

Latent Variable Modeling Using R

A Step-by-Step Guide



A. Alexander Beaujean

Latent Variable Modeling Using R

This step-by-step guide is written for **R** and latent variable model (LVM) novices. Utilizing a path model approach and focusing on the *lavaan* package, this book is designed to help readers quickly understand LVMs and their analysis in **R**. The author reviews the reasoning behind the syntax selected and provides examples that demonstrate how to analyze data for a variety of LVMs. Featuring examples applicable to psychology, education, business, and other social and health sciences, minimal text is devoted to theoretical underpinnings. The material is presented without the use of matrix algebra. As a whole the book prepares readers to write about and interpret LVM results they obtain in **R**.

Each chapter features background information, boldfaced key terms defined in the glossary, detailed interpretations of **R** output, descriptions of how to write the analysis of results for publication, a summary, **R** based practice exercises (with solutions included in the back of the book), and references and related readings. Margin notes help readers better understand LVMs and write their own **R** syntax. Examples using data from published work across a variety of disciplines demonstrate how to use **R** syntax for analyzing and interpreting results. **R** functions, syntax, and the corresponding results appear in gray boxes to help readers quickly locate this material. A unique index helps readers quickly locate **R** functions, packages, and datasets. The book and accompanying website (<http://blogs.baylor.edu/rlatentvariable>) provide all of the data for the book's examples and exercises as well as **R** syntax so readers can replicate the analyses. The book reviews how to enter the data into **R**, specify the LVMs, and obtain and interpret the estimated parameter values.

Intended as a supplementary text for graduate and/or advanced undergraduate courses on latent variable modeling, factor analysis, structural equation modeling, item response theory, measurement, or multivariate statistics taught in psychology, education, human development, business, economics, and social and health sciences, this book also appeals to researchers in these fields. Prerequisites include familiarity with basic statistical concepts, but knowledge of **R** is *not* assumed.

A. Alexander Beaujean is an Associate Professor in Educational Psychology at Baylor University.

This page intentionally left blank

Latent Variable Modeling Using R

A Step-by-Step Guide

A. Alexander Beaujean

First published 2014
by Routledge
711 Third Avenue, New York, NY 10017

and by Routledge
27 Church Road, Hove, East Sussex BN3 2FA

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

© 2014 Taylor & Francis

The right of A. Alexander Beaujean to be identified as author of this work has been asserted by him in accordance with sections 77 and 78 of the Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this book may be reprinted or reproduced or utilised in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage or retrieval system, without permission in writing from the publishers.

Trademark notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging in Publication Data

A catalog record has been requested.

ISBN: 978-1-84872-698-7 (hbk)
ISBN: 978-1-84872-699-4 (pbk)
ISBN: 978-1-315-86978-0 (ebk)

Typeset in Latin Modern Roman
by A. Alexander Beaujean

Contents

Author Biography	vii
Preface	viii
1 Introduction to R	1
1.1 Background	1
1.2 Hints for Using R	18
1.3 Summary	18
1.4 Exercises	18
1.5 References & Further Readings	20
2 Path Models and Analysis	21
2.1 Background	21
2.2 Using R For Path Analysis	27
2.3 Example: Path Analysis using <code>lavaan</code>	29
2.4 Indirect Effect	30
2.5 Summary	32
2.6 Writing the Results	32
2.7 Exercises	34
2.8 References & Further Readings	36
3 Basic Latent Variable Models	37
3.1 Background	37
3.2 Latent Variable Models	38
3.3 Example: Latent Variable Model with One Latent Variable	42
3.4 Example: Structural Equation Model	50
3.5 Summary	51
3.6 Writing the Results	51
3.7 Exercises	52
3.8 References & Further Readings	55
4 Latent Variable Models with Multiple Groups	56
4.1 Background	56
4.2 Invariance	56
4.3 Group Equality Constraints	61
4.4 Example: Invariance	62
4.5 Using Labels for Parameter Constraints	70
4.6 Example: Genetically Informative Design	71
4.7 Summary	74
4.8 Writing the Results	75
4.9 Exercises	75
4.10 References & Further Readings	78
5 Models with Multiple Time Periods	79
5.1 Background	79

5.2	Example: Latent Curve Model	80
5.3	Latent Curve Model Extensions	84
5.4	Summary	88
5.5	Writing the Results	88
5.6	Exercises	89
5.7	References & Further Readings	92
6	Models with Dichotomous Indicator Variables	93
6.1	Background	93
6.2	Example: Dichotomous Indicator Variables	104
6.3	Summary	109
6.4	Writing the Results	110
6.5	Exercises	111
6.6	References & Further Readings	112
7	Models with Missing Data	114
7.1	Background	114
7.2	Analyzing Data With Missing Values	117
7.3	Example: Missing Data	121
7.4	Summary	128
7.5	Writing the Results	128
7.6	Exercises	128
7.7	References & Further Readings	130
8	Sample Size Planning	131
8.1	Background	131
8.2	Summary	142
8.3	Writing the Results	142
8.4	Exercises	143
8.5	References & Further Readings	144
9	Hierarchical Latent Variable Models	145
9.1	Background	145
9.2	Summary	151
9.3	Writing the Results	151
9.4	Exercises	151
9.5	References & Further Readings	152
	Appendix A Measures of Model Fit	153
	Appendix B Additional R Latent Variable Model Packages	167
	Appendix C Exercise Answers	171
	Glossary	190
	Author Index	195
	Subject Index	198
	R Function Index	202
	R Package Index	204
	R Dataset Index	205

Author Biography

A. Alexander Beaujean received PhDs in School Psychology and Educational Psychology from the University of Missouri. His research interests are in individual differences, especially their measurement and influence on life outcomes. He is currently an associate professor at Baylor University in the Educational Psychology Department, where he teaches courses on psychological assessment, educational and psychological measurement, and multiple regression. His scholarship has won awards from the American Academy of Health Behavior, American Psychological Association, Mensa, and the Society for Applied Multivariate Research.

Preface

The use of latent variable models has seen a tremendous amount of growth in the past 30 years across a variety of academic disciplines, including the sciences, clinical professions, business, and even the humanities. Part of the reason for this growth is the increasing availability of software to estimate these models' parameters. Traditionally, most of this software has either been too expensive or too complicated for anyone without access to the resources of a large business or university. This trend is rapidly changing, however, and there are now free programs that can conduct a latent variable analysis with only a modicum of knowledge about statistical programming.

This book is designed to introduce **R**, a free statistical program, and show how to use it for latent variable modeling. Thus, the book's two aims are to help readers:

1. understand the basics of the **R** language, and
2. use **R** to analyze a variety of useful latent variable models.

To achieve these aims, this book has some distinctive features that I highlight below.

Path Model Approach to Latent Variable Modeling

Based on teaching graduate students in education, psychology, and related disciplines, I have found that using path models tends to be an effective way to help the novice learn about latent variable models. Consequently, after introducing the **R** program in Chapter 1, I then introduce path models in Chapter 2 and continue to use these models throughout the book. While relying only on path models comes at the price of excluding their matrix representations, it comes with the benefit of increasing the readers' facility of using a model-based approach to translate their research hypotheses into data analysis—an important tool for both students and professionals.

Because of my emphasis on path models throughout the book, I mostly use the **R** package `lavaan` (and packages that work with `lavaan`) to fit the latent variable models. I purposefully did this as `lavaan` uses a path model approach to specify latent variable models. Thus, the chapter text and the **R** syntax complement each other.

Real World Perspective

Having worked with scholars from many disciplines, I know that data are not always well behaved and the syntax to analyze such data are not always easy to find. Consequently, the majority of the examples I use in this text come from published work that represent real data scholars have analyzed. This data comes from a variety of disciplines including education, medicine, psychology, and sociology.

Modern Methods

Because **R** is open-source software, it is continually being updated and improved. Thus, it can use modern techniques to analyze data. While I incorporate this modernity throughout

the book, it is particularly highlighted in the last four chapters as they contain topics that are not readily available from some other latent variable programs. For example, in Chapter 7 I discuss missing data, and demonstrate methods to determine missing data patterns as well as modern methods of handling missing data—including the use of auxiliary variables. Likewise, in Chapter 8 I demonstrate how to use Monte Carlo methods to determine the sample size needed for a prospective study.

Intended Audience

This book can be used as a supplementary text alongside a more theoretical textbook in graduate courses on latent variable modeling. In addition, this book can also be used as a supplementary text in graduate or advanced undergraduate courses that survey latent variable models or courses that review LVMs such as item response theory, measurement, or multivariate statistics taught in a variety of disciplines such as psychology, education, human development, business, economics, and other social and health sciences. Third, professionals and researchers already using latent variable models, but unfamiliar with **R**, will find this book a useful tool for learning some important features of the **R** language.

I used examples from a variety of disciplines to make the context accessible to readers from many different backgrounds, such as business, economics, education, health sciences, human development, psychology, and social science. As the only prerequisite for the text is some familiarity with statistical concepts, both **R** novices and experts should find the text accessible.

Learning Tools

There are some key features in this text to help readers use its material.

Chapter Structure

Every chapter except the first follows the same structure. They all start with some background information, then I work through one or two examples in step-by-step detail, explicitly showing **R** syntax needed for the analyses and interpreting the output. I end each chapter describing how to write the results from that chapter's content for use in a report or publication, as well as providing practice exercises and references/suggested readings. Some of the exercises follow directly from the in-text examples, while others are designed to extend the chapter's content. Most of the exercises require only the use of sample statistics to fit the latent variable model, which I provide in the book. For the exercises that require raw data, I have the files on the book's website at <http://blogs.baylor.edu/rlatentvariable>.

Glossary and Indexes

At the end of the book there are two reader-centered items. The first is a glossary of terms that are likely new and unfamiliar to the latent variable modeling novice. The second are the indices. In addition to the author and subject indices, I also placed three **R** indexes. The first one contains **R** functions, while the second and third contain **R** packages and datasets, respectively. I separated these out purposefully so that the readers do not have to scour the entire index if they forget a **R** function, package, or dataset name.

Text Formatting

This is a hint!

Term

`example.function()`

- In the margins I periodically place hints, suggestions, and information that I have found useful. These notes are designed to help readers as they write the **R** syntax for their own models as well as understand some of the complexities involved with latent variable models.
- Every time I introduce a key term, I use **boldface** and place the term in the margin. This should help readers find the areas of interest quickly when they use the book to create their own latent variable models. These terms are then defined in the end of text glossary.
- Every time I discuss a **R** function or package, I use a **truetype** font. I attach parentheses to the **R** functions [e.g., `example.function()`], and place the name in the margin anytime I introduce a new function or go into substantial detail about it. This will help readers find these functions quickly when using the book to write their own **R** syntax and analyze their own data.
- I placed all my **R** syntax in a gray box on the page, with resulting output given in the same gray box with two pound symbols **##** on the left.

R syntax

Results

Book Contents

In Chapter 1, I introduce the **R** program, and discuss how to acquire it, input/import data, and execute some simple functions. The subsequent chapters follow a sequence found in many latent variable textbooks. Chapter 2 introduces path models, while Chapter 3 extends the path models to include latent variables. In Chapter 4 I discuss how to analyze a latent variable model with data from more than one group (including twin data), while in Chapter 5 I discuss how to analyze a latent variable model with data from more than one time period.

The last four chapters are unique for an applied latent variable modeling book. In Chapter 6, I discuss how to handle dichotomous variables, using both the traditional latent variable model perspective as well as an item response theory (IRT) perspective. Further, using a worked example, I show to convert the results from one type of analysis to the other. I devote the entirety of Chapter 7 to fitting a latent variable model with missing data. I discuss types of missing data, methods to determine missing data patterns, and modern methods of handling missing data—including the use of auxiliary variables.

In Chapter 8 I demonstrate how to determine a study's sample size using Monte Carlo simulation. This is not the typical method most textbooks discuss concerning sample size planning, but I chose to focus on this method as it can be used with a wide range of statistical models as well as account for missing data. In the last chapter, Chapter 9, I focus on latent variable models with different levels (i.e., hierarchical models). I include fitting both higher-order models as well as bi-factor models.

After the last chapter, I placed three appendices. Appendix A is about measures of model fit. I do not emphasize the use of any particular model fit index in the book, but in this ap-

pendix I present a variety of common fit indices, including their formulae and interpretation. The second appendix covers a different area. Throughout this book, I mostly use the **lavaan** package. There are other **R** packages that will fit latent variable models, but it has been my experience that it is confusing to learn multiple programs concurrently, as there is a tendency to mix the syntax. Thus, in Appendix B, I provide syntax for other **R** latent variable models packages for readers wishing know how they compare to **lavaan**. Appendix C contains answers (mostly **R** syntax) for each chapter's exercises, although I do suggest trying the exercises yourself before looking at the answers!

While I included as much content as I could, due to space considerations I had to exclude two au courant areas in latent variable modeling. The first area concerns models with a categorical latent variable (i.e., latent class, latent profile). There are **R** packages available for their estimation (e.g., **poLCA**, **mclust**) and the interested reader should read their documentation for more information. The second area is Bayesian estimation. With the integration of **winBUGS** and **JAGS** with **R** (e.g., **R2WinBUGS**, **R2jags**), Bayesian estimation of latent variable model is more accessible to **R** users than ever before. Using Bayesian estimation, however, requires much more information about the process of parameter estimation than I provide in this text.

Website

There is a companion website for this book at <http://blogs.baylor.edu/rlatentvariable>. It includes raw data files, **R** syntax for the book examples in a copy-and-paste format, links to related websites with helpful information about **R** and latent variable models, as well as supplemental chapters on creating latent variable model diagrams, LISREL notation, and bootstrapping.

Acknowledgments

I am indebted to many individuals for their help with this book. In particular, I want to thank the individuals who have provided feedback on previous drafts of this text: Danielle Fearon (Baylor University), Darrell Hull (University of North Texas), Grant Morgan (Baylor University), Sonia Parker (Baylor University), Terrill Saxon (Baylor University), Yanyan Sheng (Southern Illinois University-Carbondale), Kara Styck (University of Texas-San Antonio), Phil Wood (University of Missouri), as well as all the students in my latent variable and multiple regression courses.

I also wish to thank the people at Routledge/Taylor & Francis, especially Senior Editor Debra Riegert. While I am responsible for any errors remaining in the text, the book is much better as a result of their input.

I wish to thank Yves Rosseel and Sunthud Pornprasertmanit for answering my questions about their **R** packages, and Mori Jamshidian for the advanced material concerning the **MissMech** package. In addition, thanks to the Law School Admissions Council for allowing me to use some example *Figure Classification* items in the text, and to Craig Enders for allowing the use of his *Eating Attitudes Test* data.

Finally, I owe much to my family: Christine, Susana, and Byron Limbers for their help and support while I wrote the book, Susanna and Aleisa for being my little co-authors, and William and Lela Beaujean for their support that allowed me to learn about latent variable models in the first place.

A. Alexander Beaujean
Waco, Texas

1 | Introduction to R

Chapter Contents

1.1	Background	1
1.1.1	Installing R	2
1.1.2	Starting R	2
1.1.3	Functions	2
1.1.4	Packages	4
1.1.5	Data Input	5
1.1.6	Access a Variable Within a Dataset	8
1.1.7	Example: Entering Data and Accessing Variables	9
1.1.8	Data Manipulation	10
1.1.9	Missing Data	11
1.1.10	Categorical Data	12
1.1.11	Summarize Data	12
1.1.12	Common Statistics	14
1.2	Hints for Using R	18
1.3	Summary	18
1.4	Exercises	18
1.5	References & Further Readings	20

1.1 Background

R is an open-source statistical software programming language and environment for statistical computing. It is currently maintained by the **R** Development Core Team (an international team of volunteer developers), and the **R** web page (also known as Comprehensive **R** Archive Network [CRAN]) is <http://www.r-project.org>. This is the main site for **R** information and obtaining the software.

