factor each item is associated with. (No specification is needed for the general factor, since the bifactor structure assumes all items are associated with the general factor.) The "`rep()`" function is used to more easily indicate the specific factor, where the first argument is the value to be repeated and the second argument is the number of times the values are to be repeated.

The "`rep(1,7)`" indicates that "1" is repeated "7" times, or that items 1 through 7 load onto the first specific dimension. "`rep(2,7)`" indicates that "2" is repeated "7" times, or that the next seven items, i.e., items 8 through 14, load onto the second specific dimension. "`rep(3,7)`" indicates that "3" is repeated "7" times, or that the next seven items, i.e., items 15 through 21, load onto the third specific dimension. Alternatively, one could use "`"spec<-c(1,1,1,1,1,1,2,2,2,2,2,2,2, 3,3,3,3,3,3,3)"`."

Items do not need to load on specific dimensions in this orderly fashion. For example, for a six-item test, assume items 1 and 2 load on the second specific dimension, item 3 loads on the first specific dimension, item 4 loads onto the second specific dimension, and the last two items, i.e., items 5 and 6, load on the first specific dimension. Then "`spec<-c(2,2,1,2,1,1)`" would be used. No other syntax is required other than this specific factor definition in the use of the "`bfactor()`" function in specifying the item loading and dimension specification.

#6.1_4. The 4-dimensional bifactor 2PLM having one primary and three specific dimensions is estimated using the "`bfactor()`" function according to the "`model=spec`" that was defined in #6.1_3. The parameter standard error computation is also requested with "`SE=T`." Because the "`itemtype`" is not specified, the 2PLM is used, by default, for the dichotomous data.

#6.1_5. Convergence check results are shown. No particular issue was detected.

#6.1_6. The model estimation time is printed. The total amount of time used for this bifactor model was about 24 seconds using the author's computer. (The use of the default option of "EM" in "`mirt()`" is not efficient at all to estimate the 4-dimensional bifactor model and would have taken much longer.)

#6.1_7. The point estimates of the model parameters are printed in a simplified table using "`simplify=T`."

```
> coef(mod.md, simplify=T)
$'items'
        a1    a2    a3    a4      d  g  u
V1   1.944 0.664 0.000 0.000  1.336  0  1
V2   0.879 0.327 0.000 0.000  0.902  0  1
...
V6   0.762 0.554 0.000 0.000 -1.375  0  1
V7   1.210 0.039 0.000 0.000 -0.901  0  1
V8   2.177 0.000 1.061 0.000 -0.922  0  1
V9   1.651 0.000 0.640 0.000  1.513  0  1
...
V13  0.935 0.000 0.283 0.000 -1.440  0  1
V14  1.543 0.000 0.944 0.000  1.922  0  1
V15  1.111 0.000 0.000 0.687  0.106  0  1
V16  2.192 0.000 0.000 1.154  1.075  0  1
...
```

```
V20  2.263 0.000 0.000 0.473 -2.006 0  1
V21  1.319 0.000 0.000 1.562 -2.202 0  1

$means
G S1 S2 S3
0  0  0  0

$cov
   G S1 S2 S3
G  1 NA NA NA
S1 0  1 NA NA
S2 0  0  1 NA
S3 0  0  0  1
```

"a1" is the primary dimension since all items, "V1" through "V21" have an estimate for this slope. "a2," "a3," and "a4" are the three specific dimensions. Items "V1" through "V7" have an estimated "a2" slope, corresponding to their association with the first specific dimension, and "a3" and "a4" are 0. Items "V8" through "V14" have an estimated "a3" slope, corresponding to their association with the second specific dimension, and "a2" and "a4" are 0. Items "V15" through "V21" have an estimated "a3" slope, corresponding to their association with the third specific dimension, and "a2" and "a3" are 0.

The mean of the slopes in the primary dimension is 1.50. The means of the slopes for the first, the second, and the third specific dimensions are 0.51, 0.61, and 1.18, respectively. Overall the primary dimension is the strongest to affect item responses across all items. Of the three specific dimensions, the third specific dimension is the most salient.

The intercept parameter estimates are shown under "d," which indicates item easiness.

The means of the ability distribution for each dimension are fixed as zeros, and the covariance matrix of the four latent traits is the identity matrix, i.e., variances equal to 1 and all covariances equal to 0. These fixed values are the bifactor model identification constraints.

#6.1_8. The output for both point estimates and their standard errors are shown by including "printSE=T" in the "coef()" function.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.SE = T,
  QMC=T) #6.1_9
```

#6.1_9. Individual latent trait scores for the primary and the three specific dimensions are estimated via the EAP estimator. The "QMC=T" option, which represents the quasi-Monte Carlo, was used to raise the efficiency of the estimation process in this 4-dimensional bifactor model case. This option

is always recommended for higher-order structures, i.e., those with four or more dimensions. The estimated ability on the primary is printed under "F1," and the three specific dimension ability estimates are shown under "F2," "F3," and "F4," respectively. The rest of the columns correspond to the standard errors for the four ability estimates.

2PLM bifactor model application with the "mirt()" function

This bifactor model can be also estimated by the "mirt()" function with a suitable choice of a high-dimensional estimation technique such as MHRM, similar to the technique described in section 5.7. The following commands are shown for those who may be interested in running the bifactor model with the MHRM method in the "mirt()" function.

```
spec2<-'F1=1-21
        F2=1-7
        F3=8-16
        F4=17-21' #6.1_10
mod.md2<-mirt(ddd,model=spec2,itemtype="2PL", SE=T,
method='MHRM',
SE.type = 'FMHRM', technical=list(MHRM_SE_draws=5000))
  #6.1_11
mod.md2 #6.1_12
coef(mod.md2, simplify=T) #6.1_13
```

#6.1_10. The factor and item loading structure is specified. All items load onto the primary dimension "F1." Items 1 through 7, 8 through 16, and 17 through 21 load onto each of the three specific dimensions, "F1," "F2," and "F3," respectively.

The population latent trait distribution is a multivariate (4-dimensional) normal distribution. The default of this multivariate distribution is that the means are 0 and the covariance matrix is the identity matrix (i.e., the variances are 1 and covariances are 0); thus, no separate specification for these are necessary in the bifactor modeling.

#6.1_11. The bifactor model with the "spec2" model structure is fit using the "mirt()" function according to the "2PL" model. As mentioned in section 5.7, the "method='MHRM'" is recommended for the high-dimensional model estimation, but it tends to fail the second-order convergence check test. See #5.7_8 through #5.7_11. To increase the chance of passing the second-order test and approximate the information matrix better, the number of draws is increased to 5000 using "technical=list(MHRM_SE_draws=5000)" in the stochastic estimation process of the information matrix, and the specification of the argument "SE.type = 'FMHRM'" is used. See "Try2" in section 5.7.

#6.1_12. The convergence check results show that the second-order test passed.

#6.1_13. The model parameter estimates are shown. Note that the estimation methods in "`mirt()`" using MHRM and in "`bfactor()`" are different, and the parameter estimates are not the same. Both are expected to yield similar results, in general, as the sample size increases, but for a particular measurement situation, having specific numbers of items (e.g., long or short test) and test-takers (small or large), they may show some differences. The author's personal choice is to use "`bfactor()`" when fitting a bifactor model structure if there is no other particular reason not to.

Rasch bifactor model application with the "`mirt()`" function

Here is an example of estimating the bifactor Rasch modeling via "`mirt()`" with the MHRM estimation method. (Another way to estimate a Rasch bifactor model using the "`bfactor()`" function is shown in #6.3_9.)

```
spec.r<-'F1=1-21
         F2=1-7
         F3=8-16
         F4=17-21'
mod.md.rasch.mhrm<-mirt(ddd,model=spec.r,
itemtype='Rasch', SE=T, method='MHRM')
mod.md.rasch.mhrm
coef(mod.md.rasch.mhrm, simplify=T)
```

A syntax to specify the item loading is created and saved as "`spec.r`." (This item loading structure is the same as "`spec2`" in #6.1_10.) The covariances of the latent trait dimensions are set to be zeros as default. There is no need to specify the constraint of the slope equal to 1 because of the use of the argument "`itemtype='Rasch'`," which fixes all the slopes to 1s and allows the variances of the dimensions to be free parameters. In this case, even without using "`SE.type = 'FMHRM'`" or increasing the maximum number of draws for approximating the information matrix, the bifactor Rasch model calibration with the "`method='MHRM'`" passed the second-order test.

6.2 Bifactor modeling for polytomous item responses

When polytomous response data, e.g., data from Likert style items with item options having strongly disagree, disagree, agree, and strongly agree, follow a bifactor structure, the "`bfactor()`" function in the "mirt" package can be used. The use of "`bfactor()`" is an effective way to estimate a bifactor model even with a higher-dimensional modeling, which works well under the embedded default setting in the program. The IRT model fitted by the "`bfactor()`" is the bifactor GRM. The "`bifactor()`" function does not currently have the capabilities of fitting the GPCM (or PCM) directly, so the use of "`mirt()`" is illustrated with a proper choice of a high-dimensional estimation method as an alternative. However, the

choice of a high-dimensional estimation method and how to deal with a potential issue of failing the second-order test in checking convergence usually requires more knowledge, more number of choices and decisions, and attention from the user.

GRM bifactor model application with the "`bfactor()`" function

The cumulative response probability of the *i*th item for *k* or higher in the GRM version of bifactor modeling is

$$P\left(X_{ikj} \geq k | \theta_{jp}, \theta_{js}\right) = \frac{\exp\left[a_{ip}\theta_{jp} + a_{js}\theta_{js} + d_{ik}\right]}{1 + \exp\left[a_{ip}\theta_{jp} + a_{js}\theta_{js} + d_{ik}\right]}, \quad (6.2\_1)$$

where X_{ikj} is the *j*th person's response in the category *k* or higher to the *i*th item, and d_{ik} is the intercept for the *i*th item related to the category boundary parameter for category *k* or higher.

```
ddd<-read.table("c6_Bif_poly.dat") #6.2_1
dim(ddd) #6.2_2
apply(ddd,2,table) #6.2_3
```

#6.2_1. The ordered item response data file, "c6_Bif_poly.dat," which has a space-delimited format, is read by "`read.table()`" and saved as "`ddd.`" The data are assumed to follow a bifactor structure where all items load on the general or primary dimension. There are four specific dimensions, where items 1 through 6 are associated with only the first specific dimension, items 7 through 12 are associated with only the second specific dimension, items 13 through 18 are associated with only the third specific dimension, and items 19 through 24 are associated with only the fourth specific dimension. The 24 polytomous responses are from items with a four-category response option (0, 1, 2, or 3). The default variable or item names are given in "`ddd`," which are "V1," "V2," . . . , "V24."

#6.2_2. The size of the data set is printed. The number of rows, or test-takers, is 1821, and the number of columns, or items, is 24.

#6.2_3. Frequency analysis of item responses is printed. For all items, all four categories are utilized. If the frequency check for a particular item, e.g., item 1, is desired, "`table(ddd$V1)`" can be used, which shows the first item "V1" category frequency.

```
spec<-c(rep(1,6),rep(2,6),rep(3,6),rep(4,6)) #6.2_4
mod.md<-bfactor(ddd, model=spec, SE=T) #6.2_5
mod.md #6.2_6
extract.mirt(mod.md, what="time") #6.2_7
coef(mod.md, simplify=T) #6.2_8
coef(mod.md, printSE=T) #6.2_9
```

#6.2_4. The item loading structure for the four specific dimensions in a 5-dimensional bifactor model (one primary and four specific dimensions) is specified. The first six items (items 1 through 6) load on the first specific dimension. The second set of six items (items 7–12) loads on the second specific dimension. The third set of six items (items 13–18) loads on the third specific dimension. The last set of six items (items 19–24) loads on the fourth specific dimension. (Note, a bifactor structure does not require that the number of items associated with each specific dimension be equal.) The specification of the item loadings does not need to be ordered as in this example. See #6.1_3. There is no need to specify that all items load on the primary dimension when using the "`bfactor()`" function.

#6.2_5. The 5-dimensional bifactor GRM is estimated and the results are saved under "`mod.md`." The default CRF used for polytomous data in "`bfactor()`" is the GRM, so no argument is needed to specify this.

#6.2_6. Convergence check results are shown. No particular issue was detected.

#6.2_7. The model estimation time is printed. It took about 84 seconds on the author's computer. Given the number of dimensions is five, this indicates again that the dimension reduction technique embedded in the "`bfactor()`" function works very efficiently in the bifactor model estimation. Using the "`mirt()`" function with the default MML EM estimation method is practically not possible to complete the model estimation due to the high dimensionality in this occasion.

#6.2_8. The model parameter estimates are printed.

```
> coef(mod.md, simplify=T)
$'items'
       a1    a2    a3    a4    a5     d1     d2     d3
V1  1.022 0.684 0.000 0.000 0.000  1.242  0.980  0.035
V2  1.657 0.860 0.000 0.000 0.000 -0.056 -1.004 -1.485
...
V5  2.092 0.662 0.000 0.000 0.000 -0.920 -1.567 -2.109
V6  1.493 0.264 0.000 0.000 0.000  1.170 -0.477 -0.900
V7  1.048 0.000 0.449 0.000 0.000  0.510  0.005 -0.976
V8  1.302 0.000 0.684 0.000 0.000  1.653  1.154  0.125
...
V11 1.625 0.000 0.737 0.000 0.000  0.967  0.487 -1.028
V12 0.874 0.000 1.013 0.000 0.000  1.063  0.789 -0.496
...
V19 1.481 0.000 0.000 0.000 0.574  0.438 -0.106 -1.067
V20 1.569 0.000 0.000 0.000 0.724  1.744  1.240  0.197
...
V23 1.775 0.000 0.000 0.000 0.700  0.834  0.312 -1.052
V24 1.607 0.000 0.000 0.000 0.869  0.999  0.717 -0.438
```

```
$means
G S1 S2 S3 S4
0  0  0  0  0

$cov
   G S1 S2 S3 S4
G  1 NA NA NA NA
S1 0  1 NA NA NA
S2 0  0  1 NA NA
S3 0  0  0  1 NA
S4 0  0  0  0  1
```

The item slope estimates for the primary dimension are shown under "a1." Item slope estimates for the four specific dimensions are shown under "a2," "a3," "a4," and "a5." The intercept parameter estimates $\left(\hat{d}_{ik}\right)$ for the kth category are shown under "d1," "d2," and "d3." For this four-category polytomous data, there are three intercept parameters.

The population ability means, variances, and covariances are fixed as 0, 1, and 0, respectively. Theses fixed values are for the bifactor model identification purpose.

#6.2_9. Implementing the command line provides the model parameter estimates and their standard errors together.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.SE = T,
  QMC=T) #6.2_10
```

#6.2_10. Person latent trait scores for the primary and the four specific dimensions are estimated using the EAP estimator with the quasi-Monte Carlo option for integration ("QMC=T") in this high-dimensional bi-factor model. The first five columns ("F1," "F2," . . . , "F5") are for person estimates, and the remaining of the columns are for their standard errors ("SE_F1," "SE_F2," . . . , "SE_F5").

PCM bifactor model application with the "`mirt()`" function

The CRF of the bifactor GPCM for responding in category $k \in \{0,1,2,\ldots,m_i\}$ of the ith item is:

$$P\left(X_{ikj} = k \middle| \theta_{jp}, \theta_{js}\right) = \frac{\exp\sum_{h=0}^{k}\left[a_{ip}\theta_{jp} + a_{js}\theta_{js} + d_{ih}\right]}{\sum_{c=0}^{m_i}\exp\sum_{h=0}^{c}\left[a_{ip}\theta_{jp} + a_{js}\theta_{js} + d_{ih}\right]}, \qquad (6.2\_2)$$

where X_{ikj} is the jth person's response in category k to the ith item, d_{ih} is the intercept for the ith item related to the kth category threshold parameter when $h = k$, and the other parameters are defined the same as Equation 6.1_1. The constraint of $a_{ip} = 1$ for all items and $a_{is} = 1$ on the associated specific dimension ($a_{is} = 0$ on the other specific dimensions), defines the PCM version bifactor modeling from the GPCM.

The PCM with a bifactor structure is fit using the "`mirt()`" function with the MHRM estimation. For simplicity, the illustration fits a 3-dimensional (one primary and two specific dimension) bifactor structure for 12 Likert style items. In using the "`mirt()`" function for bifactor modeling, the constraints to specify the PCM structure and the choice of an appropriate high-dimensional model estimation method (MHRM in this example) should be provided.

```
ddd2<-ddd[,1:12] #6.2_11
spec <- ' G = 1-12
          S1 = 1-6
          S2 = 7-12
          START = (1-12, a1, 1.0)
          START = (1-6, a2, 1.0)
          START = (7-12, a3, 1.0)
          FIXED = (1-12, a1)
          FIXED = (1-6, a2)
          FIXED = (7-12, a3)
          FREE = (GROUP, COV_11)
          FREE = (GROUP, COV_22)
          FREE = (GROUP, COV_33)' #6.2_12
mod.md.pcm <- mirt(ddd2, model=spec, itemtype='gpcm',
  SE=T, method='MHRM') #6.2_13
mod.md.pcm #6.2_14
coef(mod.md.pcm, simplify=T) #6.2_15
```

#6.2_11. For the purpose of an illustration here, a 12-item Likert item response data set "`ddd2`" is created by taking the first 12 columns in the data set "`ddd`" used for the GRM application.