Since **R** is syntax-based, as opposed to using a point-and-click interface, it may appear too complex for a non-specialist, but this really is not the case. Using syntax allows **R** a level of ease and flexibility not available with other programs. Take, for example, the process of analyzing a multiple regression model. While point-and-click type software can provide quick results for a single analysis, to analyze different models (e.g., using different predictor sets) or use the information from the regression for another analysis (e.g, make a scatterplot with a line of best fit, check model assumptions), it often takes many point-and-click iterations to produce the desired results. Moreover, if you have to stop your analysis and return to it days or weeks later, it can be hard to remember what you previously accomplished with the analysis or even the point-and-click sequences used to obtain the previous results. With **R**, though, many of these problems are not an issue. As **R** can store the results from the regression into *objects*, you can specify the parts of the regression results that need to be extracted for subsequent analysis. Furthermore, you can analyze multiple models and have **R** display their coefficients in a single window instead of opening many results windows, as many point-and-click programs would produce. Because these multiple models were analyzed

```
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Figure 1.1 Typical on-screen text when starting **R**.

using syntax, if you save the syntax in an external file, then you can return to the analysis months later and exactly reproduce the previous results by simply pasting the syntax back into **R**.

1.1.1 Installing R

R can be run under Windows, Mac, and Unix-type operating systems. To download **R**, go to <http://www.r-project.org/> and select the *CRAN* hyperlink. This opens a list of places (mirrors) from which to download the program. Select a hyperlink from a mirror in your country, which loads a page with hyperlinks to download **R** for your operating system (select the precompiled binary distribution).

There are some graphical user interfaces (GUIs) for **R** developed by third parties. A partial list can be found at **R** Wiki (<http://rwiki.sciviews.org/doku.php?id=guis:projects>) and CRAN (<http://www.r-project.org/GUI>). There are also many text editors that are either designed to interact with **R**, or can be modified to do so. Typing R TEXT EDITOR (or something similar) into an Internet search engine will bring up many different options as well as people's opinions about them.

1.1.2 Starting R

When initially starting **R** in interactive mode (as opposed to batch mode), the screen looks something like Figure 1.1. The `>` symbol is called the prompt. It is not typed; instead, it is used to indicate where to type. When writing syntax in **R** directly, type in all commands at the `>` prompt. If a command is too long to fit on a single line, a `+` is used for the continuation prompt.

If you type `>`, **R** interprets it as "greater than."

1.1.3 Functions

R stores variables, data, functions, results, etc, in the computer's active memory in the form of named *objects*. The user can then do actions on these objects with *operators* (arithmetic, logical, comparison) and *functions* (which are themselves objects). Much of **R**'s functionality comes from applying functions to data or other objects. **R** functions are a set of instructions that take input, compute the desired value(s), and return the result. **R** comes pre-loaded with a set of commonly used functions, but there are many additional ones to add by loading

packages with the desired functions, or by writing a function. To use functions: (a) give the function's name followed by parentheses; (b) in the parentheses, give the necessary values for the function's argument(s).

1.1.3.1 Some Useful Functions

Below are helpful **R** functions that I find myself using repeatedly.

- *Comment.* This is not really a function, but in **R** anything after the `#` sign is assumed to be a comment and **R** ignores it. Comments are extremely helpful, as annotating **R** syntax can save a lot of future time and effort. # (Comment)
- *Assign.* Another symbol that most **R** users will encounter frequently is the left arrow, `<-`, which is **R**'s standard assignment operator (another option is using `=`, but it is better to reserve using `=` for defining values for arguments). The `<-` is **R**'s way of assigning whatever is on the right of the arrow to the object on the left of the arrow. <- (Assign)
- *Concatenate.* The *concatenate* function, `c()`, concatenates the arguments included in the function. Using `c()` in conjunction with `<-` assigns the concatenated objects into a new object. For example, to make a dataset of 5 observations with the values 4, 5 3, 6, 9, and name it `newData`, I would use the following syntax: c()

```
newData <- c(4, 5, 3, 6, 9)
```

- *Help.* The `help()` function returns information about a function (or certain special words or characters). A shortcut for `help()` is a question mark, `?`. For example, the following two lines of syntax return the same results. help()
?

```
help(mean)
?mean
```

The `help()` function returns a page that (at a minimum) describes the function, its arguments, and gives some examples of how to use it. Some help pages have much more detail than others. To just execute the example syntax for a function, use the `example()` function. example()

```
example(mean)

##
## mean> x <- c(0:10, 50)
##
## mean> xm <- mean(x)
##
## mean> c(xm, mean(x, trim = 0.10))
## [1] 8.8 5.5
```

To obtain help on an entire **R** package, use the `package` argument in the `help()` function.

```
help(package = psych)
```

If you do not know exactly what you need help with in **R**, search through **R**'s documentation using the `help.search()` function. The function's argument needs to be enclosed in quotation marks. For example, if I was interested in testing to see if a variable follows a normal distribution, I could type:

`help.search()`

Topic	Package	Description
jarque.test	moments	Jarque-Bera test for normality

Figure 1.2 Example output from `help.search()` function. The results from this output indicate that in the `moments` package there is a function called `jarque.test()` that performs the Jarque-Bera test for normality.

```
help.search("normality")
```

The resulting output contains functions from packages that might be of interest, such as shown in Figure 1.2.

Another useful way to get help is to use the Rseek website (<http://www.rseek.org/>), which is a site that uses Google to help find **R** functions, lists, syntax, etc.

`help.start()` If you find yourself totally lost on where to start asking for help, then type `help.start()` into **R**. The resulting output consists of many important documents useful for navigating **R**, as well as provides another search engine (*Search Engine & Keywords*) for **R** help materials.

1.1.3.2 Writing a Function

`function()` In **R**, if a function is not available to do the desired analysis or data manipulation, there is an option to write a new function using the `function()` function. The following syntax is an example of a function I wrote to calculate the arithmetic mean, called `ArithMean()`.

```
1 # Function to calculate the arithmetic mean
2 ArithMean <- function(x) {
3   Sx <- sum(x)
4   Mean<- Sx/length(x)
5   return(Mean)
6 }
7 example.data <- c(5,10,15)
8 ArithMean(example.data)
```

First, I told **R** that I wanted to define the function named `ArithMean()`, which only takes one argument, `x` (see line 2). The left brace, `{`, indicates where the text of the function is going to start and the right brace, `}`, indicates where the text of the function is going to end. After defining the function, I evaluated one call to it (line 8). Since the sum of the numbers in the vector `example.data` is 30 and the length of the vector (i.e., the number of elements) is 3, the call to the function returned the value 10.

In the `ArithMean()` function, `x` is the *formal argument*, whereas in the call to function, `example.data`, is the *actual argument*. The formal argument is a placeholder, but `example.data` is the value used in the computation. Sometimes **R** functions have default arguments, which are values that a function’s argument(s) automatically initialize unless you specify a different value.

1.1.4 Packages

`mean()`
`var()` Using packages is a vital component to using **R**. With the initial download, **R** includes some base packages that provide the backbone functions of many statistical analysis, such as `mean()` and `var()`. These functions, however, may not do a particular analysis of interest.

Thus, I can see if a contributed **R** package has a function for the needed analysis. These **R** packages usually consist of functions and example data that were written in the **R** language (although sometimes they are written in FORTRAN or C and then linked back into **R**).

The ability for users to contribute packages is extremely powerful, as there are many experts across a variety of fields who have contributed packages that contain functions to compute almost any statistical analysis. A list of **R** packages, along with a short description of what they do, can be found in CRAN, but it is very long and hard to navigate unless looking for a specific package by name. An alternative is to examine *CRAN task views* [<http://cran.r-project.org/web/views/>], which is designed to help users find packages associated with specific types of fields or analyses. For example, the *Psychometrics* view [<http://cran.r-project.org/web/views/Psychometrics.html>] has many packages dealing with item and test analysis.

To install a package, use the (oddly enough named!) `install.packages()` function, naming the package to install in quotation marks. For example, to install the *BaylorEdPsych* package, I use the following syntax:

`install.packages()`

```
install.packages("BaylorEdPsych", dep = TRUE)
```

The `dep = TRUE` argument tells **R** that, in addition to the package of interest, I also want to download any other package upon which the package of interest is dependent. Installing the dependent packages saves the time of having to download each required package separately. Packages only need to be installed on a computer's hard disk once, but need to be loaded into **R**'s memory each time **R** is restarted and there is a need to use one of the package's functions. Load an already-installed package by using the `library()` function.

The interactive version of **R** provides an alternative method of installing packages via the *Packages & Data* menu.

`library()`

```
library(BaylorEdPsych)
```

Since **R** is case sensitive, `Install.packages("BaylorEdPsych", dep = TRUE)`, `install.packages("BaylorEdpsych", dep = TRUE)`, `install.Packages("BaylorEdPsych", dep = TRUE)`, or any other permutation will result in an error message.

1.1.5 Data Input

1.1.5.1 Concatenate

The easiest way to enter data into **R** is to directly type it using the `c()` function, and then assign it to an object. To verify that the data are in the object (in this case, a vector), just type the object's name.

```
newData <- c(4, 5, 3, 6, 9)
newData

## [1] 4 5 3 6 9
```

Once the data is in an **R** object, I can then apply functions to the object, e.g.,

```
mean(newData)
sum(newData)
```

1.1.5.2 Import Data from an External Source

Unless the dataset has one variable with a few observations (e.g., data from a textbook example), it is usually better to store the data in an external file and then import into **R**. Here are three suggestions when saving data in an external file.

- Code all missing values to **NA**, which is the default indicator in **R** for a missing value.
- Make sure the variable names do not have spaces. Use a period in lieu of a space (e.g., `first.name`).
- Save the data as a plain text file, using either tabs, spaces, or commas as delimiters. Typically when data storage programs export space- or tab-delimited files they append a `.txt` extension and use a `.csv` extension for comma-delimited files. Most spreadsheet and database programs can export data in at least one of these formats.

Variable names need to start with letters.

Besides a period, variable names should not have any other non-alphanumeric characters.

You can use other extensions for data files, such as `.dat`.

After the data are properly stored in the external file, the `read.table()` function can import the externally stored data. The main argument for this function is the external file's name and location. **R** requires the use of a forward slash or double backslash to indicate a file location. For example, say I stored a dataset in a file named `data.csv`, which I stored in a folder named *name* that is further nested in a folder named *file*. To import the data using any of the following syntaxes.

`read.table()`

```
# Windows
new.data <- read.table("C:\\file\\name\\data.csv", sep = ",")
new.data <- read.csv("C:\\file\\name\\data.csv")
new.data <- read.csv("C:/file/name/data.csv")

# Mac and Unix-type systems
new.data <- read.table("/Users/first_last/file/name/data.csv", sep = ",")
new.data <- read.csv("/Users/first_last/file/name/data.csv")
```

The `sep` argument tell the function how the variables are delimited in the data file. The default option is tab delimitation, so specifying `sep=","` is needed to indicate comma delimitation. The `read.csv()` function works just like the `read.table()` function, only it assumes the data are comma-delimited.

`read.csv()`

By default, the `read.table()` function returns a *data frame*. A data frame is a type of **R** object that stores variables as columns. Data frames are useful as they can store different types of variables, such as strings and numbers.

If the location of the data file is under many sublayers of folders, (or, more typically for me, I forget its exact location), I can search for file using the `file.choose()` function.

`file.choose()`

```
new.data <- read.table(file.choose(), header = TRUE, sep = ",")
```

This opens a dialog box that allows me to choose the file interactively.

1.1.5.3 Import Other Programs' Data

R can read in data from file formats other than plain text. Table 1.1 lists some packages and functions along with the data types they read.

Table 1.1 R Packages and Functions to Import/Read Various Data Formats.

Package/Function	File types
<code>read.fwf()</code> ^a	Fixed with format
<code>read.DIF()</code> ^a	Data Interchange Format
<code>xlsReadWrite</code>	Excel (.xls)
<code>gdata</code>	Excel (.xls)
<code>xlsx</code>	Excel (.xlsx)
<code>RODBC</code>	ODBC databases, e.g., MS SQLServer, MS Access, Oracle
<code>RMySQL</code>	MySQL
<code>RJDBC</code>	JDBC compliant databases
<code>RSQLite</code>	SQLite
<code>foreign</code>	Minitab, S3, SAS, SPSS, Stata, Systat, dBase, ARFF, DBF, REC, Octave

^a Loaded in **R** by default.

1.1.5.4 Enter a Covariance Matrix as Data

In some situations there is no access to raw data, but there is access to a covariance (or correlation) matrix. Since such matrices are symmetric, I make use of the `lavaan` package's `lower2full` function. Using the `lower2full()` function requires entering the lower diagonal of the covariance matrix *by row*. The following syntax creates a correlation matrix named *example.cor* that consists of the correlations among four variables. In addition, I name the variables using the `rownames()` and `colnames()` functions.

```
# load lavaan
library(lavaan)
# input covariances
example.cor <- lower2full(c(1, 0.85, 1, 0.84, 0.61, 1, 0.68, 0.59, 0.41, 1))
# name the rows and columns
rownames(example.cor) <- colnames(example.cor) <- c("Var1", "Var2", "Var3", "Var4")
example.cor

##      Var1 Var2 Var3 Var4
## Var1 1.00 0.85 0.84 0.68
## Var2 0.85 1.00 0.61 0.59
## Var3 0.84 0.61 1.00 0.41
## Var4 0.68 0.59 0.41 1.00
```

```
lower2full()
rownames()
colnames()
```

Published articles seldom give raw data, but often provide covariances. Thus, it is important to be able to use a covariance matrix as data for an analysis.

1.1.5.5 Package Datasets

Many **R** packages include datasets that are used to demonstrate their functions. The `data()` function produces a list of the datasets.

```
data()
```

```
# list datasets available from currently loaded packages
data()
# list datasets in all the installed packages (whether loaded or not)
data(package = .packages(all.available = TRUE))
```

Table 1.2 Common Probability Distributions Defined in **R**.

Distribution	R syntax
Binomial	<code>binom</code>
χ^2	<code>chisq</code>
F	<code>f</code>
Normal	<code>norm</code>
Poisson	<code>pois</code>
Student's t	<code>t</code>

To load a dataset from a specific package, use the dataset's name as an argument for the `data()` function.

```
library(BaylorEdPsych)
# load the MLBPitching2011 dataset from the BaylorEdPsych package
data(MLBPitching2011)
```

1.1.5.6 Simulate Data

In addition to importing data, **R** has the capability of simulating data from some well-known probability distributions. I have listed some of the common distributions in Table 1.2. To simulate data, use the `rdist()` function, where *dist* is the name of the distribution from Table 1.2. For example, the following syntax simulates 500 observations from a normal distribution with a mean of 0.0, and standard deviation of 1.0.

`rnorm()`

```
x <- rnorm(500, m=0, sd=1)
```

`seq()`

In Table 1.3, I give examples of other **R** functions that use probability distributions. The sequence function, `seq()`, can be very useful when simulating a probability density. It generates a sequence of numbers from a starting place to an ending place by a specified increment. I show an example of its use in the first row of Table 1.3. I discuss data simulation in more detail in Chapter 8 within the context of sample size planning.

1.1.6 Access a Variable Within a Dataset

Once a dataset is loaded into **R** and saved into an object, I can either do operations on the entire dataset or individual variables within the dataset. To reference a variable within a dataset, use the `$` operator.

`$`

Table 1.3 **R** Probability Distribution Functions.

Function	Description	Example
<code>ddist</code>	Density	<code>dnorm(seq(-3,3,.4))</code>
<code>pdist</code>	Cumulative probability	<code>pnorm(-1.96)</code>
<code>qdist(m)</code>	Quantile (value at the m percentile of the distribution)	<code>qnorm(0.025)</code>
<code>rdist(b)</code>	b random numbers from the distribution	<code>rnorm(100, m=0, sd=1)</code>

```
new.data$variable
```

An alternative to the `$` function is to attach a dataset using the `attach()` function. After attaching a dataset, I can directly access the variables within it.

```
attach()
```

```
attach(new.data)
variable
```

Using the `attach()` function can cause trouble, though, if the name of a dataset's variable is already being used by **R**. Specifically, if I attach two datasets to **R** and both have a variable with the same name, **R** automatically uses the variable from the most recently attached dataset. An alternative to using `attach()` is to wrap the syntax in the `with()` function. For example, say I have a dataset named *new.data*, and a variable in the dataset is named *age*. The following syntax calculates the mean of the *age* variable.

The `detach()` function detaches a dataset from **R**'s local memory.

```
with()
```

```
with(new.data, mean(age))
```

Some functions have an argument to specify the dataset to use, in which case there is no need to use the `with()` function or `$` notation. For example, the following syntax regresses the *age* variable on the *IQ* variable in the *new.data* dataset.¹

```
lm(age ~ IQ, data = new.data)
```

1.1.7 Example: Entering Data and Accessing Variables

Say I have the following data stored in a tab-delimited file named *SampleData.txt*:

```
Age IQ Height Sex
18 100 65 F
21 110 68 M
45 103 65 M
54 120 69 M
```

I use the `read.table()` function to import the dataset, and then I store the imported data in an object named *example.data*. To import the data: (a) tell **R** where the data is located (within parentheses), and (b) tell **R** the name to assign the newly-created data frame. In addition, I specify that the *SampleData.txt* file has variable names (i.e., a header) using the `header=TRUE` argument.

```
example.data <- read.table(file = "SampleData.txt", header = TRUE)
```

To show (print) the data on screen, type the name of the **R** object.

```
example.data

##   Age  IQ Height Sex
## 1  18 100      6   F
## 2  21 110     68   M
## 3  45 103     65   M
## 4  54 120     69   M
```

If the dataset is large, use the `head()` function to examine the first 6 lines.

To specify a specific variable within a dataset, use the `$` operator, the `attach()` function, or the `with()` function.

Make a habit of checking your data immediately after importing it to make sure it was read correctly.

¹The `lm()` function is the default regression function in **R**.