#6.2_12. Item loading structure for the primary dimension ("`G`") and four specific dimensions ("`S1`," "`S2`") were specified as follows: all items loading onto the "`G`" dimension; items 1 through 6 loaded onto "`S1`"; and items 7 through 12 loaded onto "`S2`." The commands of "`START`" and "`FIXED`" were used to impose the fixed value of the item slopes according to the PCM. All items, items 1 through 12, have a fixed slope on the first general dimension, "`a1`," of "`1.0`." Items 1 through 6 have a fixed slope on the second dimension, "`a2`," corresponding to the first specific dimension, of "`1.0`." Items 7 through 12 have a fixed slope on the third dimension, "`a3`," corresponding to the second specific dimension, of "`1.0`."

The "`FREE`" command relaxes the default constraint imposed on the population ability distribution variances, making the variances of the ability distribution of the three dimensions free parameters. The means of the three dimensions are constrained to be 0 by default. Notice that the model specification does not include the "`COV`" definition. In a bifactor structure there is no covariance (or correlation) between the dimensions. Since these are 0 according to the model constraint by default when using "`mirt()`," no additional specification is necessary.

#6.2_13. The 3-dimensional bifactor PCM is estimated using the previously defined "`model=spec`" according to the GPCM with "`itemtype='gpcm'`." The "`method='MHRM'`" is used for a more stable estimation of the higher-dimension structure.

#6.2_14. The convergence check results are shown. No particular issues are detected in this application. The second-order test was passed as well. If the second-order test fails, using an additional argument "`SE.type = 'FMHRM'`" with a specification of a larger number of fixed draws or increasing the maximum number of draws for approximating the information matrix may be tried. See #6.1_11 and section 5.7.

#6.2_15. The item parameter and the population parameter estimates are printed. As explained in #5.4_8, "`mirt()`" uses the $a_{iq}\theta_{jq} + d^{\star}_{ik}$ parameterization when the "`itemtype='gpcm'`," where $d^{\star}_{ik} = \sum_{h=0}^{k} d_{ih}$, and d_{ik} is the regular item intercept parameter equal to $-a_{iq}b_{ik}$. Because $a_{iq} = 1$ in this PCM, $d^{\star}_{ik} = -\sum_{h=0}^{k} b_{ik}$ here. To convert the printed intercept to the regular intercept values, refer back to #5.4_10 through #5.4_14.

```
fscores(mod.md.pcm, method="EAP", full.scores=T, full.
  scores.SE = T)
```

Person ability scores are estimated using the EAP estimator. Each person has three estimates, one for the general dimension, shown under "`F1`," and one for each of the specific dimensions, shown under "`F2`" and "`F3`." Their standard errors are shown under "`SE_F1`," "`SE_F2`," and "`SE_F3`."

In general, the use of "`mirt()`" with the MHRM method may have complications due to more frequent convergence issues when using the default settings and the longer estimation time of the model parameters when compared with the "`bfactor()`." Unless there is a specific reason to use "`mirt()`" (e.g., to fit a bifactor PCM, as in this case), "`bfactor()`" is easier to use and provides faster results.

6.3 Testlet model application

A testlet is a set of items that share the same stimuli. For instance, a reading comprehension test with 35 items could have five passages, with each passage representing some unique content (e.g., article about history or sport). Testlets pose a threat against the fundamental assumption of local independence in IRT. To cope with the potential violation of the local independence, a testlet model was proposed. In this section, the application of the 2PLM testlet model for dichotomous data is presented using the "`bifactor()`" function, and the Rasch testlet model for dichotomous data is presented with the "`bifactor()`" and "`mirt()`" functions. In general, a testlet model may be viewed as a constrained version of a bifactor model (Li, Bolt, & Fu, 2006).

The 3PLM version of the testlet model (Bradlow, Wainer, & Wang, 1999; Wang, Bradlow, & Wainer, 2002) is

$$P\left(X_{ij}=1|\,\theta_j,\gamma_{jd}\right)=g_i+(1-g_i)\frac{\exp\left[a_i\left(\theta_j-b_i+\gamma_{jd}\right)\right]}{1+\exp\left[a_i\left(\theta_j-b_i+\gamma_{jd}\right)\right]}, \qquad (6.3\_1)$$

where θ_j is the jth person ability for the target dimension and γ_{jd} is the interaction between jth person and the dth testlet ($d = 1, 2, \ldots, D$). More precisely speaking, γ_{jd} is defined as $\gamma_{jd(i)}$ for the jth person and the item i nested within the dth testlet; dropping the item i notation is taken here for simplicity because it poses no problem in the interpretation of the model and the following description. In the proposed testlet model, a_i, b_i and g_i are defined as usual in the 3PLM (see Ip (2010) for the precise interpretations of the testlet model parameters). The person ability (θ_j) and interaction (γ_{jd}) parameters are treated as random effects across persons, following a normal distribution, $\theta_j \sim N(0,1)$ and $\gamma_{jd} \sim N(0,\sigma_d^2)$. The exponent part of the model may be re-expressed as $a_i\theta_j + a_i\gamma_{jd} - a_ib_i = a_i\theta_{jp} + a_i\theta_{js} + d_i$, where θ_j is equated to the latent ability on the primary dimension, θ_{jp}, γ_{jd} is equated to latent ability on the specific dimension, θ_s ($s = 1, 2, \ldots, D$), and $-a_ib_i$ is equated to the item parameter, d_i. Then the prior testlet model has the form:

$$P\left(X_{ij}=1|\,\theta_{jp},\theta_{js}\right)=g_i+\left(1-g_i\right)\frac{\exp\left(a_i\theta_{jp}+a_i\theta_{js}+d_i\right)}{1+\exp\left(a_i\theta_{jp}+a_i\theta_{js}+d_i\right)} \qquad (6.3\_2)$$

This IRF has the same form as the 3PLM bifactor model in Equation 6.1_1 and the ith item slopes for the primary dimension, θ_p, and the specific dimension, θ_s, are constrained to be the same.

In the 3PLM bifactor modeling in Equation 6.1_1, the variance of the ability distribution for a specific dimension is set to be 1, which is part of the model constraints, and the item slope parameters on the primary and specific dimensions are freely estimated. In the testlet modeling (Equation 6.3_1 or 6.3_2), the item slope parameters of the primary and the specific dimensions are constrained to be the same, but the variances of the ability distributions on the specific dimensions (or the variances of testlet effects) are freely estimated. See Li, Bolt, and Fu (2006) for more discussion of the bifactor model and the testlet model.

In the testlet modeling, the variances of the specific dimensions (σ_d^2) represents the overall magnitude of the testlet effect, which is a convenient summary of the amount of testlet effect directly estimated from data. Because the variance of the primary dimension is set as 1, $\sigma_d^2 > 1$ implies a sizable amount of testlet effect. In terms of model-data fit, because the bifactor model has more parameters to estimate by parameterizing all item slopes as free parameters for specific dimensions, it has a higher chance of providing a better model-data fit than the testlet model. In this section, an illustration of the 2PLM testlet model is made with dichotomous response data.

```
ddd<-read.table("c6_Testlet_Dicho.dat") #6.3_1
dim(ddd) #6.3_2
```

#6.3_1. The dichotomous response data file "c6_testlet_dicho.dat" is read by "`read.table()`" into R and saved as "`ddd`." The data set contains responses to 21 items, and the variable or item labels given by the "`read.table()`" are "V1," "V2," . . . , "V21."

#6.3_2. The number of rows, or test-takers, and the number of columns, or items, are shown. They are 2500 and 21, respectively. It is assumed here that there are three testlets, each of which has seven items.