```
example.data$Age

## [1] 18 21 45 54

with(example.data, Age)

## [1] 18 21 45 54

attach(example.data)
Age

## [1] 18 21 45 54
```

1.1.8 Data Manipulation

Once the data is loaded in **R**, there are many ways to manipulate it, some of which I list in Table 1.4. To access individual elements, use square brackets, `[]`. On the inside of the brackets, specify the element(s) to access (i.e., *index*). For example, the following syntax accesses the third element of the variable (vector) *new.data*:

```
new.data <- c(4, 5, 3, 6, 9)
new.data[3]

## [1] 3
```

To access multiple elements within a vector (i.e., *subset* the data), specify *all* the elements of interest, e.g.,

```
# subset elements 1 through 3
new.data[1:3]

## [1] 4 5 3

# subset elements 1, 3, and 5
new.data[c(1, 3, 5)]

## [1] 4 3 9
```

Suppose I made an error typing the data in *new.data*: the 9 should be a 4. I could re-type the entire dataset again, or just tell **R** I only want to re-type the fifth value:

```
new.data[5] <- 4
new.data

## [1] 4 5 3 6 4
```

To remove a value (or multiple values), use the minus sign (`-`) along with the index number(s) of the value(s) to remove. For example, the following syntax removing the third and fourth values from *new.data*, and names the new object *newer.data*.

Table 1.4 R Functions for Data Manipulation.

Function	Description
<code>summary()</code>	Return a summary of a dataset or vector
<code>name.data[i, j]</code>	Return the value of the datum located in row i and column j
<code>name.data[i:k, j]</code>	Return the values of the data located in rows i through j in column j
<code>name.data[i:k, j:l]</code>	Return the values of the data located in rows i through j in columns j through l
<code>name.data[order(var),]</code>	Sort the data in numerical order by the variable var
<code>name.data[rev(order(var)),]</code>	Sort the data in reverse numerical order by the variable var
<code>name.data[var==m,]</code>	Select observations with values for the variable var equal to m
<code>name.data[var > m,]</code>	Select observations with values for the variable var greater than m
<code>name.data[var < m,]</code>	Select observations with values for the variable var less than m

```
newer.data <- new.data[-c(3, 4)]
new.data
```

```
## [1] 4 5 3 6 4
```

```
newer.data
```

```
## [1] 4 5 4
```

1.1.9 Missing Data

When using data from external sources, often there are missing values. **R** only recognizes missing data when they are coded as NA.

```
missing.data <- c(23, 56, 34, NA, 12, 56, 10)
missing.data
```

```
## [1] 23 56 34 NA 12 56 10
```

There are multiple ways to deal with missing data, and *Models with Missing Data* is devoted to showing robust methods to handle them. The simplest way to handle missing values is simply to delete the cases with missing values (i.e., listwise deletion). Some functions do this automatically, but others require the user to specify how to handle the missing values.

```
mean(missing.data)
```

```
## [1] NA
```



```
mean(missing.data, na.rm = TRUE)
```

```
## [1] 32
```

na.omit()

Another option is to make a new dataset that removes the missing values using the `na.omit()` function.

```
nomissing.data <- na.omit(missing.data)
mean(nomissing.data)
```

```
## [1] 32
```

1.1.10 Categorical Data

These are factors in the experimental design sense, not the latent variable sense.

Categorical variables have a countably finite number of possible values. For such variables, their values can be coded either qualitatively (e.g., “Male”, “Female”) or numerically (e.g., Male=0, Female=1). If a variable is coded qualitatively, then **R** has a special data class for them: *factors*.

```
qual.data <- c("Male", "Female", "Female", "Male", "Male")
qual.data
```

```
## [1] "Male" "Female" "Female" "Male" "Male"
```

```
factor(qual.data)
```

```
## [1] Male Female Female Male Male
## Levels: Female Male
```

factor()

If a categorical variable is coded numerically, **R** assumes it is a continuous variable unless told differently. To make **R** recognize it as a categorical variable, either make it a factor by using the `factor()` function, or qualitatively re-code it.

```
quant.data <- c(0, 1, 1, 0, 0)
# make quant.data a factor
quant.data <- factor(quant.data)
# make the quant.data qualitative by giving labels to the factors
quant.data <- factor(x = quant.data, levels = 0:1, labels = c("Male", "Female"))
quant.data
```

```
## [1] Male Female Female Male Male
## Levels: Male Female
```

1.1.11 Summarize Data

summary()
describe()

R’s `summary()` function is the default method of summarizing data. I typically use the `psych` package’s `describe()` function, though, as it provides most of the common summary statistics for data description and inspection.

```
library(BaylorEdPsych)
data(MLBPitching2011)
# summary statistics
summary(MLBPitching2011$ERAP)

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##      13      75      98      108    126     531      22

library(psych)
describe(MLBPitching2011$ERAP)

##   var   n mean sd median trimmed mad min max range skew kurtosis se
## 1  1 620 108 57   98    101  36  13 531   518  2.7    13 2.3
```

1.1.11.1 Tables

Another way to summarize data is to make a frequency table for each observed value of a variable. The `table()` function produces such frequency tables.

`table()`

```
# frequency of each value of ERAP variable
table(MLBPitching2011$ERAP)

##
## 13 17 18 29 31 32 33 34 36 37 38 40 41 42 43 44 45 46 47 48 49 50
##  1  2  1  2  1  2  1  1  3  4  4  2  2  3  3  2  3  1  3  1  1  3
## 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
##  5  1  2  4  3  2  7  3  2  5  7  5  1  2  1  6  7  7  5  2  6  7

..<Output Omitted>..

##  1  1  1
```

I can manipulate the `table()` function to give alternative output, such as grouping the values and dividing by the total n to get a relative frequency table.

`cut()`

```
# make cut points for frequency table groupings--here I used 50
boundaries <- seq(0, 550, 50)
# frequency table
table(cut(MLBPitching2011$ERAP, boundaries))

##
## (0,50] (50,100] (100,150] (150,200] (200,250] (250,300] (300,350] (350,400] (400,450]
##      46      281      212      49      18      5      1      4      2
## (450,500] (500,550]
##       1       1

# relative frequency table
table(cut(MLBPitching2011$ERAP, boundaries))/length(MLBPitching2011$ERAP)

##
## (0,50] (50,100] (100,150] (150,200] (200,250] (250,300] (300,350] (350,400] (400,450]
## 0.0717 0.4377 0.3302 0.0763 0.0280 0.0078 0.0016 0.0062 0.0031
## (450,500] (500,550]
## 0.0016 0.0016
```

The `cut()` function divides the range of a variable into intervals.

`length()`

The `length()` function returns the size of a vector (i.e., number of observations).

1.1.12 Common Statistics

1.1.12.1 Correlation and Covariance

`cor()`

Two frequently used correlations are the Pearson and Spearman versions, both of which can be calculated using the `cor()` function.

```
# Pearson correlations for losses (L) and Age
cor(MLBPitching2011$Age, MLBPitching2011$L)

## [1] 0.092

# Spearman correlation
cor(MLBPitching2011$Age, MLBPitching2011$L, method = "spearman")

## [1] 0.097
```

`cov()`

The covariance is calculated using the `cov()` function.

```
# covariance for losses (L) and Age
cov(MLBPitching2011$Age, MLBPitching2011$L)

## [1] 1.5
```

1.1.12.2 Z Score

A *Z* score is a linear transformation of a variable to make the mean 0.0 and the standard deviation 1.0.² When a variable is transformed to a *Z* score, it is called a *standardized* variable because its scale is standard deviation units. This transformation is used frequently in latent variable models, and it works no matter what probability distribution a variable follows. If the variable originally came from a normal distribution, though, the *Z*-transformed variable will follow a standard normal distribution, i.e., have a mean of 0.0 and variance of 1.0. To demonstrate this, I first simulate data from two different distributions: Normal and Poisson (see Section 1.1.5.6). The data are shown graphically in Figure 1.3.

```
# simulate data from Normal distribution with a mean of 100, and SD of 15
X.n <- rnorm(1000, mean = 100, sd = 15)
# simulate data from a Poisson distribution with a mean and variance of 2
X.p <- rpois(1000, lambda = 2)
# calculate mean and variance of Normal data
mean(X.n)

## [1] 99

var(X.n)
```

² The function for the *Z*-transformation is

$$Z_i = \frac{X_i - \bar{x}}{\sqrt{s_X^2}}$$

where X_i is a raw score, \bar{x} is the mean of the X variable, and s_X^2 is the variance of the X variable.

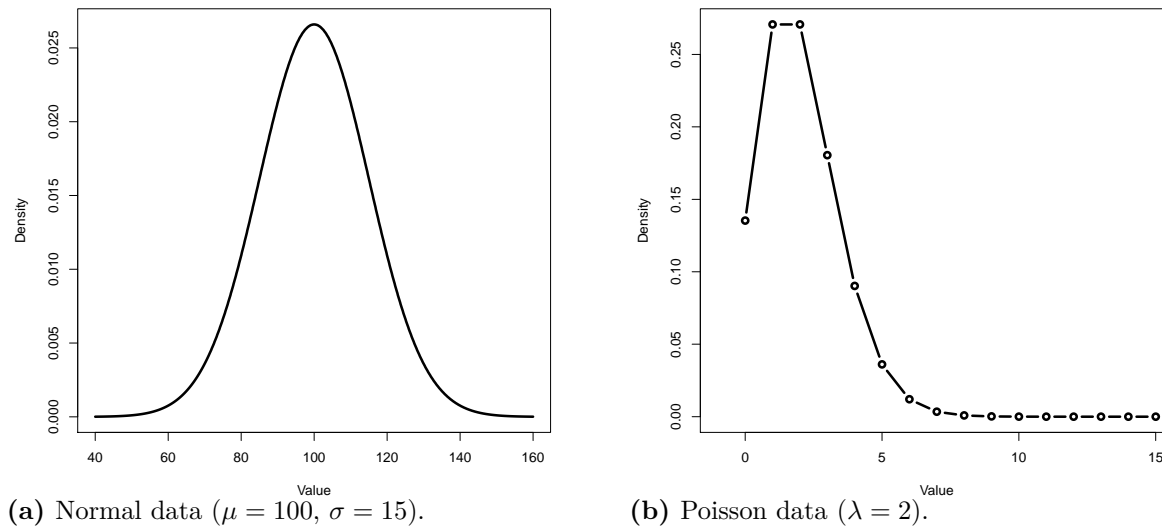


Figure 1.3 Plots of simulated data.

```
## [1] 227

# calculate mean and variance of Poisson data
mean(X.p)

## [1] 2

var(X.p)

## [1] 2.1
```

To create Z scores, **R** provides the `scale()` function.

`scale()`

```
# Z scores using the scale() function

# normal variable
Z.X.n <- scale(X.n)
# Poisson variable
Z.X.p <- scale(X.p)

# calculated mean and variance of normal Z-scores
mean(Z.X.n)

## [1] -3.4e-16

var(Z.X.n)

##      [,1]
## [1,]    1

# calculated mean and variance of Poisson Z-scores
mean(Z.X.p)
```

```
## [1] -6.1e-17
```

```
var(Z.X.p)
```

```
##      [,1]
```

```
## [1,]    1
```

1.1.12.3 Z Test

The Z test examines if a sample's mean is different than a specified value, assuming known population variability. No matter what distribution X follows, $Z_{Statistic}$ follows a standard normal distribution as the sample size becomes large (although sometimes the sample needs to become very large for this approximation to hold). Sometimes this Z -statistic is known as the Wald statistic, which is used to test the statistical significance of a parameter estimate.

R does not have a native Z -test function, but the `TeachingDemos` package contains the `z.test()` function. Using the `MLBPitching2011` data from the `BaylorEdPsych` package, I see if the average win-loss percentage is different than 0.50, assuming the population variance is 0.08. The `z.test()` function does not allow missing data, so use the `na.omit()` function to remove observations with a missing win-loss percentage value.

`z.test()`

`na.omit()`

```
library(TeachingDemos)
z.test(na.omit(MLBPitching2011$WLP), mu = 0.5, stdev = sqrt(0.08))

##
## One Sample z-test
##
## data:  na.omit(MLBPitching2011$WLP)
## z = -1.7, n = 531.000, Std. Dev. = 0.283, Std. Dev. of the sample mean = 0.012,
## p-value = 0.08833
## alternative hypothesis: true mean is not equal to 0.5
## 95 percent confidence interval:
##  0.46 0.50
## sample estimates:
## mean of na.omit(MLBPitching2011$WLP)
##                                0.48
```

1.1.12.4 t Tests

A one-sample t test is similar to the Z test, but uses the data to estimate the population variance. **R** contains the base function `t.test()` that conducts a one-sample t test.

`t.test()`

```
t.test(MLBPitching2011$WLP, mu = 0.5, alternative = "two.sided", conf.level = 0.95)

##
## One Sample t-test
##
## data:  MLBPitching2011$WLP
## t = -1.8, df = 530, p-value = 0.0735
## alternative hypothesis: true mean is not equal to 0.5
## 95 percent confidence interval:
```

```
## 0.46 0.50
## sample estimates:
## mean of x
## 0.48
```

To test if one group's mean is different from another group's mean, use an independent samples t test. **R**'s `t.test()` function can work with independent samples. It assumes that the variances are not equal between the groups unless specified differently using the `var.equal` argument (see Section 1.1.12.5 for tests of variance equality). For example, to test if the average win-loss percentage (WLP) is different between the two leagues (Lg), assuming the variances are the same across leagues, I would use the following syntax:

```
# Compare the mean WLP of the National league (Lg==NL) to the American league (Lg==AL)
# assuming the variances are equal
t.test(WLP ~ Lg, data = MLBPitching2011, na.rm = TRUE, var.equal = TRUE)

##
## Two Sample t-test
##
## data: WLP by Lg
## t = 0.15, df = 529, p-value = 0.8769
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.042 0.050
## sample estimates:
## mean in group AL mean in group NL
## 0.48 0.48
```

A third kind of t test is one for dependent samples, which the `t.test()` function can handle by using the `paired = TRUE` argument. For example, to test if the average number of wins (W) is different than the average number of losses (L) for the same pitchers, I would use the following syntax:

```
t.test(MLBPitching2011$W, MLBPitching2011$L, paired = TRUE)

##
## Paired t-test
##
## data: MLBPitching2011$W and MLBPitching2011$L
## t = 0.28, df = 641, p-value = 0.7807
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.21 0.28
## sample estimates:
## mean of the differences
## 0.034
```

1.1.12.5 Compare Variances

There are multiple ways to test if the variances between groups are equal (i.e., homogeneity of variances), such as the F -test, Bartlett's test, Levene's test, or the Brown-Forsyth test. The first two tests can be done in **R** using base functions, while the latter two require functions from the `car` package.