2PLM testlet model application with the "`bfactor()`" function

```
spec1<-c(rep(1,7),rep(2,7),rep(3,7)) #6.3_2
spec2<-'G = 1-21
 CONSTRAIN = (1,a1,a2),(2,a1,a2),(3,a1,a2),
             (4,a1,a2),(5,a1,a2),(6,a1,a2),(7,a1,a2)
 CONSTRAIN = (8,a1,a3),(9,a1,a3),(10,a1,a3),
             (11,a1,a3),(12,a1,a3),(13,a1,a3),(14,a1,a3)
 CONSTRAIN = (15,a1,a4),(16,a1,a4),(17,a1,a4),
             (18,a1,a4),(19,a1,a4),(20,a1,a4),(21,a1,a4)
 FREE = (GROUP, COV_22),(GROUP, COV_33),(GROUP, COV_44)' #6.3_3
```

#6.3_2. The estimation of the 4-dimensional testlet model is done by imposing constraints to a bifactor model in the "`bfactor()`" function. As with the previous bifactor models, "`spec1`" specifies the item loading structure for the specific dimension of each item. The first seven items load on the first specific dimension. The next seven items, items 8 through 14, load onto the second specific dimension. The next seven items, items 15 through 21, load onto the third specific dimension. See also #6.2_4.

#6.3_3. An additional syntax is necessary for imposing the additional modifications for the testlet model. "`G = 1-21`" defines all 21 items to load on the primary dimension "G." The following "`CONSTRAIN`" statements impose the equality constraint for item slopes between the primary dimension ("`a1`") and the specific dimension ("`a2`" for the first testlet of items 1 through 7, "`a3`" for the second testlet of items 8 through 14, and "`a4`" for the third testlet of items 15 through 21). The variances of the three specific dimensions (i.e., the overall magnitudes of testlet effects estimated) are treated as free parameters by the "`FREE`" argument (notice the "`COV_11`" is not included, so the variance of the general dimension is still fixed to 1.0). As an alternative to the "FREE" argument, users may use "`COV = S1*S1, S2*S2, S3*S3`" to free the variances of the specific dimensions.

```
mod.md<-bfactor(ddd, model=spec1, model2=spec2, SE=T,
SE.type='crossprod') #6.3_4
mod.md #6.3_5
extract.mirt(mod.md, what="time") #6.3_6
coef(mod.md, simplify=T) #6.3_7
coef(mod.md, printSE=T) #6.3_8
```

#6.3_4. The 4-dimensional testlet model is estimated with the defined "spec1" and "`spec1`." The "`model=`" argument defines the item loading structure on the specific dimensions, here "`spec1`," and is always necessary when using the "`bfactor()`" function. The "`model2=`" argument is used to define additional model constraints, as were defined in "`spec2`." If no "`model2=`" is included in the "bfactor()" function, the default constraints are used.

The "`crossprod`" standard error estimation method is selected, which is a computationally very efficient alternative. The error covariance matrix produced by the default standard error estimation method "`Oakes`" led to the failure of passing the second-order test in checking the convergence. Instead of using the default, "`crossprod`" was used. The "mirt" package is equipped with several methods for the estimation of the error covariance matrix. See more information of the standard error (or error covariance matrix in general) estimation methods in Cai (2008); Chalmers (2018); Paek and Cai (2014); Tian, Cai, Thissen, and Xin (2013); and Yuan, Cheng, and Patton (2014).

#6.3_5. Convergence check results are shown. No particular issue was detected.

#6.3_6. The model calibration time is printed. This 4-dimensional testlet model estimation took about 37 seconds for the author's computer.

#6.3_7. The model parameter estimates are shown.

```
> coef(mod.md, simplify=T)
$'items'
       a1    a2    a3    a4      d g u
V1  0.983 0.983 0.000 0.000  1.087 0 1
V2  1.152 1.152 0.000 0.000  0.946 0 1
...
V6  0.927 0.927 0.000 0.000 -1.567 0 1
V7  1.074 1.074 0.000 0.000 -0.939 0 1
V8  1.470 0.000 1.470 0.000 -0.861 0 1
V9  0.957 0.000 0.957 0.000  1.473 0 1
...
V13 1.222 0.000 1.222 0.000 -1.418 0 1
V14 1.710 0.000 1.710 0.000  1.857 0 1
V15 1.026 0.000 0.000 1.026 -0.051 0 1
V16 0.904 0.000 0.000 0.904  1.019 0 1
...
V20 1.346 0.000 0.000 1.346 -1.592 0 1
V21 1.123 0.000 0.000 1.123 -1.220 0 1

$means
G S1 S2 S3
0  0  0  0
```

```
$cov
   G   S1    S2    S3
G  1   NA    NA    NA
S1 0 0.54    NA    NA
S2 0 0.00 1.445    NA
S3 0 0.00 0.000 0.475
```

The item slopes for the primary dimension is shown under "a1," and the item slopes on the three specific dimensions are shown under "a2," "a3," and "a4." As in Equation in #6.3_2, the item slopes in the testlet model are constrained to be equal on the primary and the appropriate specific dimension within each item.

The "$cov" reports the overall magnitudes of testlet effects estimated by $\hat{\sigma}_d^2$ (d = 1, 2, and 3). These are 0.540, 1.445 and 0.475. The second testlet (items 8–14) showed a sizable amount of testlet effect (i.e., $\hat{\sigma}_d^2 = 1.445 > 1$), and the largest magnitude of the three testlets.

#6.3_8. The model parameter estimates and their standard errors are printed together.

```
fscores(mod.md, method="EAP", full.scores=T, full.scores.
 SE = T, QMC=T) #6.3_9
```

6.3_9. Person latent trait scores are estimated using "fscores()." The EAP estimator was used with the option of the quasi-Monte Carlo method "QMC=T" in this 4-dimensional testlet model. For models with more than three dimensions, "mirt" recommends the use of the option "QMC=T." Four estimates for each person are provided – one for the primary dimension and one for each of the three specific dimensions. These are shown as "F1" for the primary dimension and "F2," "F3," and "F4" on the specific dimensions. The "SE_F1," "SE_F2," "SE_F3," and "SE_F4" are the estimated standard errors.

*Rasch testlet model application with the "**bfactor()**" and "**mirt()**" functions*

The Rasch testlet model (Wang & Wilson, 2005) can be estimated by imposing appropriate constraints in the use of the "bfactor()" function with dichotomous data as shown.

```
spec1<-c(rep(1,7),rep(2,7),rep(3,7))
spec2<-'G = 1-21
        START = (1-21,a1,1.0),(1-7,a2,1.0),(8-14, a3, 1.0),
                (15-21, a4, 1.0)
        FIXED = (1-21,a1),(1-7,a2),(8-14,a3),(15-21,a4)
        FREE = (GROUP, COV_11),(GROUP, COV_22),
               (GROUP, COV_33),(GROUP, COV_44)'
mod.bif.rasch<-bfactor(ddd, spec1, spec2, SE=T,
SE.type='crossprod')
```

"spec1" is defined in the same way (see #6.3_2). "spec2" fixes and specifies the item slopes as one for all free slope parameters in the model to specify the Rasch IRF. In the Rasch testlet model, the slope of all items on the first dimension ("a1") is equal to "1.0." The item slope on the dimension associated with the specific testlet of items is also set to equal "1.0," i.e., items 1 through 7 on the first specific dimension, "a2," items 8 through 14 on the second specific dimension, "a3," and items 15 through 21 on the third specific dimension "a4." The variances of the specific dimensions, which represent overall magnitudes of testlet effects, and the variance of the primary dimension are specified as a free parameters using the "FREE" argument. See #6.1_13 for an alternative estimation approach using the "mirt()" function to estimate the Rasch bifactor model. See #6.3_4 for more details of the "bfactor()" function.

In the bifactor modeling, all dimensions are assumed to be unrelated to each other, which is also part of the model identification. In a special situation where the Rasch testlet model is used, it is possible to relax this independence of dimensions (Paek, Yon, Wilson, & Kang, 2009). (See also Jeon and Rijmen (2013) for a more general modeling of a bifactor multiple group approach where some of the restrictions on the dimension correlations are relaxed.) A situation for the Rasch testlet model where a dimension correlation is to be estimated is when a test consists of one set of items without having any testlets and the other set of items that belong to a testlet. For example, suppose a test consists of 15 items. The first eight items are independent items that do not share any common stimuli. The remaining seven items share a common stem (or passage). Here is the syntax for estimating the extended Rasch testlet model for this particular type of data, where the two dimensions are allowed to be correlated. In the estimation of this extended Rasch testlet model, "mirt()" is used.

The data set for this illustration is "c6_ext_Rasch_Testlet.dat," which has 15 dichotomous item responses, and the number of test-takers is 1100.

```
ddd<-read.table("c6_ext_Rasch_Testlet.dat")
spec<-'G = 1-15
       S = 9-15
       START = (1-15,a1,1.0),(1-8,a2,0.0),(9-15,a2,1.0)
       FIXED = (1-15,a1), (1-8,a2), (9-15,a2)
       FREE = (GROUP, COV_11),(GROUP, COV_22),(GROUP, COV_21)'
mod.bif.er<-mirt(ddd, spec, itemtype='2PL', SE=T)
mod.bif.er
coef(mod.bif.er, simplify=T)
```

The "spec" specifies the dimension structure and imposes the constraint that the item slope values are one. All 15 items load on the general dimension, "G." "S" is the specific dimension for the testlet that has seven items (items 9 through 15). To fix the item slopes as one, the "START" and "FIXED" commands were used. The item slope of all items on the first general dimension, "a1," is fixed to "1.0." The first eight items are not associated with a specific dimension, so "a2" is fixed to "0.0."

The last seven items, items 9 through 15, are associated with the second dimension, and the item slope of these items on the second dimension, "a2," is fixed to"1.0." Using the "FREE" command, the covariance between the primary and the specific dimensions is set as a free parameter ("COV_21"), and the variances of the two dimensions ("COV_11" and "COV_22") are also freely estimated. The last "coef()" command shows the estimation results.

```
> coef(mod.bif.er, simplify=T)
$'items'
   a1 a2      d g u
V1  1  0  1.266 0 1
V2  1  0 -1.064 0 1
....
V7  1  0 -0.747 0 1
V8  1  0  1.594 0 1
V9  1  1  0.735 0 1
V10 1  1  0.010 0 1
...
V15 1  1  0.899 0 1

$means
G S
0 0

$cov
      G     S
G 1.063    NA
S 0.543 0.846
```

The estimated variances of the general dimension and the specific dimension are 1.063 and 0.846, respectively. The specific dimension variance is almost 80% of that of the primary dimension (0.846/1.063 = 0.796), which indicates more than a moderate size of the testlet impact. In addition, the correlation of the two dimensions is $0.57 (= 0.543 / \sqrt{1.063 \times 0.846})$.

The regular Rasch testlet model, whose syntax is shown here, can be fit with the "bfactor()" function; however, the two dimensions are constrained to be uncorrelated.

```
spec1<-c (rep(NA,8), rep(1,7))
spec2<-'G = 1-15
        START = (1-15,a1,1.0),(9-15,a2,1.0)
        FIXED = (1-15,a1),(9-15,a2)
        FREE = (GROUP, COV_11),(GROUP, COV_22)'
mod.bif.rt<-bfactor(ddd, model=spec1, model2=spec2, SE=T)
mod.bif.rt
coef(mod.bif.rt, simplify=T)
```

"`spec1`" is used to define the specific dimension item loading. Because the first eight items do not load on a specific dimension, "`NA`" is used. "`spec2`" is similar to the previously defined "`spec`," excluding the "`S`" dimension items, and excluding the freely estimated covariance between dimensions ("`COV_21`"). The bifactor model is fit using the "`bfactor()`" function and the model specifications "`spec1`" and "`spec2`" (see also #6.3_4). The estimated coefficients are printed with "`coef()`."

```
> coef(mod.bif.rt, simplify=T)
$'items'
   a1 a2      d g u
V1  1  0  1.328 0 1
V2  1  0 -1.115 0 1
...
V7  1  0 -0.783 0 1
V8  1  0  1.670 0 1
V9  1  1  0.705 0 1
...
V15 1  1  0.862 0 1

$means
G S1
0  0

$cov
      G    S1
G  1.41    NA
S1 0.00 0.946
```

The estimated variance of the general dimension is 1.410, while that of the specific dimension is 0.946. When the "`mirt()`" function was used, where the covariance between the two dimensions was freely estimated, these were 1.063 and 0.846, respectively. When the dimension correlation (or covariance) is not modeled, the variance estimates of both dimensions became larger. The relative size of the specific dimension variance is about 67% of the general dimension (0.946/1.41 = 0.67).

6.4 Two-tier IRT modeling

The two-tier model (Cai, 2010) may be viewed as an extension of bifactor modeling. A bifactor model has a single general or primary dimension and multiple specific dimensions. A two-tier model may have more than one general or primary dimension in addition to the multiple specific dimensions. The multiple general or primary dimensions are permitted to be correlated. The general or primary dimensions are considered one tier, and the specific dimensions comprise a second tier. See Cai (2010), Paek, Park, Cai, and Chi (2014), and Paek, Li, and Park (2016) for the use of two-tier modeling with large-scale educational assessment data, with longitudinal data, and for scale development.

The demonstrations of the two-tier modeling in this section use the 2PLM IRF form for dichotomously scored data and the GRM CRF form for polytomous response data using the "`bfactor()`" function in "mirt." As mentioned earlier, the use of "`bfactor()`" has the advantage of using the dimension reduction technique, which makes the estimation of the model very efficient. The current "`bfactor()`" function can estimate the parameters of the 2PLM IRF and GRM CRF for the two-tier modeling. If a different IRF or CRF is desired, e.g., the 3PLM IRF where the lower asymptote parameters are estimated, the "`mirt()`" function can be used for the two-tier modeling, where a high-dimensional model estimation method, such as the MHRM, can be used (see section 5.7). Although the dimension reduction technique helps the efficiency of estimating the two-tier model, as a rule of thumb, if the number of primary dimensions is greater than three, the "`bfactor()`" also becomes inefficient. In such a case, "`mirt()`" with MHRM would be a more efficient choice.

Two-tier model application with the 2PLM IRF using the "`bfactor()`" function

The two-tier model with the 2PLM IRF for the ith item has the following form, which is similar to the bifactor model IRF shown in Equation 6.2_1.

$$P\left(X_{ij}=1|\,\theta_{j(p=1)},\theta_{j(p=2)},\cdots,\theta_{j(p=P)},\theta_{js}\right) = \frac{\exp\left[\sum_p a_{ip}\theta_{jp}+a_{is}\theta_{js}+d_i\right]}{1+\exp\left[\sum_p a_{ip}\theta_{jp}+a_{is}\theta_{js}+d_i\right]}, \quad (6.4\_1)$$

where $\sum_p a_{ip}\theta_{jp} = a_{i1}\theta_{j1} + a_{i2}\theta_{j2} + \ldots + a_{ip}\theta_{jp} + \ldots + a_{iP}\theta_{jP}$ ($p = 1, 2, \ldots, P$, which is the maximum number of primary dimensions). Item parameters of the general or primary dimensions are the item slope on the pth primary dimension, a_{ip}, and the jth person's ability on the pth primary dimension is θ_{jp}. Item parameters of the specific dimensions are the item slope on the sth specific dimension, a_{is}, and the jth person's ability on the sth specific dimension is θ_{js}. d_i is the ith item intercept.

```
ddd<-read.table("c6_two_tier_dich.dat") #6.4_1
dim(ddd) #6.4_2
```

#6.4_1. The data file "c6_two_tier_dich.dat" is read into R using "`read.table()`." The dichotomous response data file contains responses to 22 items. It is assumed that there are two general or primary dimensions of interest that are correlated. The first primary dimension is measured by items 1 through 13, while the second primary dimension is measured by items 11 through 22. Note that items 11, 12, and 13 measure both primary dimensions.

The test has four testlets, or subsets of related items. Testlet 1 has items 1 through 6. Testlet 2 has items 7 through 12. Testlet 3 has items 13 through 18. The last testlet (4) has items 19 through 22. The default variable (item) labels are "V1," "V2," . . . , "V22."

#6.4_2. This command shows the size of the data set. The number of rows, or test-takers, is 1511, and the number of columns, or items, is 22.

```
spec1<-c(rep(1,6), rep(2,6), rep(3,6), rep(4,4)) #6.4_3
spec2<-'G1 = 1-13
        G2 = 11-22
        FREE = (GROUP, COV_21)' #6.4_4
mod.md<-bfactor(ddd, model=spec1, model2=spec2, SE=T)
#6.4_5
mod.md #6.4_6
coef(mod.md, simplify=T) #6.4_7
coef(mod.md, printSE=T) #6.4_8
```

6.4_3. The structure of item loading for the specific dimensions, or testlets, are specified. There are four testlets each containing six items. Items 1 through 6 are associated with the first testlet, items 7 through 12 are associated with the second testlet, items 13 through 18 are associated with the third testlet, and items 19 through 22 are associated with the fourth testlet. See also #6.1_3 for a different way to specify the items for specific dimensions.