```
var.test()
bartlett.test()
leveneTest()
```

```
# F-test
var.test(WLP[Lg == "NL"], WLP[Lg == "AL"], na.rm = TRUE)
# Bartlett's Test
bartlett.test(WLP, Lg)
library(car)
# Levene's test
leveneTest(WLP, Lg, center = "mean")
# Brown-Forsyth Test
leveneTest(WLP, Lg, center = "median")
```

1.2 Hints for Using R

The following are some hints to optimize your **R** experience.

source()

There are many text editors for **R**. Most of them allow you to try the product for free, so I suggest trying many different editors to see which one best fits your needs.

1. Store **R** syntax in an external file and use that file to execute commands instead of typing them into **R** directly. **R** comes with a built in text-editor, but there are many others available, such as R Studio [<http://www.rstudio.com>] and emacs [<http://www.gnu.org/software/emacs/>, <http://emacsformacosx.com>]. **R** syntax files should be saved using the .R extension. Most **R** editors allow you to run part or all of the syntax. If you want to run all of the syntax in an .R file, use the `source()` function with the main argument being the name and location of the file.
2. Comment **R** syntax frequently (i.e., the `#` character). Syntax makes perfect sense the day you write it, but when revisiting it a year later, you will have difficulty remembering why you set certain values, the meaning of variables, or even what the variable represents!
3. Store the original data in a plain text file with commas or tabs as delimiters. Once you have verified that the variables' values are correct (i.e., no data coding errors), *do not do any further operations on this master dataset!* Instead, do all data manipulation and new variable creation within **R**. With some point-and-click statistics programs, you have to conduct all the analyses directly on the file in which the data is stored. Once you begin altering the variables (e.g., convert standard scores to percentiles), then it becomes too easy to override the original values. With **R**, data are read into the program and only the internal representations of the data are analyzed. Unless you purposefully choose to write over the original data file, **R** does not alter the original data. Thus, if you make a mistake in your variable creation or transformation, you have not harmed the original data.

1.3 Summary

In this chapter, I presented a small portion of what **R** can do. I explained how to install and load packages, enter data, and use some basic functions (including getting additional help on their use). For more information, type **R** and the statistic of interest in an Internet search engine and, most likely, many resources will appear. In addition, I have listed some other material on using **R** in Section 1.5.

1.4 Exercises

- 1.1 Simulate a dataset with two variables and a sample size of 100. Have variable one, T , follow a normal distribution with a mean of 50 and standard deviation of 10. Have

Table 1.5 Height (in Inches) and Weight (in Pounds) for 25 Individuals.

Participant	Height	Weight	Participant	Height	Weight
1	65.78	112.99	14	67.12	122.46
2	71.52	136.49	15	68.28	116.09
3	69.40	153.03	16	71.09	140.00
4	68.22	142.34	17	66.46	129.50
5	67.79	144.30	18	68.65	142.97
6	68.70	123.30	19	71.23	137.90
7	69.80	141.49	20	67.13	124.04
8	70.01	136.46	21	67.83	141.28
9	67.90	112.37	22	68.88	143.54
10	66.78	120.67	23	63.48	97.90
11	66.49	127.45	24	68.42	129.50
12	67.62	114.14	25	67.63	141.85
13	68.30	125.61			

variable two, Z , follow a normal distribution with a mean of 0 and standard deviation of 1. Combine the variables using the `cbind()` and `data.frame()` functions. Name the dataset *prac1.data*.

`cbind()`
`data.frame()`

- 1.2 Using the data generated in Exercise 1.1, calculate the Pearson correlation between T and Z .
- 1.3 Table 1.5 contains a dataset.
 - 1.3.a Copy the data into a spreadsheet (with the variable names) and save the file, using comma- or tab-delimiters, naming it *HeightWeight.dat*.
 - 1.3.b Read the data into **R** and call the new object **HW.data**.
 - 1.3.c Use the `head()` function to examine the first 6 observations.
- 1.4 This exercise requires a dataset from the **psych** package.
 - 1.4.a Load the `galton` dataset from the **psych** package. If you have not previously downloaded this package to **R**, you need to do so before loading the data.
 - 1.4.b Regress the *child* variable onto the *parent* variable using the `lm()` function.

1.5 References & Further Readings

- Beaujean, A. A. (2012). BaylorEdPsych: R package for Baylor University Educational Psychology quantitative courses (Version 0.5) [Computer software]. Waco, TX: Baylor University.
- R Development Core Team. (2013). *R: A language and environment for statistical computing* [Computer program]. Vienna, Austria: R Foundation for Statistical Computing.
- Crawley, M. J. (2013). *The R book* (2nd ed.). Hoboken, NJ: Wiley.
- Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. Thousand Oaks, CA: Sage.
- Matloff, N. (2011). *The art of R programming: A tour of statistical software design*. San Francisco, CA: No Starch Press.
- Revelle, W. (2012). *psych: Procedures for psychological, psychometric, and personality research* [Computer program]. Evanston, IL: Northwestern University.

2 | Path Models and Analysis

Chapter Contents

2.1	Background	21
2.1.1	Variables Relationships in a Path Model	21
2.1.2	Path Model Variations	23
2.1.3	Tracing Rules	23
2.1.4	Obtaining a Numerical Solution	24
2.1.5	Path Coefficients	26
2.2	Using R For Path Analysis	27
2.2.1	The <code>lavaan</code> Package	27
2.3	Example: Path Analysis using <code>lavaan</code>	29
2.4	Indirect Effect	30
2.4.1	Example: Indirect Effects	31
2.5	Summary	32
2.6	Writing the Results	32
2.7	Exercises	34
2.8	References & Further Readings	36

2.1 Background

A **path model** is a pictorial representation (i.e., diagram) of a theory of variable relationships. This would be handy in and of itself, but many computer programs (including **R**!) can directly use them to examine relationships among variables in a dataset. The figures used for path models are given in Figure 2.1, while Figure 2.2 shows an example of a path model for a multiple regression.

Path Model

In graph theory, the geometric figures in a path model are called *nodes* and the connecting arrow are called *edges*.

2.1.1 Variables Relationships in a Path Model

Variables can be categorized into one of two types: those that are without a direct cause (**exogenous**) and those with a direct cause (**endogenous**). In regression, the exogenous variables are sometimes called predictor, source, or upstream variables, and with an experimen-

Exogenous Variable

Endogenous Variable

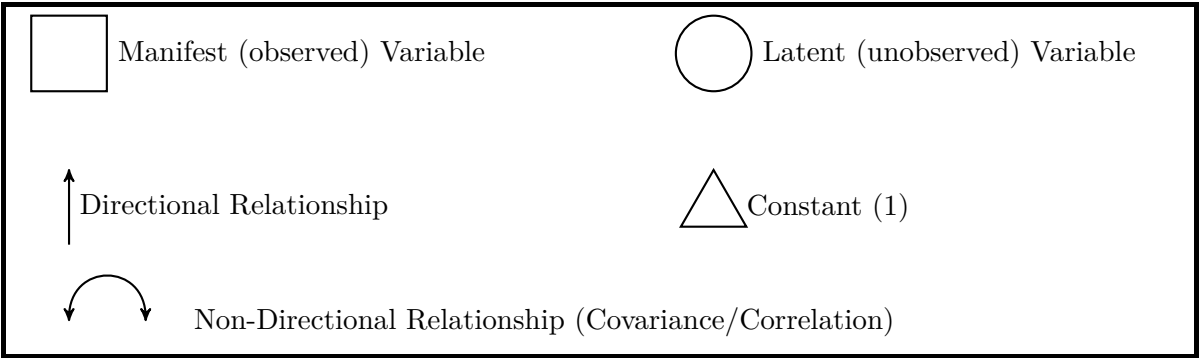


Figure 2.1 Symbols used to create path model diagrams.

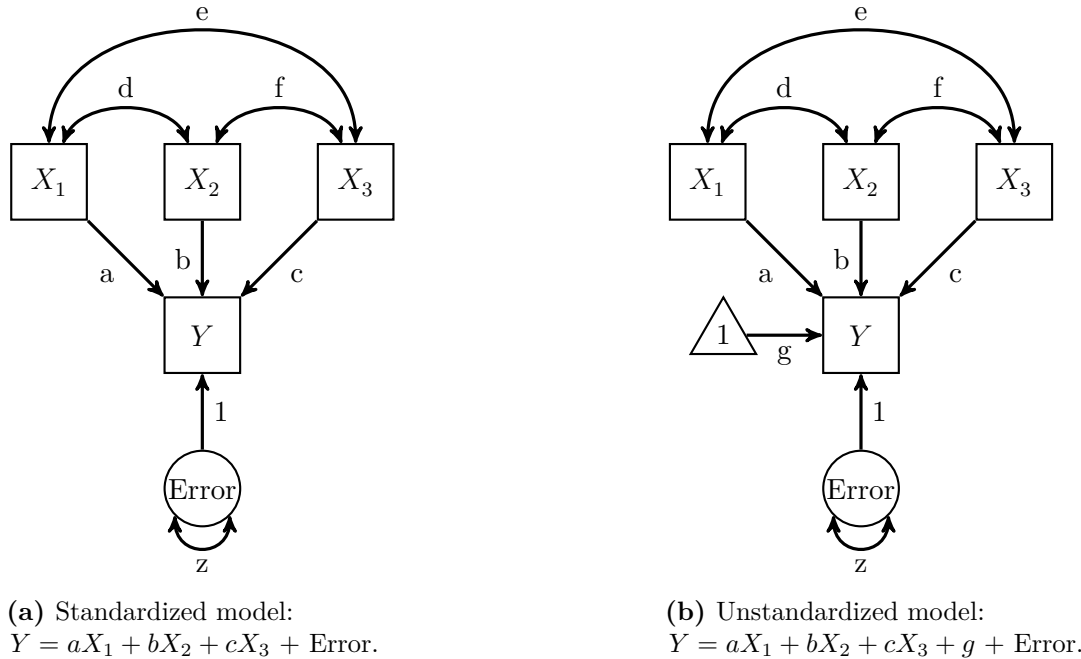


Figure 2.2 Path model of a multiple regression with three predictor (exogenous) variables.

tal design they are called independent variables. Conversely, the endogenous variables are sometimes called outcome, criterion, or downstream variables in regression, and dependent variables in an experimental design. In Figure 2.2, X_1 , X_2 , and X_3 are all exogenous, while Y is endogenous.

In path models, single-headed arrows represent direct influences, while double-headed arrows represent covariance/correlations. The former are often straight and the latter curved, but this depends on the graphics program used. The major determinant is the number of arrowheads.

Path models are assumed to be complete. That is, all important relationships are included in the model. If no relationship between two variables is shown, then no relationship is hypothesized to exist. In Figure 2.2, X_1 , X_2 , and X_3 are assumed to have a direct effect on Y as well as covary with each other.

Some texts use the language that X_1 , X_2 , and X_3 *cause* Y , but causation is more a function of the research design than the statistics.

Figure 2.2b has a shape in it that is not in Figure 2.2a: a constant, which in Figure 2.2b represents the intercept term (i.e., the value of Y when $X_1 = X_2 = X_3 = 0$). Constants are not used in basic latent variable models, but are needed for more complex models. I give more detail about them in Chapter 4.

Error

Endogenous variables always have an attached **error** term associated with them. This error term is sometimes called a *residual* or *disturbance*, but they all represent the same thing: the discrepancy between the observed values and the values predicted by the model. The variability of these discrepancies is the error variance, which is the amount of variance in an endogenous variable not explained by the other variables in the model. Because the error terms, by definition, represent something unexplained by the model, this means they have no

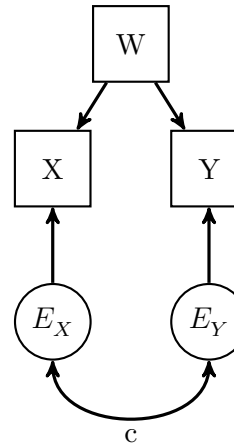


Figure 2.3 Path model of a partial correlation.

direct cause within the model and, thus, are exogenous. While endogenous variables are never connected to non-directional (double-headed) arrows, their error terms can covary with other variables. For example, Figure 2.3 shows a path model of a partial correlation. Variables X and Y are not allowed to covary since they are endogenous, but their residuals are allowed to do so. Thus, the c coefficient is the relationship between X and Y after removing the effect of W from both variables.

With regression, all the variables are **manifest variables** because they are directly observable. Path models allow the scope of the variables to be both manifest and **latent** (i.e., not directly observable), however, as long as there is enough information in the data to estimate all the parameters. In Chapter 3, I give more detail about latent variables and their use in path models.

Manifest Variable (MV)
Latent Variable (LV)

2.1.2 Path Model Variations

Different authors represent path models differently. Usually, the variation stems from whether they include and label a symbol for the error terms as well as whether they include a path for exogenous variables' variance. Figure 2.4 contains four different representations of the same path model for a simple regression. While ostensibly different, the estimated parameters are the same. In this book, I use the two path models shown on the left, making variances explicit when there is a need to do so.

If you have not previously used path models, I recommend explicitly showing all parameters in the model, such as the model on the far left in Figure 2.4.

2.1.3 Tracing Rules

The geneticist Sewall Wright first developed rules for how to estimate values for a path model's coefficients by tracing the paths within it (i.e., path analysis). These **tracing rules** (or Wright's rules) are simply a way to estimate the covariance between two variables by summing the appropriate connecting paths. For basic models (i.e., models without means), the rules are given in Figure 2.5. In Figure 4.2, I extend these rules to include a constant term.

Tracing Rules

I demonstrate the use of tracing rules by estimating the relationship between X_1 and Y , σ_{1Y} , in Figure 2.2a. To do so, I need to find all the permissible pathways from X_1 to Y . The paths are a , ec , and db (see Figure 2.6). I cannot, for instance, take the path $efda$ as it would violate the tracing rules. Thus, $\sigma_{1Y} = a + ec + db$. Similar statements can be made about σ_{2Y} and σ_{3Y} .

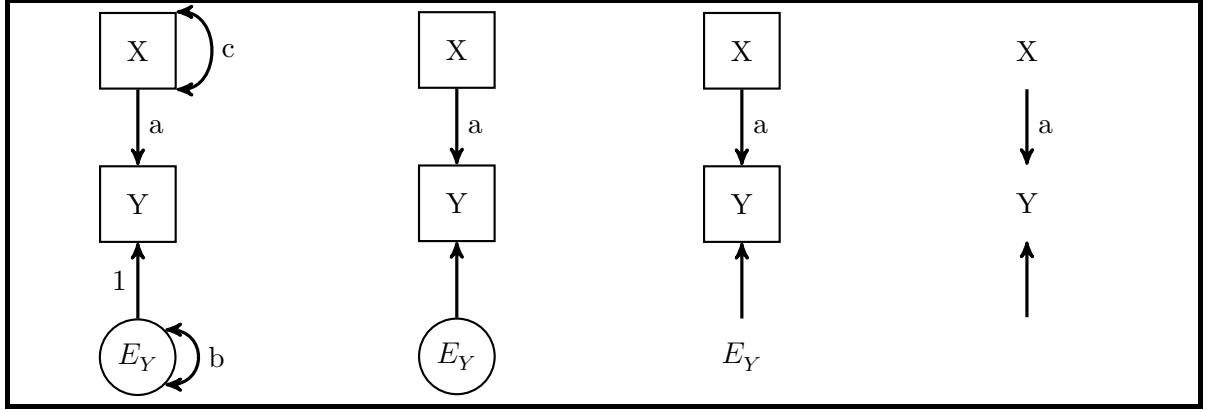


Figure 2.4 Path model variations for a simple regression.

What about the covariance of Y with itself (i.e., the variance of Y)? It consists of the paths that both start and end at Y : $aa = a^2$, b^2 , c^2 , as well as z (or, technically, $1 \times z \times 1$). The sum of a^2 , b^2 , and c^2 represent the amount of Y 's variance that X_1 , X_2 and X_3 explain (i.e., R^2), while z represents the amount left over (i.e., error variance or $1 - R^2$).

2.1.4 Obtaining a Numerical Solution

For simple path models, estimating the parameter values is a straightforward task after calculating the all the variables' covariances. I demonstrate this by estimating the parameters in Figure 2.2a using the data in Table 2.1. For convenience, I standardized the variables, which makes Table 2.1 a correlation matrix.

As d , e , and f are double-headed arrows among manifest variables, I already know their values from Table 2.1:

$$d = .20; \quad e = .24; \quad f = .30$$

- Trace all paths between two variables (or a variable back to itself), multiplying all the coefficients along a given path.
- You can start by going backwards along a single-headed arrow, but once you start going forward along these arrows you can no longer go backwards.
- No loops! That is, you cannot go through the same variable more than once for a given path.
- At *maximum*, there can be one double-headed arrow included in a path.
- After tracing all the paths for a given relationship, sum all the paths.

Figure 2.5 Tracing rules for a path model without a constant term.

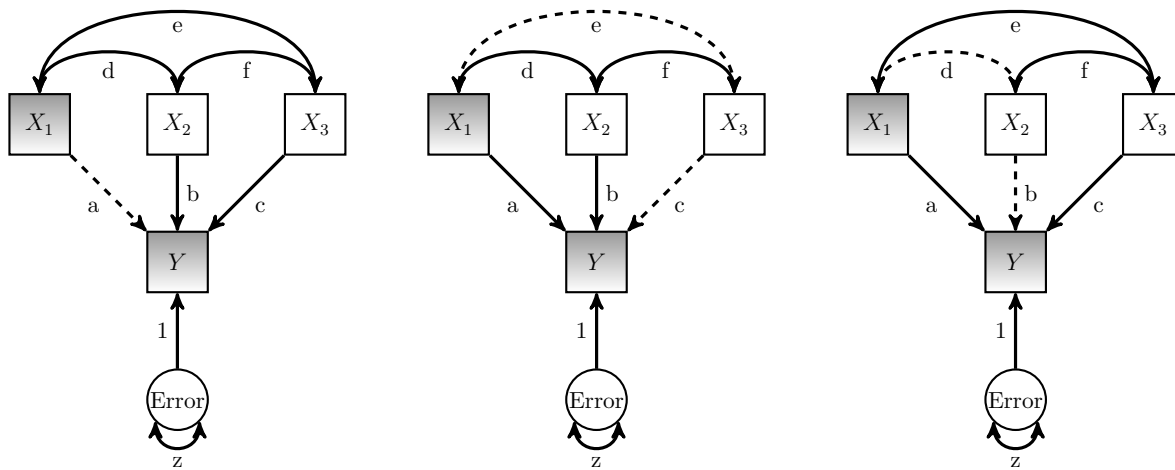


Figure 2.6 Path model from Figure 2.2a with dashed lines showing the three proper paths from X_1 to Y .

To solve for a , b , and c , it is helpful to write out the full set of paths for the correlations among endogenous variable (Y) and the three exogenous variables (X_1 , X_2 , and X_3), and then substitute the already-known values.

$$r_{1Y} = a + ec + db \rightarrow 0.70 = a + 0.24c + 0.2b$$

$$r_{2Y} = b + da + fc \rightarrow 0.80 = b + 0.2a + 0.3c$$

$$r_{3Y} = c + fb + ea \rightarrow 0.30 = c + 0.3b + 0.24a$$

Now I have three equations and three unknowns. With a little bit of algebra, I can solve for a , b , and c .

$$a = 0.57; \quad b = 0.70; \quad c = -0.05$$

To check my work, I substitute the estimated values for a , b , and c back into one of the original correlation equations (in this case, the equation for r_{1Y}) and see if it returns the correct

Table 2.1 Correlations to Accompany Figure 2.2.

	X_1	X_2	X_3	Y
X_1	1.00			
X_2	0.20	1.00		
X_3	0.24	0.30	1.00	
Y	0.70	0.80	0.30	1.00

value.

$$\begin{aligned}
 r_{1Y} &= 0.70 = a + 0.24c + 0.20b \\
 0.70 &= 0.57 + 0.24(-0.05) + 0.20(0.70) \\
 0.70 &= 0.57 - 0.01 + 0.14 \\
 0.70 &= 0.70
 \end{aligned}$$

To solve for z , I follow the same steps as solving for a , b , and c , noting that $r_{YY} = \sigma_{YY}^2 = 1$ since the variables are standardized.

$$\begin{aligned}
 r_{YY} &= a^2 + b^2 + c^2 + 2(cea) + 2(cfb) + 2(bda) + z \\
 1.00 &= 0.57^2 + 0.70^2 + (-0.05)^2 + 2(-0.05)(0.24)(0.57) + \\
 &\quad 2(-0.05)(0.30)(0.70) + 2(0.70)(0.20)(0.57) + z \\
 1.00 &= 0.335 + 0.490 + 0.002 + -0.014 + -0.021 + 0.160 + z \\
 1.00 - 0.942 &= z \\
 0.058 &= z
 \end{aligned}$$

To estimate the amount of Y 's variance the predictor variables explain (i.e., R^2), I use the same formula I used for r_{YY} , but leave out the error term (z).

$$R^2 = a^2 + b^2 + c^2 + 2(cea) + 2(cfb) + 2(bda) = 0.942$$

2.1.5 Path Coefficients

The values of $d - f$ in Figure 2.6 are correlation coefficients, but what are the values a , b , and c ? They are standardized partial regression coefficients, often referred to as just **path coefficients**. *Standardized* means are variables transformed to Z scores, which places them on standard deviation units (see Section 1.1.12.2); *partial regression* means that it is the relationship between an exogenous and endogenous variable, controlling for all the other exogenous variables going to that endogenous variable.

Path Coefficient

It is not required that the path coefficients be standardized. The variables could just as easily have been in their natural metric (i.e., raw score units), which would have made the path coefficients *unstandardized*. The situation is the same as in multiple regression when dealing with standardized (b^*) versus unstandardized (b) coefficients, and the tradeoffs mentioned there for both types of coefficients apply here as well.¹ Usually standardized values are better when comparing coefficients within the same model (or across models for the same sample), while unstandardized values are better when comparing coefficients for the same variable relationships across samples or when the raw score units are meaningful (e.g., dollars, height, age). It is relatively easy to obtain the standardized coefficients from the unstandardized ones, and vice versa, using Equation (2.1) and Equation (2.2).

¹I use asterisked coefficients to indicate standardized coefficients only when there is likely to be confusion about whether a coefficient is standardized or unstandardized.

$$b = b^* \frac{\sigma_Y}{\sigma_X} \quad (2.1)$$

$$b^* = b \frac{\sigma_X}{\sigma_Y} \quad (2.2)$$

where σ is the standard deviation, Y is the outcome variable (i.e., at the end of the single headed arrow), and X is the predictor variable (i.e., at the beginning of the single headed arrow).

2.2 Using R For Path Analysis

To conduct a path analysis in **R**, I use the **lavaan** package, as this is the same package I use for the latent variable analyses in subsequent chapters. In Appendix B, I present some alternative **R** packages that conduct path analysis.

2.2.1 The lavaan Package

lavaan (*LA*tent *VA*riable *AN*alysis) is an **R** package designed for general structural equation modeling. Information and documentation about it can be found on the package's web page: <http://www.lavaan.org>.

Installing **lavaan** from CRAN follows the same steps as installing other **R** packages (see Section 1.1.4).

```
install.packages("lavaan", dependencies = TRUE)
```

To compute a model in **lavaan** requires two steps: first specify the path model, then analyze the model. Specifying the model requires using the package's pre-defined model specification commands, most of which are shown in Table 2.2. As an example, the following syntax specifies the path model in Figure 2.7. Although not required, I label each parameter to match the labels in Figure 2.7.

```
1 example.model<- '
2 C ~ y*B + w*A
3 D ~ z*C + x*A
4 # optional label of residual variance
5 C~~C_Resid*C
6 # optional label of residual variance
7 D~~D_Resid*D
8 '
```

Line 1 consists of the name of the model (**example.model**), **R**'s assignment operator (**<-**) and a single apostrophe (**'**). The apostrophe tells **R** that the subsequent syntax needs to be stored as text. Line 2 defines the part of the path model that has C as the outcome and, likewise, line 3 defines the path model for D as the outcome. Each unique specification for the model needs to start on a new line (as I have done) or be separated by a semi-colon.

Lines 5 and 7 are optional. By default, **lavaan** creates an error term for each endogenous variable and estimates its variance while constraining the path from the error term to 1.0. Likewise, by default **lavaan** creates a covariance term for each exogenous variable (represented by the double-headed arrow from A to B in Figure 2.7). So, the only contribution of

I find it very helpful to use comments (#) when specifying models to help remember why I did what I did when I return to the model after days, weeks, or months of not working on it.

Table 2.2 lavaan Syntax for Specifying Path Models.

Syntax	Command	Example
~	Regress onto	Regress B onto A: $B \sim A$
~~	(Co)variance	Variance of A: $A \sim\sim A$ Covariance of A and B: $A \sim\sim B$
~1	Constant/mean/intercept	Regress B onto A, and include the intercept in the model: $B \sim 1 + A$ or $B \sim A$ $B \sim 1$
=~	Define reflective latent variable	Define Factor 1 by A-D: $F1 =\sim A+B+C+D$
<~	Define formative latent variable	Define Factor 1 by A-D: $F1 <\sim 1*A+B+C+D$
:=	Define non-model parameter	Define parameter u2 to be twice the square of u: $u2 := 2*(u^2)$
*	Label parameters (the label has to be pre-multiplied)	Label the regression of Z onto X as b: $Z \sim b*X$
	Define the number of thresholds (for categorical endogenous variables)	Variable u has three thresholds: $u \mid t1 + t2 + t3$

lines 5 and 7 is to label the error variances. Line 8 is another single apostrophe to indicate the end of **R** interpreting syntax as text.

Once the model is specified, use either the `cfa()` (for *C*onfirmatory *F*actor Analysis models), or `sem()` (for *S*tructural *E*quation Models) function to fit the model. The `cfa()` and `sem()` functions are really just the calls to a more general `lavaan()` function with some of the default argument values specified differently.

`cfa()`
`sem()`
`lavaan()`

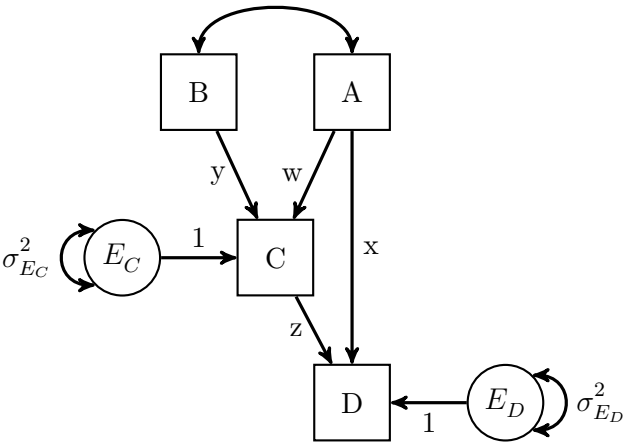


Figure 2.7 Path model example for lavaan syntax demonstration.

lavaan accepts either a covariance matrix (and sample size) or raw data as input, using the `sample.cov` (and `sample.nobs`) or `data` arguments, respectively. When using a covariance matrix as input, there is an option to also input a mean vector with the `sample.mean` argument. By default, lavaan uses normal-theory maximum likelihood as the parameter estimation technique for models with continuous variables as indicators (the typical model), but other estimators are available.

As an example, say I collected data from 500 individuals on variables *A-D* used for the path model in Figure 2.7 that I specified in `example.model`. Further, suppose I already imported the data into **R** and named it `example.data`. To estimate the parameters specified in `example.model` using `example.data`, I would use this call to lavaan:

```
example.fit <- sem(example.model, data = example.data)
```

If I wanted to use the variables' covariance matrix (named `example.cov`) instead of the raw data, I would use this call to lavaan:

```
example.fit <- sem(example.model, sample.cov = example.cov, sample.nobs = 500)
```

Both calls return nothing on-screen because the results are stored in the `example.fit` object. To obtain the lavaan results, one option is to use the `summary()` function.

```
summary(example.fit)
```

By default, lavaan's `summary()` function returns: (a) a note indicating if the parameter estimation algorithm converged; (b) the sample size; (c) estimator; (d) fit statistic (χ^2) and its degrees of freedom and *p*-value; (e) the unstandardized parameter estimates; (f) the parameter estimates' standard errors; (g) the ratio of the parameter estimates and their standard errors, (i.e., Wald statistic); and (h) the *p*-value for the Wald statistic. The `summary()` function has these default specifications for its arguments: `standardized=FALSE`, `fit.measures=FALSE`, `rsquare=FALSE`, and `modindices=FALSE`. Setting `standardized=TRUE` will include standardized estimates in the results, setting `fit.measures=TRUE` will include fit indices in the results (see Appendix A for a list and description), setting `rsquare=TRUE` will include the R^2 for each endogenous variable, and setting `modindices=TRUE` will include modification indices (see Chapter 3).

2.3 Example: Path Analysis using lavaan

As a numerical example of using lavaan, I check the parameter estimate values I calculated in Section 2.1.4. First, I input the covariance matrix in Table 2.1 (see Section 1.1.5.4). Next, I label the rows and columns; otherwise, lavaan does not know where to find the correct (co)variances.

```
# create a correlation matrix
library(lavaan)
regression.cor <- lower2full(c(1.0,0.20,1,0.24,0.30,1,0.70,0.80,0.30,1))
# name the variables in the matrix
colnames(regression.cor) <- rownames(regression.cor) <- c("X1", "X2", "X3", "Y")
regression.cor
```

To obtain the default values for the `cfa()` or `sem()` functions, as well as their various arguments and available estimators, type `?cfa` or `?sem`, respectively.

`summary()`

Many packages have their own `summary()` functions. The correct one is called based on the function that created the object.

I typically name my **R** objects with an extension that gives their type, e.g.: a covariance matrix has a `.cov` extension; a correlation matrix: `.cor`; a mean vector: `.mean`; and a SD vector: `.sd`.

```
##      X1 X2 X3 Y
## X1 1.00 0.2 0.24 0.7
## X2 0.20 1.0 0.30 0.8
## X3 0.24 0.3 1.00 0.3
## Y  0.70 0.8 0.30 1.0
```

Next, I specify the path model.

```
regression.model<-'
# structural model for Y
Y ~ a*X1 + b*X2 + c*X3
# label the residual variance of Y
Y ~~ z*Y
'
```

sem()

Last, I estimate the parameters using the `sem()` function, assuming the $n = 1000$.

```
library(lavaan)
regression.fit <- sem(regression.model, sample.cov = regression.cor, sample.nobs = 1000)
summary(regression.fit, rsquare = TRUE)
```

```
## lavaan (0.5-16) converged normally after 25 iterations
##
##   Number of observations              1000
##
##   Estimator                          ML
##   Minimum Function Test Statistic    0.000
##   Degrees of freedom                  0
##   P-value (Chi-square)                1.000
##
## Parameter estimates:
##
##   Information                        Expected
##   Standard Errors                    Standard
##
##               Estimate Std.err Z-value P(>|z|)
## Regressions:
##   Y ~
##     X1      (a)    0.571   0.008   74.539   0.000
##     X2      (b)    0.700   0.008   89.724   0.000
##     X3      (c)   -0.047   0.008   -5.980   0.000
##
## Variances:
##   Y      (z)    0.054   0.002
##
## R-Square:
##
##   Y              0.946
```

Notice that the coefficients in the *Estimate* column agree with what I found by hand, within rounding error.

2.4 Indirect Effect

All the models I have shown thus far have been concerned with direct effects of one variable on another. Another type of effect a variable can have is an **indirect effect**. An indirect effect is the influence one variable has on another variable, through a second (or third, or

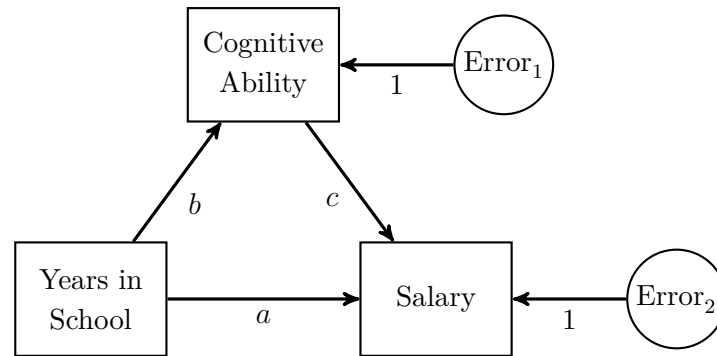


Figure 2.8 Example of a path model with an indirect effect.

fourth, ...) variable. An example of a path model with an indirect effect is shown in Figure 2.8. The model posits there is a direct influence of years in school on salary (path a), but also posits there is an indirect influence of years in school on salary through cognitive ability. The indirect influence is a compound effect. Using the tracing rules, I find that the path to get from years in school to salary through cognitive ability is $b \times c = bc$. This, then, is the measure of the indirect influence.

2.4.1 Example: Indirect Effects

Beaujean (2008) provides data to estimate the parameters in Figure 2.8, the covariances of which are in Table 2.3.

I cannot directly estimate the indirect effects, but I can use `lavaan`'s `:=` operator, which defines a new parameter based on other defined (and labeled) parameters already in the model. To use it, first I label the parameters I want to use. Then, I define the new parameter. Examine the following model specification:

```

1  beaujean.model <- '
2  salary ~ a*school + c*iq
3  school ~ b*iq
4  ind := b*c
5  '

```

Lines 1-3 are typical `lavaan` specification syntax, but in line 4 I define a new parameter, `ind`, which is the product of parameters b and c .

Estimation of the parameters follows the same steps as for the first example: input the data, specify the model, and estimate the parameters.

Table 2.3 Covariances for Figure 2.8 ($n = 300$).

	Salary	Years in School	Cognitive Ability
Salary	648.07	30.05	140.17
Years in School	30.05	8.64	25.57
Cognitive Ability	140.17	25.57	233.20

Data taken from Beaujean (2008), and are available at http://jwosborne.com/bp_ch28.html

```
# input the covariances and name the rows/columns
beaujean.cov <- lower2full(c(648.07, 30.05, 8.64, 140.18, 25.57, 233.21))
colnames(beaujean.cov) <- rownames(beaujean.cov) <- c("salary", "school", "iq")

# specify the path model
beaujean.model <- '
salary ~ a*school + c*iq
school ~ b*iq
ind:= b*c
'

# estimate parameters
beaujean.fit <- sem(beaujean.model, sample.cov=beaujean.cov, sample.nobs=300)
```

The `summary()` results are similar to those from the previous example, only now there is a *Defined Parameters* section that houses the estimate for the newly defined indirect effect parameter.

```
summary(beaujean.fit)

## lavaan (0.5-16) converged normally after 23 iterations
##
##   Number of observations              300
##
##   Estimator                          ML
##   Minimum Function Test Statistic    0.000
##   Degrees of freedom                  0
##   P-value (Chi-square)                0.000
##
## Parameter estimates:
##
##   Information                        Expected
##   Standard Errors                    Standard
##
##           Estimate Std.err Z-value P(>|z|)
##
..<Output Omitted>..

## Defined parameters:
##   ind           0.036   0.012   2.984   0.003
```

2.5 Summary

In this chapter, I discussed the basic components of creating a path model and conducting a path analysis with manifest variables. In doing so, I introduced and defined some new terms that I continue to use with latent variable models. I made a point of differentiating between direct and indirect effects, and worked through examples with both types of effects. In addition to the new terms, I demonstrated how to estimate the coefficients in a path model “by hand” as well by using the **R** package `lavaan`. It is important to understand how to use `lavaan`’s `sem()` and `cfa()` functions, as I use them throughout the rest of the book. Working through the exercises in Section 2.7 should help with your mastery of these functions.

2.6 Writing the Results

Path analysis can cover a large variety of more traditional statistical models (e.g., ANOVA, regression), so there is not a single way to write the results. Typically, though, it is good to

have a table that reports the unstandardized path coefficients and their standard errors, the standardized path coefficients, and effects sizes (e.g., R^2). If you are familiar with the \LaTeX language, the **R** package `xtable` contains the `xtable()` function that can be very helpful in creating these tables.²

`xtable()`

As an example, the following **R** syntax produced all the necessary \LaTeX syntax for Table 2.4, which includes the parameters estimated (columns 1-3), unstandardized coefficients (column 4), the standard errors (column 5), and the standardized coefficients (column 6) for the analysis I did of the model in Figure 2.2a.³