6.4_4. The item loading structure for the two primary dimensions ("`G1`" and "`G2`") is defined. There is no need to specify the item loading structure of the specific dimensions in "`spec2`" when using the "`bfactor()`" function. The covariance parameter ("`COV_21`") for the two primary dimensions is released as a free parameter using the "`FREE`" definition.

#6.4_5. The two-tier model estimation is requested using the defined "`spec1`" and "`spec2`" model structures (see also #6.3_4). This is a 6-dimesional model (two primary and four specific), which is practically very challenging or nearly impossible to run with the regular MML EM estimation method (with a reasonable number of fixed quadratures with the "`mirt()`" function). However, the two-tier model with the "`bfactor()`" function is estimated with great efficiency due to the dimension reduction technique that reduces the number of integrals to be evaluated to three (two for the primary dimensions and one of the four specific dimensions) instead of six. With the author's computer, it took about 123 seconds to finish the estimation.

#6.4_6. Convergence check results are shown. No particular issue was detected. In case the second-order test fails and the problem is more for the information matrix calculation, the "`SE.type = 'crossprod'`" may be utilized.

#6.4_7. The model parameter estimates are printed. The item slopes for the two primary dimensions are shown under "`a1`" and "`a2`." The item slopes for the four specific dimensions are shown in "`a3`," "`a4`," "`a5`," and "`a6`." "`d`" shows the intercept parameter estimate $\left(\hat{d}_i\right)$. Items 11, 12, and 13 are cross-loading items that have slope estimates for both of the primary dimensions.

The estimated covariance between two primary dimensions, which is equivalent to the correlation due to the fixed unit variance for every dimension, is 0.606. The means of the ability distributions on all dimensions was fixed to 0, and the variances were fixed to 1 for model identification purposes. Additionally, the specific dimensions were uncorrelated.

```
> coef(mod.md, simplify=T)
$'items'
       a1    a2    a3    a4    a5    a6      d g u
V1  1.157 0.000 0.762 0.000 0.000 0.000  1.258 0 1
V2  1.818 0.000 0.979 0.000 0.000 0.000  0.919 0 1
...
V6  1.524 0.000 0.156 0.000 0.000 0.000 -1.687 0 1
V7  0.903 0.000 0.000 0.852 0.000 0.000  1.431 0 1
V8  1.183 0.000 0.000 0.423 0.000 0.000  0.619 0 1
...
V11 1.609 1.583 0.000 0.823 0.000 0.000 -1.623 0 1
V12 0.910 1.061 0.000 1.043 0.000 0.000 -2.068 0 1
V13 1.572 1.952 0.000 0.000 0.403 0.000 -1.405 0 1
V14 0.000 1.776 0.000 0.000 1.552 0.000 -1.711 0 1
...
V18 0.000 1.056 0.000 0.000 0.627 0.000 -0.757 0 1
V19 0.000 1.401 0.000 0.000 0.000 0.523  0.431 0 1
...
V22 0.000 0.914 0.000 0.000 0.000 0.891  1.088 0 1

$means
G1 G2 S1 S2 S3 S4
0   0  0  0  0  0

$cov
      G1 G2 S1 S2 S3 S4
G1 1.000 NA NA NA NA NA
G2 0.606  1 NA NA NA NA
S1 0.000  0  1 NA NA NA
S2 0.000  0  0  1 NA NA
S3 0.000  0  0  0  1 NA
S4 0.000  0  0  0  0  1
```

#6.4_8. Both model parameter point estimates and their standard errors are printed.

```
pscore<-fscores(mod.md, method="EAP", full.scores=T,
full.scores.SE = T, QMC=T) #6.4_9
head(pscore) #6.4_10
```

#6.4_9. Person latent trait scores for the two primary dimensions and the four specific dimensions are estimated using the EAP estimator. The result is saved in the "`pscore`" object. The "`QMC=T`" option was used for the efficiency of the computation in this 6-dimensional bifactor model case.

#6.4_10. The "`head()`" command shows the estimated ability parameters for the first six rows, or test-takers. To observe the last six rows, or test-takers', estimates, use "`tail(pscore)`." The first six columns are the person estimates and the last six columns are their standard errors.

Two-tier model application with the GRM CRF using the "`bfactor()`" function

The two-tier version of the GRM has the following CRF, which is similar to the CRF shown in Equation 6.2_1. The cumulative response probability of the *i*th item for category *k* or higher in the two-tier GRM is

$$P\left(X_{ikj} \geq k \middle| \theta_{j(p=1)}, \theta_{j(p=2)}, \cdots, \theta_{j(p=P)}, \theta_{js}\right) = \frac{\exp\left[\sum_p a_{ip}\theta_{jp} + a_{js}\theta_{js} + d_{ik}\right]}{1 + \exp\left[\sum_p a_{ip}\theta_{jp} + a_{js}\theta_{js} + d_{ik}\right]}, \quad (6.4\_2)$$

All the terms in the previous equation are defined as before in Equation 6.2_1.

Suppose a psychological test consists of 14 Likert style items (0 = *Strongly Disagree*, 1 = *Disagree*, 2 = *Agree*, and 3 = *Strongly Agree*). The items are assumed to measure two dimensions (e.g., being extroverted and assertiveness). Items 1 through 9 and item 11 are for the first dimension, and the items 9 through 14 are for the second dimension. In addition, the test items are from the three testlets: items 1 through 4 share the first common stem; items 5 through 9 share the second common stem; and items 10 through 14 are associated with the last common stem. The two-tier model where the two dimensions are treated as primary and the three testlets have their own specific dimensions would be a researcher's choice in this occasion.

```
ddd<-read.table("c6_two_tier_poly.dat") #6.4_11
dim(ddd) #6.4_12
```

#6.4_11. The item response data file "c6_two_tier_poly.dat" is read into R and saved as "`ddd`." This data set is assumed to follow the dimensional structure previously defined. The data contains item responses to the 14 items, each of which have a four-category response scale (0, 1, 2, 3).

#6.4_12. The size of the data set is printed. The number of rows, or test-takers, is 2205. The number of columns, or items, is 14.

```
spec1<-c(rep(1,4), rep(2,5), rep(3,5)) #6.4_13
spec2<-'G1 = 1-9,11
        G2 = 9-14
        FREE = (GROUP, COV_21)' #6.4_14
mod.md<-bfactor(ddd, model=spec1, model2=spec2, SE=T) #6.4_15
mod.md #6.4_16
coef(mod.md, simplify=T) #6.4_17
coef(mod.md, printSE=T) #6.4_18
```

#6.4_13. Item loading structure is specified for the three testlets, or specific dimensions. The first four items are related to the first specific dimension, the next five items (items 5 through 9) are related to the second specific dimension, and the last five items (items 10 through 14) are related to the third specific dimension.

#6.4_14. The item loading structure for the primary dimensions is defined, where items 1 through 9 and item 11 are associated with the first primary dimension, "`G1`," and items 9 through 14 are associated with the second primary dimension, "`G2.`" The covariance of the two primary dimensions is specified as a free parameter.

#6.4_15. The two-tier GRM for the specified dimensional structure via "`spec1`" and "`spec2`" is estimated using the "`bfactor()`" function.

#6.4_16. Convergence check results are shown. No particular issue was present in this application. The use of "`extract.mirt(mod.md, what="time")`" shows that the run time was about 7.2 minutes with the author's computer.

With different sample sizes, test lengths, and dimensional structures, the author's experience has been that the two-tier model estimation may have convergence problems. The usual favorable condition for a smooth estimation is to have a large sample size. Increasing the number of EM cycles may also be helpful when the default 500 iterations are not enough. To boost the estimation speed when the number of primary dimensions is relatively high (e.g., three or four), reducing the number of quadratures could help. However, the reduction of the number of quadrature could decrease the accuracy of the model estimation. The trade-off between the speed of model estimation and the accuracy should be carefully evaluated when a smaller number of quadratures is used. (Also, practically speaking, the reduction of the number of quadratures could lead to other estimation issues such as a non-positive definite matrix for the covariance matrix of the latent trait.) When the number of primary dimensions becomes high (e.g., about four or higher), "`mirt()`," with a high-dimensional model estimation method such as MHRM, would be a better choice than "`bfactor()`," as mentioned earlier. For readers' information, the example command that controls for the EM cycles and the number of quadratures is shown here.

```
mod.md<-bfactor(ddd, model=spec1, model2=spec2, SE=T,
quadpts = 15, technical=list(NCYCLES=3000))
```

In the previous command lines, the number of quadratures per dimension was reduced to 15 (31 is the default) and the number of iterations for the EM cycle was increased to 3000 (500 is the default). These modifications were made with "`quadpts=15`" and "`technical=list(NCYCLES=3000)`," respectively.

#6.4_17. The model parameter estimates are displayed.

```
> coef(mod.md, simplify=T)
$'items'
       a1    a2     a3    a4    a5     d1     d2     d3
V1  1.074 0.000  0.081 0.000 0.000  1.127 -0.410 -0.802
V2  1.411 0.000 -0.321 0.000 0.000  0.579  0.059 -0.987
V3  1.919 0.000  0.110 0.000 0.000  1.648  1.163  0.229
V4  1.113 0.000  0.848 0.000 0.000 -0.438 -0.875 -1.402
V5  1.321 0.000  0.000 0.693 0.000  0.572  0.316 -0.209
V6  2.115 0.000  0.000 0.370 0.000  1.125  0.813 -0.502
V7  0.943 0.000  0.000 0.819 0.000  0.458 -0.640 -1.161
V8  1.748 0.000  0.000 0.564 0.000 -0.916 -1.275 -1.694
V9  0.693 1.441  0.000 0.215 0.000  1.542  0.183 -0.694
V10 0.000 1.640  0.000 0.000 0.768  0.894  0.577 -0.205
V11 1.024 1.433  0.000 0.000 0.921  0.357 -0.022 -0.351
V12 0.000 1.447  0.000 0.000 1.277  0.437 -0.023 -1.028
V13 0.000 0.636  0.000 0.000 0.781  1.644  1.151  0.179
V14 0.000 1.560  0.000 0.000 1.038 -0.398 -0.821 -1.207

$means
G1 G2 S1 S2 S3
0   0  0  0  0

$cov
      G1 G2 S1 S2 S3
G1 1.000 NA NA NA NA
G2 0.608  1 NA NA NA
S1 0.000  0  1 NA NA
S2 0.000  0  0  1 NA
S3 0.000  0  0  0  1
```

The item slope parameter estimates for the two primary dimensions are shown in "`a1`" and "`a2`." The slope estimates for the three specific dimensions are shown in "`a3`," "`a4`," and "`a5`." All items but two (items 9 and 11) load onto one primary and one specific dimension. Items 9 and 11 are cross-loading items that have a non-zero slope on both primary dimensions. The estimated correlation between the two primary dimensions is 0.608.

#6.4_18. The point estimates of the model parameters and their standard errors are shown together.

```
pscore<-fscores(mod.md, method="EAP", full.scores=T, full.
scores.SE = T, QMC=T) #6.4_19
head(pscore) #6.4_20
```

#6.4_19. The person latent trait scores using the EAP estimator are computed. Due to the high-dimensional structure (5-dimensional modeling in this application), the quasi-Monte Carlo option "`QMC=T`" was used in this application. See also #5.7_14 and #6.1_9 for more details on this option. The estimation results are stored in the R object "`pscore.`"

#6.4_20. "`head()`" extracts the first six rows of the person latent trait scores object, "`pscore`," and their standard errors. In the output, the first five columns are the person estimates for the two primary dimensions ("`F1`" and "`F2`") and the three specific dimensions ("`F3`," "`F4`," and "F5"). The remaining columns ("`SE_F1`," "`SE_F2`," . . . , "`SE_F5`") hold the corresponding standard errors of the person estimates.

References

Bradlow, E. T., Wainer, H., & Wang, X. (1999). A Bayesian random effects model for testlets. *Psychometrika*, *64*, 153–168.

Cai, L. (2008). SEM of another flavour: Two new applications of the supplemented EM algorithm. *British Journal of Mathematical and Statistical Psychology*, *61*, 309–329.

Cai, L. (2010). A two-tier full-information item factor analysis model with applications. *Psychometrika*, *75*, 581–612.

Cai, L. (2013). *flexMIRT version 3.51: Flexible multilevel multidimensional item analysis and test scoring* (Computer software). Chapel Hill, NC: Vector Psychometric Group.

Chalmers, R. P. (2018). Numerical approximation of the observed information matrix with Oakes' identity. *British Journal of Mathematical and Statistical Psychology*, *71*, 415–436.

Gibbons, R. D., Darrell, R. B., Hedeker, D., Weiss, D. J., Segawa, E., Bhaumik, D. K., . . . Stover, A. (2007). Full-Information item bifactor analysis of graded response data. *Applied Psychological Measurement*, *31*, 4–19.

Gibbons, R. D., & Hedeker, D. R. (1992). Full-information item bi-factor analysis. *Psychometrika*, *57*, 423–436.

Ip, E. H. (2010). Interpretation of the three-parameter testlet response model and information function. *Applied Psychological Measurement*, *34*, 467–482.

Jeon, M., & Rijmen, F. (2013). Modeling differential item functioning using a generalization of the multiple-group bifactor model. *Journal of Educational and Behavioral Statistics*, *38*, 32–60.

Li, Y., Bolt, D. M., & Fu, J. (2006). A comparison of alternative models for testlets. *Applied Psychological Measurement*, *30*, 3–21.

Lin, Z., & Paek, I. (2016). *A comparison of recently implemented point and standard error estimation procedures for multidimensional IRT modeling between R package 'mirt' and flexMIRT*. Presented at the 2016 Modern Modeling Methods Conference, Storrs, CT.

Paek, I., & Cai, L. (2014). A comparison of item parameter standard error estimation procedures for unidimensional and multidimensional item response theory modeling. *Educational and Psychological Measurement*, *74*, 58–76.

Paek, I., Li, Z., & Park, H-J. (2016). Specifying ability growth models using a multidimensional item response model for repeated measures categorical ordinal item response data. *Multivariate Behavioral Research*, *51*, 569–581.

Paek, I., Park, H., Cai, L., & Chi, E. (2014). A comparison of three IRT approaches to examinee ability change modeling in a single-group anchor test design. *Educational and Psychological Measurement*, *74*, 659–676.

Paek, I., Yon, H., Wilson, M., & Kang, T. (2009). Random parameters structure and the testlet model: Extension of the Rasch testlet model. *Journal of Applied Measurement*, *10*, 394–407.

Rijmen, F., Vansteelandt, K., & De Boeck, P. (2008). Latent class models for diary method data: Parameter estimation by local computations. *Psychometrika*, *73*, 167–182.

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*, 412–439.

Wang, W-C., & Wilson, M. (2005). The Rasch testlet model. *Applied Psychological Measurement*, *29*, 126–149.

Wang, X., Bradlow, E. T., & Wainer, H. (2002). A general Bayesian model for testlets: Theory and applications. *Applied Psychological Measurement*, *26*, 109–128.

Yuan, K. H., Cheng, Y., & Patton, J. (2014). Information matrices and standard errors for MLEs of item parameters in IRT. *Psychometrika*, *79*, 232–254.

7

LIMITATIONS AND CAVEAT

With the availability of the free, open-source programming language R, many IRT programs have been written as packages that work in the R environment. This book uses three IRT packages, or programs, written for the R environment: "eRm," "ltm," and "mirt." These cover most IRT applications in which applied researchers, students, instructors, and practitioners are interested. IRT models covered in this textbook have included those for dichotomous and polytomous data, and models designed for unidimensional or multidimensional data.

Nonetheless, several limitations of the current book exist. First, the choice of those three IRT packages is subjective. The current R package library lists more than 20 CRAN packages for IRT calibrations, for example, Bayesian analysis of IRT models (e.g., "bairt"; Martinez & Mosquera, 2018), supplementary item response theory models ("sirt"; Robitzsch, 2019), and test analysis modules (TAM; Robitzsch, Kiefer, & Wu, 2018), to name a few. The three presented in this textbook were chosen for their popularity, stable execution, and familiarity to the authors.

Second, all IRT programs used in this book employ the full information maximum likelihood estimation method. IRT models can be estimated using different methods, for example, the Bayesian Marko Chain Monte Carlo (MCMC) and limited information estimation approaches. The latter is typically used for factor analytic modeling or structural equation modeling (SEM). R provides packages specifically for SEM, for example, the "lavaan" package (Rosseel, 2012) and "sem" (Fox et al., 2017). The limited information estimation approach provides an alternative estimation method for IRT modeling, which is not included in this book. We plan to expand our text in the future to include the limited information estimation approach for IRT models.