```
xtable(parameterEstimates(regression.fit, standardized=TRUE)[,c(1:3,5:6,12)],
caption="Example Table Made Using \\texttt{xtable()} Function.")
```

Table 2.4 Example Table Made Using `xtable()` Function.

	lhs	op	rhs	est	se	std.all
1	Y	~	X1	0.57	0.01	0.57
2	Y	~	X2	0.70	0.01	0.70
3	Y	~	X3	-0.05	0.01	-0.05
4	Y	~~	Y	0.05	0.00	0.05
5	X1	~~	X1	1.00	0.00	1.00
6	X1	~~	X2	0.20	0.00	0.20
7	X1	~~	X3	0.24	0.00	0.24
8	X2	~~	X2	1.00	0.00	1.00
9	X2	~~	X3	0.30	0.00	0.30
10	X3	~~	X3	1.00	0.00	1.00

Even if you do not use \LaTeX , the `xtable()` function can be useful as its onscreen results are systematically delimited with a \$ to separate columns. Thus, they can be copy-and-pasted into many spreadsheet programs for easy table creation.

Over the past few years, there has been a growing trend of making research more replicable (e.g., Asendorpf et al., 2013). To that end, it would be helpful to include a covariance matrix of the data along with the **R** syntax used for analysis in an appendix of any manuscript. The `xtable()` function can be helpful for creating these covariance matrixes. A gallery of `xtable` examples can be found at <http://cran.r-project.org/web/packages/xtable/vignettes/xtableGallery.pdf>.

In addition to tables, it is helpful to include a nicely created picture of the final path model. Typically, the path model would include the standardized path coefficients along a given path, but there is no reason that they could not be unstandardized coefficients instead (or both). Just be sure to indicate if the coefficients are unstandardized in the figure's

² There are **R** packages to translate results to Microsoft packages as well, see <http://cran.r-project.org/web/views/ReproducibleResearch.html>

³I go into more detail about `lavaan`'s `parameterEstimates()` function in Chapter 3.

Table 2.5 Correlations for Exercise 2.2 ($n = 18,058$).

	Race	SES	CogAbil	SchTyp	AcadAch
Race	1.000	0.178	0.230	0.106	0.195
SES	0.178	1.000	0.327	0.245	0.356
Cognitive Ability	0.230	0.327	1.000	0.183	0.721
School Type	0.106	0.245	0.183	1.000	0.178
Academic Achievement	0.195	0.356	0.721	0.178	1.000

Data taken from Page and Keith (1981, p. 14). *Race*: White/Hispanic=1, Black=0; *CogAbil*: Composite of verbal and nonverbal subtests; *SchTyp*: Public=0, Private=1; *AcadAch*: Composite of reading and math subtests.

caption. Nicol and Pexman (2010) provide examples of both tables and path models for use when preparing a manuscript.

2.7 Exercises

- 2.1 On this book's website there is a dataset named *MathHmwk.txt* that contains 100 observations on two variables: (a) number of hours typically spent completing math homework (*MathHomework*), and (b) score on a standardized math achievement test (*MathAchievement*). The question of interest is: Is the time spent on math homework directly related to math achievement?
 - 2.1.a Import the data into **R**, naming the object *MathHmwk.data*.
 - 2.1.b Do a typical regression using the `lm()` function.
 - 2.1.c Draw a standardized and unstandardized path model of the regression.
 - 2.1.d Do the analysis as a path model in **lavaan** using the `sem()` function. Obtain both the standardized and unstandardized path coefficients.
 - 2.1.e How are the results in Exercise 2.1.d related to the results from Exercise 2.1.b?
- 2.2 Page and Keith (1981) used respondents from the 1980 *High School and Beyond* data to investigate the following question: What is the relationship between school type and academic achievement? The data are given in Table 2.5 in the form of a correlation matrix; a path model for the analysis is shown in Figure 2.9.
 - 2.2.a Input the correlation matrix into **R**.
 - 2.2.b Create the path model using **lavaan** syntax. Be sure to label the parameters the same as Figure 2.9.
 - 2.2.c What is the relationship between school type and academic achievement (i.e., path j)?
- 2.3 MacKinnon (2008, p. 113) provides a dataset from a hypothetical study of teacher expectancies and student achievement. His path model is shown in Figure 2.10 and the covariances for the model are given in Table 2.6.
 - 2.3.a Input the covariances into **R**.
 - 2.3.b Write the syntax for the model. Use the `:=` operator to define both indirect effects from teacher expectancies to student achievement ($a_1 \times b_1$ and $a_2 \times b_2$).

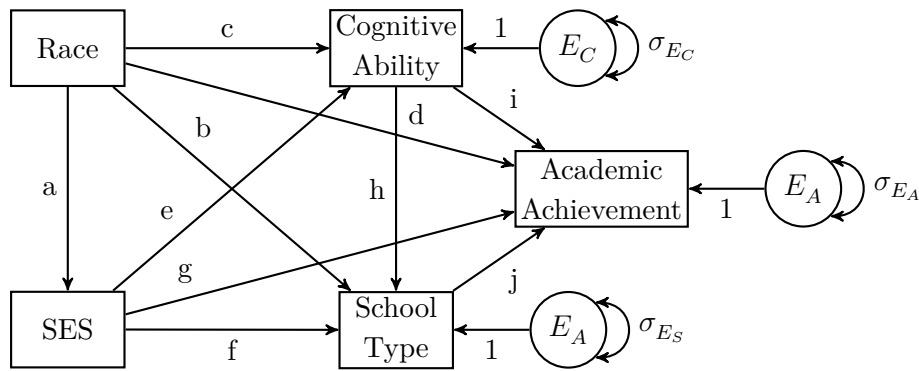


Figure 2.9 Path model for Exercise 2.2.

2.3.c What are the indirect effects?

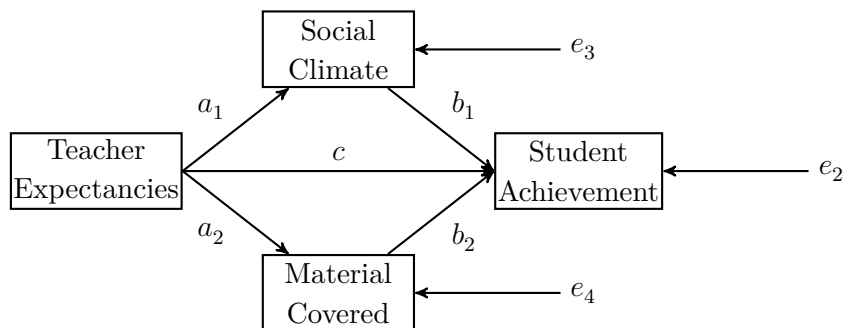


Figure 2.10 Path model for Exercise 2.3.

Table 2.6 Covariances for Exercise 2.3 ($n = 40$).

	Teacher Expectancies	Social Climate	Material Covered	Student Achievement
Teacher Expectancies	84.85	71.28	18.83	60.05
Social Climate	71.28	140.34	-6.25	84.54
Material Covered	18.83	-6.25	72.92	37.18
Student Achievement	60.05	84.54	37.18	139.48

2.8 References & Further Readings

- Asendorpf, J. B., Conner, M., De Fruyt, F., De Houwer, J., Denissen, J. J. A., Fiedler, K., . . . Wicherts, J. M. (2013). Recommendations for increasing replicability in psychology. *European Journal of Personality*, 27, 108-119. doi: 10.1002/per.1919
- Beaujean, A. A. (2008). Mediation, moderation, and the study of individual differences. In J. W. Osborne (Ed.), *Best practices in quantitative methods* (pp. 422-442). Thousand Oaks, CA: Sage.
- Boker, S. M., & McArdle, J. J. (2005). Path analysis and path diagrams. In B. Everitt & D. C. Howell (Eds.), *Encyclopedia of behavioral statistics* (Vol. 3, pp. 1529-1531). West Sussex, England: Wiley.
- MacKinnon, D. P. (2008). *Introduction to statistical mediation analysis*. Mahwah, NJ: Erlbaum.
- Nicol, A. A. M., & Pexman, P. M. (2010). *Presenting your findings: A practical guide for creating tables* (6th ed.). Washington, DC: American Psychological Association.
- Page, E. B., & Keith, T. Z. (1981). Effects of U.S. private schools: A technical analysis of two recent claims. *Educational Researcher*, 10, 7-17. doi: 10.2307/1174256
- Rosseel, Y. (2012). lavaan: An **R** package for structural equation modeling. *Journal of Statistical Software*, 48(2), 1-36. Retrieved from www.jstatsoft.org/v48/i02/.
- Wolfe, L. M. (1999). Sewall Wright on the method of path coefficients: An annotated bibliography. *Structural Equation Modeling: A Multidisciplinary Journal*, 6, 280-291. doi: 10.1080/10705519909540134
- Wolfe, L. M. (2003). The introduction of path analysis to the social sciences, and some emergent themes: An annotated bibliography. *Structural Equation Modeling: A Multidisciplinary Journal*, 10, 1-34. doi: 10.1207/S15328007SEM1001_1

3 | Basic Latent Variable Models

Chapter Contents

3.1	Background	37
3.2	Latent Variable Models	38
3.2.1	Identification of Latent Variable Models	39
3.3	Example: Latent Variable Model with One Latent Variable	42
3.3.1	Alternative Latent Variable Scaling	47
3.3.2	Example: Latent Variable Model with Two Latent Variables	48
3.4	Example: Structural Equation Model	50
3.5	Summary	51
3.6	Writing the Results	51
3.7	Exercises	52
3.8	References & Further Readings	55

3.1 Background

A **structural equation model** (SEM) is a broad class of statistical models (see Figure 3.1) that consists of two parts: the structural model and the latent variable model. The structural model consists of the regression-like relationships among the variables (i.e., the path analysis from Chapter 2). The **latent variable model** (LVM) forms the latent variables (LVs) used in the structural model. When a LVM is analyzed without a structural model, it is sometimes called a **confirmatory factor analysis** (CFA). If there was not a hypothesized structure for the latent variable model, then it would be an **exploratory factor analysis** (EFA). The majority of this book is focused on confirmatory LVMs, although in Section 3.4 I demonstrate fitting a full SEM. Beaujean (2013) demonstrates how to conduct an EFA in **R**.

*Structural
Equation Model
(SEM)
Latent Variable
Model (LVM)
Confirmatory
Factor Analysis
(CFA)
Exploratory
Factor Analysis
(EFA)*

The *factor* in factor analysis is synonymous with a latent variable.

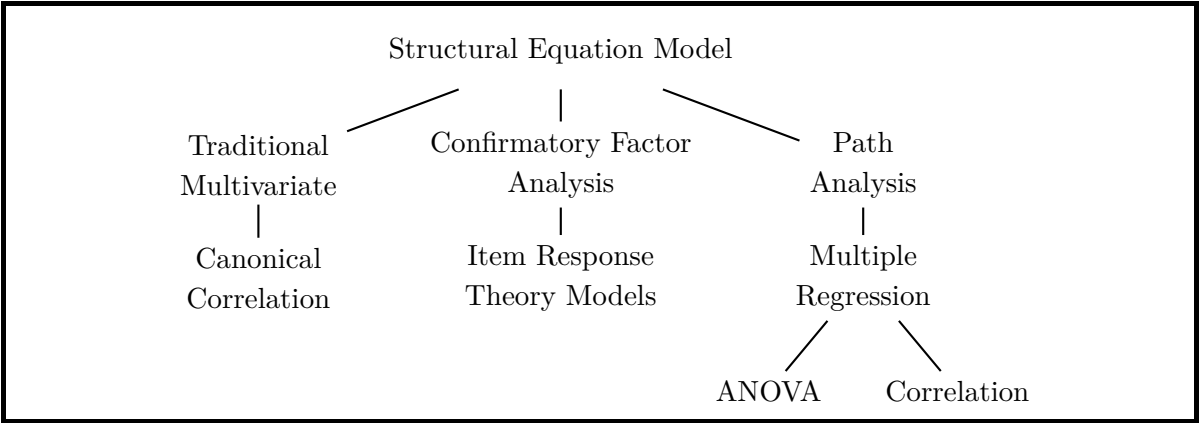


Figure 3.1 Example of data analysis methods subsumed by a structural equation model.

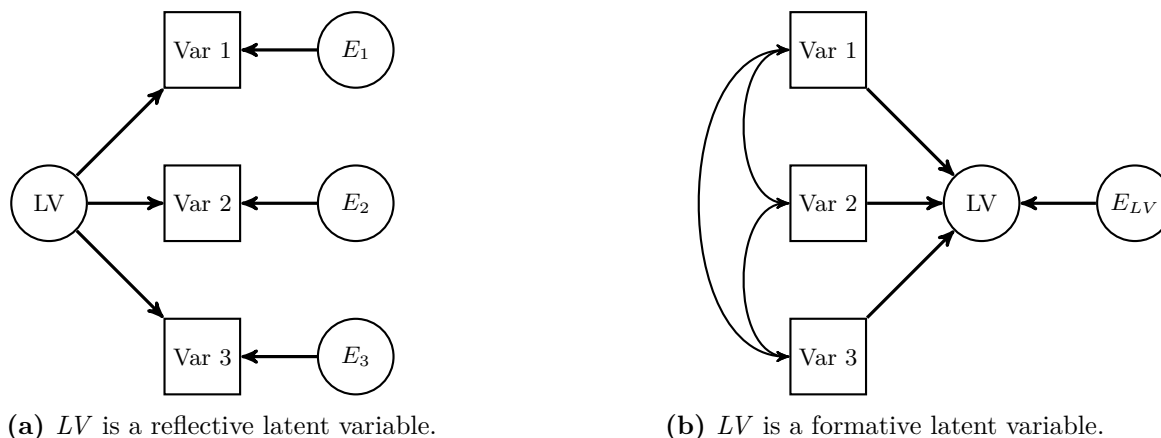


Figure 3.2 Reflective and formative latent variable path models.

3.2 Latent Variable Models

There are two types of latent variables: **reflective** and **formative**. Reflective LVs are thought to cause other variables to covary, whereas formative LVs are thought to be the result of variables' covariation (similar to a regression model). Reflective and formative LVs are shown in Figure 3.2a and Figure 3.2b, respectively. I focus on reflective LVs in this book, although I give some further reading on formative LVs in Section 3.8, and Exercise 3.3 requires the use of a formative LV.

Reflective and Formative Latent Variables

Indicator Variable

Factors are used with ANOVA models, too. For the purposes of this book, consider that usage unrelated to its use with LVMs.

Not all definitions of a LV would include the error term as a latent variable. See Bollen (2002) for more about alternative definitions.

The purpose of a (reflective) LVM is to understand the underlying structure that produced relationships among multiple manifest variables (i.e., covariance matrix). The manifest variables (MVs) that are directly influenced by the LV are the **indicator variables**. The proposed causal agents in these underlying structures are LVs. I first introduced latent variables in Chapter 2 when discussing regression, as the error term is a latent variable: it is unobserved and influences a manifest variable (i.e., the outcome/endogenous variable); it is measured differently than typical LVs, though.

The idea behind a LVM is that there are a small number of LVs within a given domain (e.g., personality, self-efficacy, religiosity) that influence each of its indicator variables and, hence, produce the observed covariances. Thus the variation (or covariation if there is more than one) in the LVs causes (loosely defined) covariation in the indicators; conversely, covariation in the indicator variables is due to their dependence on one or more LVs. Latent variable modeling, then, is the method used to identify or confirm the number of the LVs that produce the observed covariation in the indicator variables as well as understand the nature of those LVs (e.g., what they predict, what variables predict them).

In path models, LVs are represented using ellipses. Figure 3.3 contains an example of a one-factor LVM using subtests from the Wechsler Intelligence Scale for Children-Fourth Edition (WISC-IV; Wechsler, 2003). It has one LV, *g*, and five MVs, each of which is an indicator variable.

g stands for *general intelligence*.

Factor Loading Pattern Coefficient

One measure of the influence a LV has on MVs is the **factor loading**. These are akin to regression/path coefficients. Some scholars advocate using the term **pattern coefficient** in-

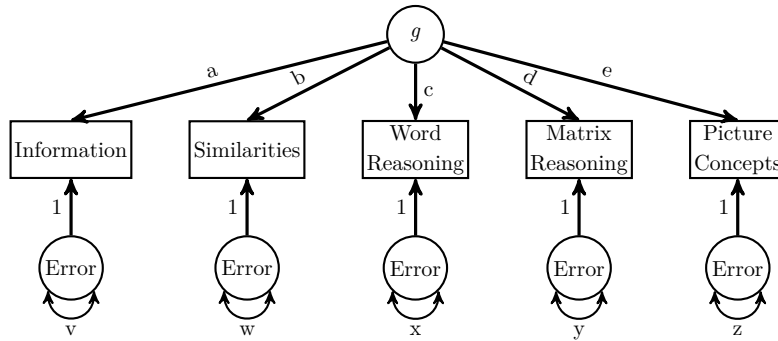


Figure 3.3 Single-factor model of five Wechsler Intelligence Scale for Children-Fourth Edition subtests.

stead of factor loading because factor loading is an ambiguous term that can be confused with **structure coefficient**, which are the correlations between the MVs and the model's LVs. For models with one LV, pattern and structure coefficients have the same values. For models with more than one LV (see Section 3.3.2), they have different values unless all the LVs are uncorrelated with each other. To minimize confusion in this book, when I use the terms *factor loading* or *loading*, I am referring to a pattern coefficient. When I discuss structure coefficients, I use the term explicitly.

*Structure
Coefficient*

In Figure 3.3, a , b , c , d , and e are all factor loadings. I can obtain something akin to a R^2 value for each MV using the tracing rules discussed in Section 2.1.3: find all the legitimate paths that go from a MV to the LV(s) and return back to the same MV. This value is called the **communality**. Conversely, the **uniqueness** is the amount of variance in the MV *not* explained by the model's LVs. For example, to find the communality for the *Information* MV in Figure 3.3, I can go to the LV, g , and then back to Information only through the a path (twice), thus the amount of variance of the Information MV that g explains is a^2 . This makes uniqueness of the Information variable: $1 - a^2 = v$.

Communality

Uniqueness

3.2.1 Identification of Latent Variable Models

At the heart of model **identification** is the question: Is there enough *non-redundant* information in the data to be able to estimate the required parameters *uniquely*? With regression, identification is never an issue because they are always **just-identified** models, meaning that the number of parameters to estimate exactly equals the amount of non-redundant information in the data. For now, think of the amount of non-redundant information in the data as being the number of non-redundant variances/covariances in the dataset. This can be easily calculated using the formula in Equation (3.1).

*Identification of
Latent Variable*

*Just-Identified
Model*

$$\text{non-redundant information in a dataset} = \frac{p(p+1)}{2} \quad (3.1)$$

where p is the number of manifest variables.

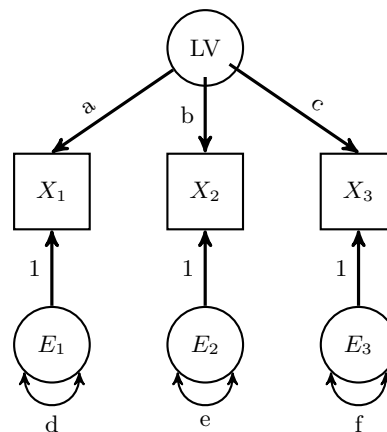
*Underidentified
Model*

With a LVM, model identification is more complex than with regression, as the models can be just-identified, **underidentified**, or **overidentified**. If there are more parameters to estimate than there are pieces of non-redundant information, then the model is underidenti-

*Overidentified
Model*

	X_1	X_2	X_3
X_1	σ_1^2		
X_2	σ_{12}	σ_2^2	
X_3	σ_{13}	σ_{23}	σ_3^2

(a) Covariance matrix showing non-redundant elements.



(b) Latent variable model.

Figure 3.4 Example with three indicator variables.

fied. Conversely, if there are more pieces of non-redundant information than there are parameters to estimate, then the model is usually overidentified. If a model is underidentified, the model's parameters cannot be estimated uniquely and `lavaan` (or any other LVM program) will return an error message. If a model is over- or just-identified, there *should* be unique estimates for each parameter. For overidentified models, not only do they provide unique parameter estimates, but they can also provide measures of model fit (see Appendix A). This is because they have **degrees of freedom** (df). In LVMs, df can be thought of as the number of non-redundant pieces of information in the data minus the number of parameters to estimate. Only overidentified models have $df > 0$.

Degrees of Freedom

For factor and path analysis, identification is relatively simple to determine. Most of the difficulty with identification is with full SEMs.

Model identification can be tricky. Instead of going into all the complexities involved, I give four rules-of-thumb conditions in the next three sections that should work for most models with reflective LVs. Meeting these conditions typically produces an overidentified LVM, or at least a just-identified LVM.

3.2.1.1 Number of Indicator Variables

If every latent variable in a model has at least four indicator variables and none of their error variances covary, then there should not be a problem with identification. *Why four variables?*, you may ask. Examine Figure 3.4 where there are $3 \times 4/2 = 6$ pieces of non-redundant information, and 6 parameters to estimate. Thus, it is just-identified. Now examine Figure 3.5, where there are $4 \times 5/2 = 10$ pieces of non-redundant information, but only 8 parameters to estimate, making it overidentified.

Even if a LV cannot have at least four indicators, it can still be just-identified under *any* of the following conditions.

1. The LV has three indicator variables, and the error variances do not covary.
2. The LV has at least two indicators with non-covarying error variances *and* the indicator variables' loadings are set equal to each other.
3. The LV has one indicator, the directional paths are set to one, *and* its error variance is fixed to some value. The fixed value is usually either:

- (a) 0.0, meaning the indicator variable is measured with perfect reliability, or
- (b) $(1 - r_{XX'})\sigma_X^2$, where $r_{XX'}$ and σ_X^2 are the variable's reliability and variance, respectively.

An example of a single-indicator LVM is shown in Figure 3.6.

3.2.1.2 Latent Variable's Scale

Because LVs are not directly observed, there are no inherent units by which to measure them. Consequently, a LVM is not identified unless some parameter estimates are constrained to set the latent variable's scale. There are three common ways to set this scale.

1. *Standardized latent variable.* This method constrains the latent variable's variance to 1.0. This, in effect, makes the latent variable a standardized variable (i.e, on a Z score scale; see Section 1.1.12.2). If the indicator variables are standardized as well, the loadings can be interpreted the same as a standardized regression coefficient: the number of standard deviations the MV changes as the LV increases one standard deviation. Moreover, if there is more than one LV, then the covariance among the LVs becomes a correlation.
2. *Marker variable.* This method requires a single factor loading for each LV be constrained to an arbitrary value (usually 1.0). The indicator variable whose loading is constrained is called the *marker variable*. This method uses the marker variable to define the LV's variance.
3. *Effects-coding.* This method estimates all the loadings, but constrains that the loadings for a given LV average 1.0, or, equivalently, that their sum is equal to the number of unique indicator variables.

These different scaling methods produce different values for the model's parameters, but should not alter how a model fits the data. I demonstrate all three scaling methods in Section 3.3.1.

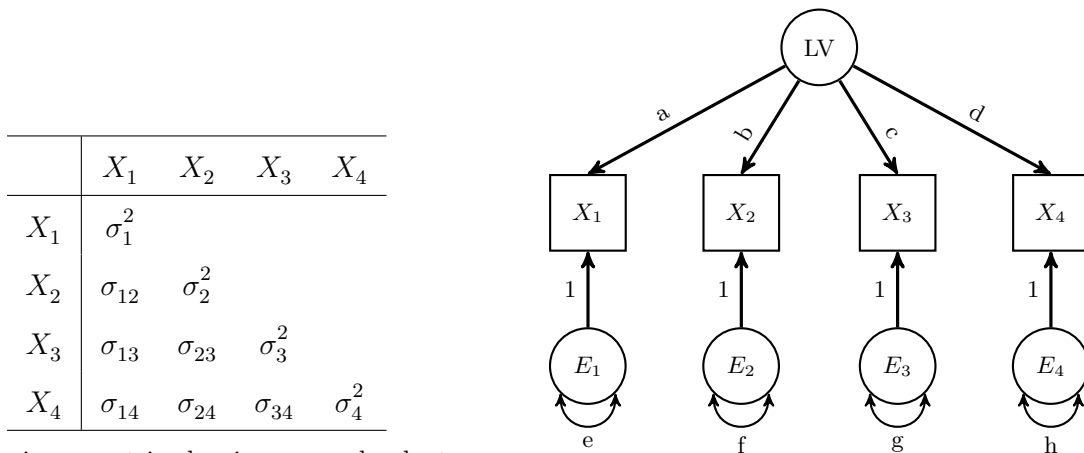


Figure 3.5 Example with four indicator variables.

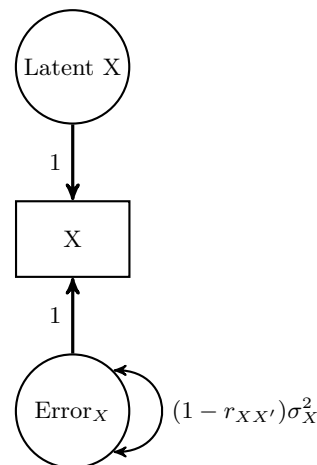


Figure 3.6 Example of a latent variable model with a single indicator.

3.2.1.3 Other Conditions

There are two other conditions for the rules-of-thumb. It is atypical for these conditions not to be met, so I do not discuss them in any detail.

1. If there is more than one LV in the model, then for every pair of LVs, either:
 - (a) There is at least one indicator variable per LV whose error variance does not covary with the error variance of the other LV's indicator variables, *or*
 - (b) The covariance between the pair of LVs is constrained to a specified value.
2. For every indicator variable, there must be at least one other indicator variable (of the same LV or a different LV) with whom the error variances do not covary.

3.2.1.4 Empirical Underidentification

Empirical underidentification is the situation where a LVM is only identified if one of the parameters (usually a factor correlation or a factor loading) is not equal, or very close to, 0.0.¹ An example is shown in Figure 3.7. As long as $|e| > 0$, the model in Figure 3.7a is overidentified because there are $4 \times 5/2 = 10$ pieces of information with 9 parameters to estimate. However if $e = 0$, then the model requires the estimation of two separate LVs, as is the case in Figure 3.7b. For both LVs, there are $2 \times 3/2 = 3$ pieces of non-redundant information, but 4 parameters to estimate, making them each underidentified.

Empirical Underidentification

3.3 Example: Latent Variable Model with One Latent Variable

I demonstrate fitting a LVM in **R** by estimating the parameters for the model in Figure 3.3 using the data in Table 3.1. My syntax to enter the correlations and standard deviations (SDs) is given below.

```
library(lavaan)
# convert vector of correlations into matrix
wisc4.cor <- lower2full(c(1,0.72,1,0.64,0.63,1,0.51,0.48,0.37,1,0.37,0.38,0.38,0.38,1))
# name the variables in the matrix
colnames(wisc4.cor) <- rownames(wisc4.cor) <- c("Information", "Similarities",
"Word.Reasoning", "Matrix.Reasoning", "Picture.Concepts")
```

¹Sometimes the issue of empirical underidentification can arise if factor correlations are very close to 1.0 as well.

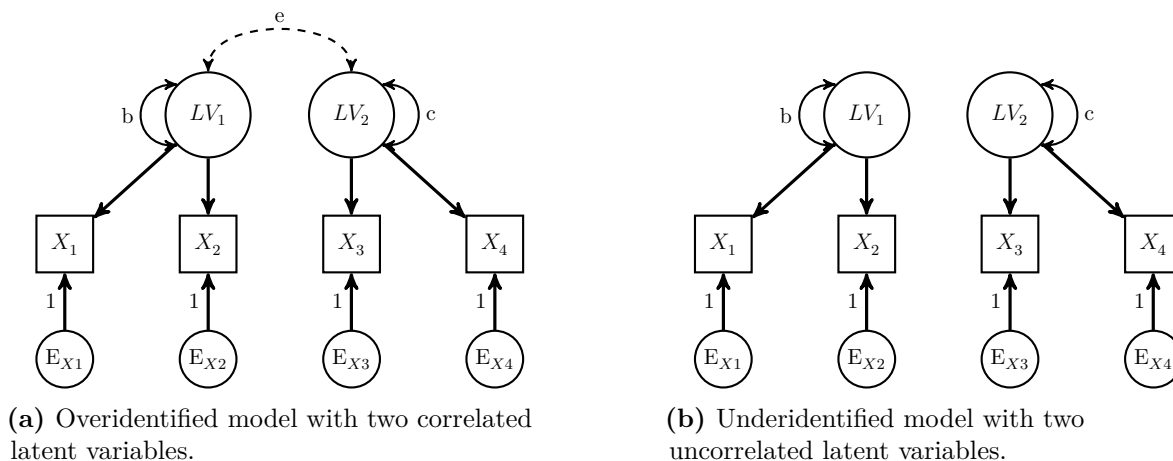


Figure 3.7 Example model showing empirical underidentification.

```
# enter the SDs
wisc4.sd <- c(3.01 , 3.03 , 2.99 , 2.89 , 2.98)
names(wisc4.sd) <- c("Information", "Similarities", "Word.Reasoning", "Matrix.Reasoning",
"Picture.Concepts")
```

Next, I transform the correlations to covariances. Using `lavaan`'s `cor2cov()` function makes the conversion an easy process. It requires two arguments: a correlation matrix and a SD vector.

```
# convert correlations and SDs to covainces
wisc4.cov <- cor2cov(wisc4.cor, wisc4.sd)
```

To analyze these covariances, I need to specify the model in `lavaan` using the terms in Table 2.2.

```
wisc4.model<-'
g =~ a*Information + b*Similarities + c*Word.Reasoning + d*Matrix.Reasoning +
e*Picture.Concepts
'
```

`cor2cov()`

If there is an option, covariances are preferred over correlations for most LVMs.

I labeled the factor loadings to match Figure 3.3. This is not required, but makes it easier to interpret the output.

Table 3.1 Correlations and Standard Deviations of Selected Wechsler Intelligence Scale for Children-Fourth Edition Subtests ($n = 550$) to accompany Figure 3.3.

	Information	Similarities	Word Reasoning	Matrix Reasoning	Picture Concepts
Information	1.00				
Similarities	0.72	1.00			
Word Reasoning	0.64	0.63	1.00		
Matrix Reasoning	0.51	0.48	0.37	1.00	
Picture Concepts	0.37	0.38	0.38	0.38	1.00
SD	3.01	3.03	2.99	2.89	2.98

Data taken from Parkin and Beaujean (2012, pp. 117-118).

The LV, g , is defined ($=\sim$) using all five WISC-IV subtests. I separate all the indicator variables for the same LV using the $+$ sign before each variable after the first. To estimate the parameters, I use the `cfa()` function.

`cfa()`

```
wisc4.fit <- cfa(model=wisc4.model, sample.cov=wisc4.cov, sample.nobs=550,
std.lv=FALSE)
```

The first two arguments to the `cfa()` function are the model and the data (in this example, the covariance matrix), which I have already defined in **R**. The third argument, `sample.nobs`, tells **lavaan** the sample size from which the covariance matrix was calculated; it needs to be used every time a covariance matrix is used for input. The last argument, `std.lv=FALSE` is actually the default option and I did not have to include it. It tells **lavaan** to estimate the LV's variance using the first indicator variable (Information) by constraining its loading to 1.0, thus scaling g using the marker variable method. If `std.lv=TRUE`, this would scale g by standardizing it, i.e., estimate all the loadings and constrain the g 's variance to 1.0.

`summary()`

`parameterEstimates()`

To print the parameter estimates, use the `summary()` or `parameterEstimates()` function. Specifying `standardized=TRUE` in either function produces both the unstandardized and standardized estimates. The unstandardized estimates (*Estimate*) use the raw score scales for the MVs and use the marker variable method to scale the LV. There are two types of standardized estimates, though. The first (*Std.lv*) standardizes the LV, but leaves the MVs in the raw score scale. The second (*Std.all*) standardizes both the LV and all the MVs.²

Unless stated differently, authors typically refer to the *Std.all* estimates when discussing standardized parameter estimates.

```
summary(wisc4.fit, standardized = TRUE)
```

```
## lavaan (0.5-16) converged normally after 30 iterations
```

```
..<Output Omitted>..
```

```
##              Estimate Std.err Z-value P(>|z|) Std.lv Std.all
## Latent variables:
##   g =~
##   Informatn (a)    1.000
##   Similarts (b)    0.985    0.045   21.708    0.000    2.578    0.857
##   Wrdr.Rsnng (c)    0.860    0.045   18.952    0.000    2.541    0.839
##   Mtrx.Rsnn (d)    0.647    0.045   14.822    0.000    2.217    0.742
##   Pctr.Cncp (e)    0.542    0.050   10.937    0.000    1.669    0.578
##
## Variances:
##   Information      2.395    0.250
##   Similarities     2.709    0.258
##   Word.Reasonng    4.009    0.295
##   Matrix.Resnng    5.551    0.360
##   Pictur.Cncpts    6.909    0.434
##   g                6.648    0.564
##               2.395    0.250    2.395    0.265
##               2.709    0.258    2.709    0.296
##               4.009    0.295    4.009    0.449
##               5.551    0.360    5.551    0.666
##               6.909    0.434    6.909    0.779
##               6.648    0.564    1.000    1.000
```

```
parameterEstimates(wisc4.fit, standardized = TRUE, ci = FALSE)
```

```
##              lhs op              rhs label est   se std.lv std.all std.nox
## 1              g =~      Information    a 1.00 0.000    2.6    0.86    0.86
```

²The `parameterEstimates()` function produces a third type of standardized variable, *std.nox*. With this standardization, all variables are standardized except for the exogenous MV.

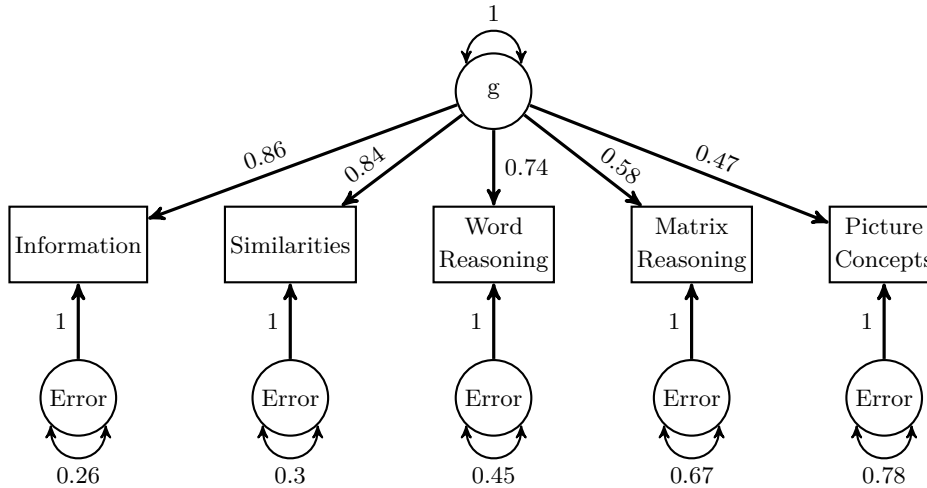


Figure 3.8 Single-factor model of five Wechsler Intelligence Scale for Children-Fourth Edition subtests with standardized parameter estimates.

## 2	g ==	Similarities	b	0.98	0.045	2.5	0.84	0.84
## 3	g ==	Word.Reasoning	c	0.86	0.045	2.2	0.74	0.74
## 4	g ==	Matrix.Reasoning	d	0.65	0.047	1.7	0.58	0.58
## 5	g ==	Picture.Concepts	e	0.54	0.050	1.4	0.47	0.47
## 6	Information ==	Information		2.40	0.250	2.4	0.27	0.27
## 7	Similarities ==	Similarities		2.71	0.258	2.7	0.30	0.30
## 8	Word.Reasoning ==	Word.Reasoning		4.01	0.295	4.0	0.45	0.45
## 9	Matrix.Reasoning ==	Matrix.Reasoning		5.55	0.360	5.6	0.67	0.67
## 10	Picture.Concepts ==	Picture.Concepts		6.91	0.434	6.9	0.78	0.78
## 11	g ==	g		6.65	0.564	1.0	1.00	1.00

The top part of `lavaan`'s `summary()` output can be used to double-check to make sure the model was specified correctly. For this model:

$$n = 550$$

$$df = \underbrace{(5 \times 6/2)}_{\text{non-redundant information}} - \underbrace{(4 + 5)}_{\text{loadings + error variances}} - \underbrace{(1)}_{\text{latent variances}} = 15 - 9 - 1 = 5$$

both of which are correct. The *Latent variables* section of the output gives the unstandardized and standardized parameter estimates (in this example, they are all loadings), both of which indicate that each indicator is a relatively strong measure of g , with Information being the most saturated (i.e., having the strongest loading) and Picture Concepts being the least. The *Variances* section gives the variances of the exogenous variables and the error variances of the endogenous variables. A path model with all the standardized parameter estimates is shown in Figure 3.8.

`lavaan` does not produce communality estimates, but I can calculate them using the tracing rules. For example, the communality estimate for the Information subtest is $a \times 1 \times a = a^2 = 0.86^2 = 0.74$, thus its uniqueness is $1 - a^2 = 1 - 0.86^2 = 0.26$. Table 3.2 shows the factor loadings and communality estimates for all the indicator variables.

Table 3.2 Standardized Loading and Community Estimates for Figure 3.3.

	Loading	Community
Information	0.86	0.74
Similarities	0.84	0.70
Word Reasoning	0.74	0.55
Matrix Reasoning	0.58	0.33
Picture Concepts	0.47	0.22

In addition to the communality estimates, I can use the tracing rules and standardized parameter estimates to calculate the correlations implied by the model's parameter estimates, which I can then compare them to the actual correlations (i.e, residual correlations). Using the labels from Figure 3.3, the relationship between the Information and Similarities subtests is defined by ab since that is the only way to go from Similarities from Information. Plugging in the values for a and b from Table 3.2, the model-implied correlation is $(0.86)(0.84) = 0.72$. To the hundredths place, this exactly matches the sample correlation.

The `fitted()` function in `lavaan` returns all the model-implied *covariances*. A little manipulation of this output using `R`'s `cov2cor()` function returns the implied correlations. I can then compare these to the actual correlations using the `residuals()` function with the additional `type="cor"` argument.

`fitted()`
`cov2cor()`

`residuals()`

Examining the residual correlations is good practice when fitting LVMs, and is the idea behind one measure of model fit: the standardized root mean square residual (see Appendix A).