Third, all IRT models covered in the book are limited to parametric IRT models, where assumptions are made on the IRF form and/or the population ability distribution. The scope of the parametric models seems to be within the popular

IRT models used frequently in practice and in research. IRT modeling can also be made non-parametrically or semi-parametrically. Non-parametric IRT modeling may be considered for exploring data, especially in the beginning of IRT analysis. For instance, IRFs can be modeled by a spline function or estimated using a Kernel smoothing method. These alterative modeling approaches (available in the "KernSmoothIRT" (Mazza, Punzo, & McGuire, 2014) and "mokken" (van der Ark, 2007, 2012) packages) and different parametric models (e.g., partially compensatory or noncompensatory logistic multidimensional models) will be considered in a future edition of the text as well.

Fourth, not all of the functionalities of each of the three R IRT programs used in this book are covered. For example, fully explanatory multiple group mixed-effects IRT modeling, latent class modeling, and the spline approach in "mirt" were not discussed. In this first edition, the essential commands for IRT analysis and their outputs were covered.

In using an R IRT program, one should exercise caution regarding the program performance when the IRT program of interest is relatively new. As a check, users could fit the same data using an R IRT program and an existing well-known software in order to check that their model estimation outputs and other related function output results are similar. Another strategy is to conduct a small-scale simulation study to check the performance of the R IRT programs. These, however, may be considered a burden to some practitioners and applied researchers. The R IRT programs used in this book are at least partly checked by the authors by conducting small-scale simulations, and we have included additional comments when necessary.

Although not all the functionalities of each command and their options have been checked, primary outputs such as point estimates of the model parameters were checked. If an R IRT program is considered for higher stakes, investigations using more systematic simulations properly designed for the specific measurement situation of interest and comparisons with existing software are recommended prior to adopting the program, though some exist in the literature.

R packages are updated by their authors periodically to fix bugs and enhance program capacities. And, the R software itself is regularly improved. Users should be aware of the continual updates and should check for major changes or updates from time to time.

References

Fox, J., Nie, Z., Byrnes, J., Culbertson, M., DebRoy, S., Friendly, M., . . . R-Core. (2017). *sem: Structural equation models*. Retrieved from https://CRAN.R-project.org/package=sem

Martinez, J., & Mosquera, I. G. (2018). *bairt: Bayesian analysis of item response theory models*. Retrieved from https://CRAN.R-project.org/package=bairt

Mazza, A., Punzo, A., & McGuire, B. (2014). KernSmoothIRT: An R package for Kernel smoothing in item response theory. *Journal of Statistical Software*, *58*, 1–34.

Robitzsch, A. (2019). sirt: Supplementary item response theory models. *R package version 3.1–80*. Retrieved from https://CRAN.R-project.org/package=sirt

Robitzsch, A., Kiefer, T., & Wu, M. (2018). TAM: Test analysis modules. *R package version 3.0–21*. Retrieved from https://CRAN.R-project.org/package=TAM

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software*, *48*, 1–36.

van der Ark, L. A. (2007). Mokken scale analysis in R. *Journal of Statistical Software*, *20*(11), 1–19. Retrieved from www.jstatsoft.org/v20/i11/

van der Ark, L. A. (2012). New developments in mokken scale analysis in R. *Journal of Statistical Software*, *48*, 1–27. Retrieved from www.jstatsoft.org/v48/i05/

INDEX

1PLM 3, 14, 49, 58, 64
2PLM 3, 14, 41, 42, 50
3PLM 3, 14, 74, 75
4PLM 3, 88

ability 1
Adams, R. J. 198, 233
AIC 14, 32
AICc 32
Akaike, H. 14, 32
Allen, N. L. 2
Andersen, E. B. 26
Andrich, D. 3, 14, 123

Baker, F. B. 15
Barton, M. A. 3, 88
between-item multidimensionality 198
BIC 14, 32
bifactor 239, 240, 245, 251, 257
Birnbaum, A. 3, 41, 75
Bock, R. D. 3, 156, 172
Bradlow, E. T. 251

C2 121
Cai, L. 4, 32, 39, 75, 120, 121, 189, 226, 229, 240, 253, 257
Camilli, G. 3
category boundary parameter 145
category characteristic curve (CCC) 102
category response function (CRF) 102
Chalmers, R. P. 6, 15, 30, 31, 32, 158, 189, 190, 230, 253
Chen, W-H. 39, 57
Cheng, Y. 88
Christensen, K. B. 26, 27
classical test theory (CTT) 1, 104
CML 6, 15
concurrent calibration 164, 172
constructed response (CR) item 164, 165, 169

de Ayala, R. J. 41, 123
De Boeck, P. 189
deviation parameter 103
DIC 78
difference model 146
differential item functioning (DIF) 172, 179
dimensionality 4
divided-by-total model 103
Drasgow, F. 39, 48, 56

EAP 35
Educational Testing Service (ETS) 2
eRm 6, 15

Fischer, G. H. 27
fixed item parameter calibration (FIPC) 184
Fox, J. 266
full information estimation 226, 233

Gelman, A. 78
general factor 241
generalized partial credit model (GPCM) 3, 102, 132, 216

Gibbons, R. D. 239
graded response model (GRM) 3, 102, 145, 221

help or ? 11
high dimensionality 226, 232, 247
Holland, P. W. 179
Houts, C. R. 229

infit/outfit 28
Ip, E. H. 251
item characteristic curve (ICC) 15, 22
item difficulty 15
item discrimination (or slope) parameter 41
item fit 15, 28, 37
item response function (IRF) 2, 4
item response theory (IRT) 1, 2

Jeon, M. 255

Kolen, M. J. 176, 184

latent trait 1, 2
LD-X2 39
Levine, M.V. 48
Li, Y. 250, 251
likelihood ratio test 49
Likert style item 104, 164, 245
limited information estimation 233, 266
limited information test 37
Lin, Z. 6, 226, 229, 240
Linacre, J. M. 4
local independence 4
Loken, E. 88
Lord, F. M. 1, 3, 85
lower asymptote 33, 75, 88
ltm 6, 42, 58, 133, 146

M2 37
M2* 120, 121
Magis, D. 85
Mair, P. 6, 15
Martin, M. O. 2
Martinez, J. 266
Masters, G. N. 3, 103
Maydeu-Olivares, A. 32, 37, 121
Mazza, A. 267
McDonald, R. 1
Metropolis-Hastings Robins-Monro (MHRM) 189, 226
"mirt" 6, 30, 50, 64, 75, 88, 115, 124, 138, 150, 157, 164, 199, 239
mixed format responses 121, 164
MML 6
multidimensionality 3, 4
multiple-choice (MC) item 2, 156
Muraki, E. 3, 116, 132

NEAT 172
nominal response model (NRM) 3, 156
non-equivalent groups anchor test 172
null category 105

one-to-one correspondence 46, 61
Organisation for Economic Co-operation and Development (OECD) 2
Orlando, M. 37, 47
overall item difficulty (location) parameter 106

Paek, I. 32, 179, 233, 253, 255, 257
partial credit model (PCM) 3, 103, 211
pattern scoring 46
person-item map 25
primary dimension 239, 241
pseudo-guessing parameter 33, 75, 88

Q3 41

Rasch, G. 3, 15
Rasch model 3, 15, 199
rating scale model (RSM) 3, 123
R Core Team 5, 15, 17
Reckase, M. D. 4, 199, 234
Rijmen, F. 240
Rizopoulos, D. 6, 42
RMSEA 37
Roberts, J. S. 4
Robitzsch, A. 266
Rosseel, Y. 266

SABIC 32
Samejima, F. 3, 145
Schilling, S. 226
Schwarz, G. 14, 32
Sclov, S. 32
simple structure 198
slip or upper asymptote parameter 88
slope and intercept form 35
Smith Jr., E.V. 28
Snijders, T. A. B. 49
specific factor or specific dimension 239
step parameter 103
sufficiency of the raw scores 15
Sugiura, N. 32

test characteristic curve (TCC) 15, 36
testlet model 240, 250
test response function (TRF) 36
test score equating 2, 164
Thissen, D. 103, 146, 157, 158
threshold parameter 103
Tian, W. 32, 253
Tollenaar, N. 64
trace line 108
two-tier IRT 240, 257

unidimensionality 14

van der Ark, L. A. 267
vertical scaling 175

Wang, X. 251, 254
Warm, T. A. 35
Wilson, M. 25, 105
within-item multidimensionality 223
Wright, B. D. 3, 15, 28, 103
"Wright Map" 25

Xu, J. 6, 37, 75, 86

Yen, W. M. 41, 47, 57, 98, 123
Yuan, K. H. 253