```
# model-implied covariances
fitted(wisc4.fit)

# transform model-implied covariances to correlations
wisc4Fit.cov <- fitted(wisc4.fit)$cov
wisc4Fit.cor <- cov2cor(wisc4Fit.cov)

# original correlations
wisc4.cor
```

```
# residual correlations
residuals(wisc4.fit, type = "cor")

## $cor
##           Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information      0.000
## Similarities      0.000 0.000
## Word.Reasoning     0.004 0.007 0.000
## Matrix.Reasoning   0.014 -0.005 -0.059 0.000
## Picture.Concepts  -0.033 -0.014 0.031 0.109 0.000
##
## $mean
##           Information      Similarities      Word.Reasoning      Matrix.Reasoning      Picture.Concepts
##                0                0                0                0                0
```

`fitMeasures()`

The `fitMeasures()` function returns measures of model fit. I discuss these model fit measures in Appendix A.

```
fitMeasures(wisc4.fit)
```

```
##          fmin          chisq          df          pvalue
##          0.024          26.775          5.000          0.000
##    baseline.chisq    baseline.df    baseline.pvalue          cfi
##    1073.427          10.000          0.000          0.980
##          tli          nnfi          rfi          nfi
##          0.959          0.959          0.950          0.975
##          pnfi          ifi          rni          logl
##          0.488          0.980          0.980          -6378.678
##  unrestricted.logl          npar          aic          bic
##          -6365.291          10.000          12777.357          12820.456
##          ntotal          bic2          rmsea    rmsea.ci.lower
##          550.000          12788.712          0.089          0.058
##    rmsea.ci.upper    rmsea.pvalue          rmr          rmr_nomean
##          0.123          0.022          0.298          0.298
##          srmr          srmr_bentler    srmr_bentler_nomean    srmr_bollen
##          0.034          0.034          0.034          0.034
##  srmr_bollen_nomean    srmr_mplus    srmr_mplus_nomean          cn_05
##          0.034          0.034          0.034          228.408
##          cn_01          gfi          agfi          pgfi
##          310.899          0.982          0.947          0.327
##          mfi          ecvi
##          0.980          0.085
```

If I believed the model inadequately fit the data, I could use the `modificationIndices()` function to determine if there were places in the model to adjust.

```
modificationIndices()
```

```
modificationIndices(wisc4.fit)
```

```
##          lhs op          rhs    mi    epc sepc.lv sepc.all sepc.noX
## 1          g =~    Information    NA    NA    NA    NA    NA
## 2          g =~    Similarities 0.00 0.000 0.000 0.000 0.000
## 3          g =~    Word.Reasoning 0.00 0.000 0.000 0.000 0.000
## 4          g =~    Matrix.Reasoning 0.00 0.000 0.000 0.000 0.000
## 5          g =~    Picture.Concepts 0.00 0.000 0.000 0.000 0.000
## 6    Information ~~    Information 0.00 0.000 0.000 0.000 0.000
## 7    Information ~~    Similarities 0.01 0.034 0.034 0.004 0.004
## 8    Information ~~    Word.Reasoning 0.28 0.147 0.147 0.016 0.016
## 9    Information ~~    Matrix.Reasoning 1.45 0.280 0.280 0.032 0.032
## 10   Information ~~    Picture.Concepts 5.49 -0.565 -0.565 -0.063 -0.063
```

```
..<Output Omitted>..
```

```
## 21          g ~~          g 0.00 0.000 0.000 0.000 0.000
```

The modification index (column *mi*) is a rough estimate of how much the χ^2 test statistic would improve by unconstraining the parameter (see Appendix A for an elaboration of the χ^2 test statistic). The expected parameter change (column *epc*) is the expected value of the parameter if it were estimated (*sepc* is a standardized version of *epc*).

3.3.1 Alternative Latent Variable Scaling

In the previous example, I fit the model by using the marker variable method to scale the LV, which is `lavaan`'s default method. Now, I fit the same model using the standardized LV and effects-coding methods of LV scaling.

I can standardize the LV in one of two ways. First, I could do it explicitly in the model's specification. Alternatively, I could use the `std.lv=TRUE` argument in the `cfa()` function. To standardize the LV using the model's specification, I have to constrain the LV's variance to 1.0 and free the loading for the first MV variable. To free the loading, premultiply the variable by `NA`, which is **R**'s default code for a missing value.

```
# marker variable
wisc4.model.Std<-'
g =~ NA*Information + a*Information + b*Similarities + c*Word.Reasoning +
d*Matrix.Reasoning + e*Picture.Concepts

# constrain the LV variance to 1
g~~1*g
'

wisc4.fit.Std <- cfa(wisc4.model.Std, sample.cov=wisc4.cor, sample.nobs=550)
# equivalent model
wisc4.fit.Std <- cfa(wisc4.model, sample.cov=wisc4.cor, sample.nobs=550, std.lv=TRUE)
```

The effects-coding method is a little trickier. Not only do I have to free the first indicator variable's loading, but I have to constrain the sum of all the loadings as well. Specifically, the sum of the loading estimates for a given LV needs to equal the number of the LV's indicator variables. I issue this constraint by using the `==` logical operator.³

```
# effects-coding
wisc4.model.effects<-'
g =~ NA*Information + a*Information + b*Similarities + c*Word.Reasoning +
d*Matrix.Reasoning + e*Picture.Concepts
# constrain the loadings to sum to one
a + b + c + d + e == 5
'

wisc4.fit.effects <- cfa(wisc4.model.effects, sample.cov=wisc4.cor, sample.nobs=550)
```

3.3.2 Example: Latent Variable Model with Two Latent Variables

Lets say the model in Figure 3.3 is wrong; it should have had two LVs instead of one. Specifically, *g* should be replaced with LVs representing Verbal Comprehension and Fluid Reasoning, as is shown in Figure 3.9. The steps in estimating this model are very similar to the model with one LV, except I have to specify a slightly different model.

```
# two-factor model of the WISC-IV data
wisc4.model2<-'
V =~ a*Information + b*Similarities + c*Word.Reasoning
F =~ d*Matrix.Reasoning + e*Picture.Concepts
V~~f*F
'

wisc4.fit2 <- cfa(wisc4.model2, sample.cov=wisc4.cov, sample.nobs=550)
```

The results are given in Table 3.3. The table includes both loadings (pattern coefficients) and structure coefficients (see Section 3.2). To calculate the structure coefficients, I used the tracing rules to trace the paths from a MV to a given LV. For example the structure coefficient for the Information subtest and the Fluid LV is: $af = (0.86)(0.82) = 0.71$.

³The less than, <, and greater than, >, logical operators can be used in **lavaan** as well.

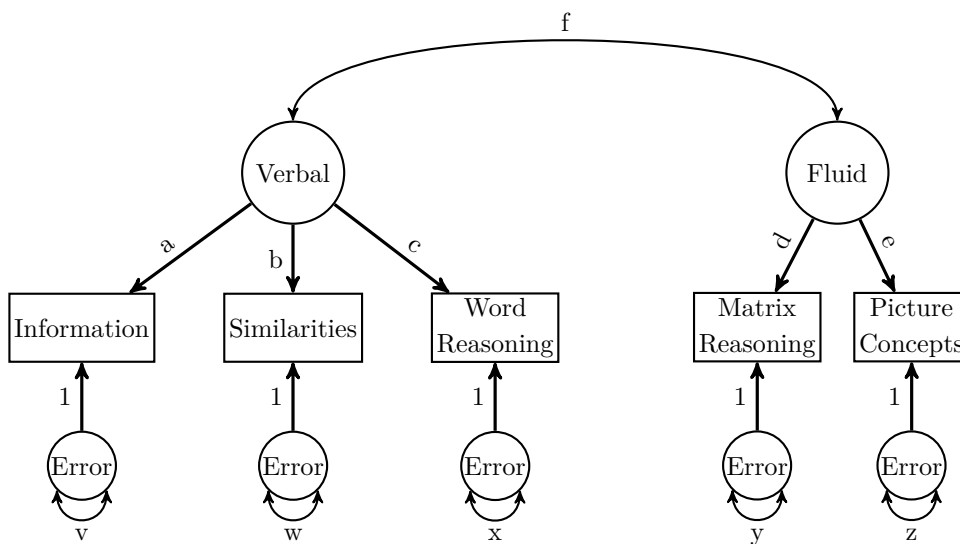


Figure 3.9 Model of five subtests from the Wechsler Intelligence Scale for Children-Fourth Edition with two latent variables.

Table 3.3 Standardized Factor Pattern, Structure, and Correlation Coefficients for the Latent Variable Model in Figure 3.9.

Variable	Factor Pattern		Factor Structure		Communality
	V	F	V	F	
Information	0.86	0.00	0.86	0.71	0.74
Similarities	0.84	0.00	0.84	0.71	0.71
Word Reasoning	0.74	0.00	0.74	0.61	0.55
Matrix Reasoning	0.00	0.69	0.57	0.69	0.47
Picture Concepts	0.00	0.55	0.45	0.55	0.30

Factor Correlations			
	V	F	
V	1.00	0.82	
F	0.82	1.00	

V: Verbal Comprehension; F: Fluid Reasoning.

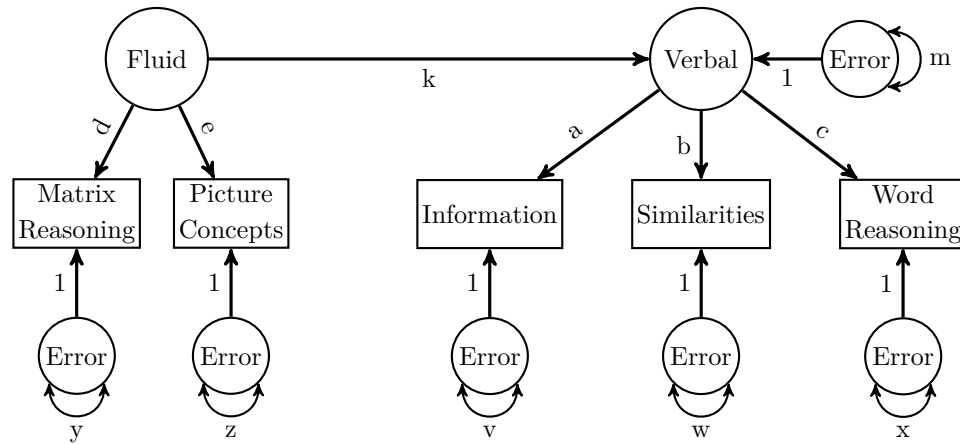


Figure 3.10 Example of a structural equation model using five Wechsler Intelligence Scale for Children-Fourth Edition subtests.

3.4 Example: Structural Equation Model

A structural equation model (SEM) typically combines a LVM with a path analysis (i.e., the structural equations). An example of a SEM using the WISC-IV data is shown in Figure 3.10. Here, the Fluid Reasoning LV is thought to have a direct influence on the Verbal Comprehension LV.

To specify a SEM in `lavaan` requires using both the LV definition (`=~`) and regression (`~`) operations.

```
# structural equation model
wisc4SEM.model <- '
# define latent variables
V =~ a*Information + b*Similarities + c*Word.Reasoning
F =~ d*Matrix.Reasoning + e*Picture.Concepts
# define structural relations
V~k*F
'
wisc4SEM.fit <- cfa(wisc4SEM.model, sample.cov=wisc4.cov, sample.nobs=550)
```

The `summary()` function's output for a SEM is similar to that from the LVMs I fit in this chapter, except now the output has both a *Latent variables* and a *Regressions* section. Of note, because the Verbal Comprehension LV is endogenous in Figure 3.10, its reported variance in the *Variances* section is the *error* variance, i.e., the variance not explained by Fluid Reasoning.

```
summary(wisc4SEM.fit, standardized = TRUE)

## lavaan (0.5-16) converged normally after 39 iterations

..<Output Omitted>..

##
##               Estimate Std.err Z-value P(>|z|)   Std.lv Std.all
## Latent variables:
##   V =~
```

```
##      Informatn (a)      1.000                      2.587      0.860
##      Similarts (b)      0.984      0.046      21.625      0.000      2.545      0.841
##      Wrdr.Rsnng (c)      0.858      0.045      18.958      0.000      2.219      0.743
##      F =~
##      Mtrx.Rsnn (d)      1.000                      1.989      0.689
##      Pctr.Cncp (e)      0.825      0.085      9.747      0.000      1.642      0.552
##
## Regressions:
##      V ~
##      F      (k)      1.070      0.114      9.376      0.000      0.823      0.823
##
## Variances:
##      Information      2.352      0.253                      2.352      0.260
##      Similarities      2.685      0.261                      2.685      0.293
##      Word.Reasonng      4.000      0.295                      4.000      0.448
##      Matrix.Resnng      4.380      0.458                      4.380      0.525
##      Pictur.Cncpts      6.168      0.451                      6.168      0.696
##      V      2.164      0.485                      0.323      0.323
##      F      3.957      0.569                      1.000      1.000
```

3.5 Summary

In this chapter, I discussed basic latent variable models (LVM), and how they were part of a more general structural equation model (SEM) framework. I described model identification, including the influence of the number of indicators and different methods to scale the LV. In addition, I demonstrated how to fit a LVM in R using the `lavaan` package, using models with one and two LVs, as well as a SEM with latent exogenous and endogenous variables. I demonstrated how to obtain measures of model fit and model modification, although I did not go into great detail about how to interpret the values. In Appendix A, I discuss measures of model fit in more depth, while some of the references in Section 3.8 go into more detail about model modification. All the models in this chapter assumed that data was collected from a single group, which is not always the case. In the next chapter I discuss how to fit models with more than one group.

3.6 Writing the Results

Believe it or not, there is actually literature on writing LVM literature (e.g., Boomsma, Hoyle, & Panter, 2012; McDonald & Ho, 2002). While there are some differences in the authors' suggestions, there are also many areas of consensus. They all agree that a LVM should include:

1. A theoretical and empirical justification for the hypothesized model;
2. A complete description of how the LVMs were specified (i.e., the indicator variables for each LV, the scaling of the LVs, a description of what parameters were estimated and constrained);
3. A description of sample (i.e., demographic information, sample size, sampling method);
4. A description of the type of data used (e.g., nominal, continuous) and descriptive statistics;
5. Tests of assumptions (specifically that the indicator variables follow a multivariate normal distribution and estimator used (see Appendix A));
6. A description of missing data and how the missing data was handled (see Chapter 7);

7. The software and version used to fit the model;
8. Measures, and the criteria used, to judge model fit (see Appendix A); and
9. Any alterations made to the original model based on model fit or modification indices.

In addition, all reports should have a table with all parameter estimates (i.e., loadings, error variances, latent [co]variances), their standard errors, and standardized versions for the final model as well as any other significant model if more than one model was fit to the data. In Section 2.6, I discussed the `xtable()` function, which works on `lavaan` output from the `parameterEstimates()` function, thus creating an easy way to export `lavaan` parameter results into a table. Likewise, to increase research replicability, the LVM manuscript should also include the sample covariance matrix(es) either in the body of the text or in an appendix. The `xtable()` function can help facilitate this process as well.

`xtable()`

While not mandatory, it is very useful to include a neatly drawn path model of the final LVM, as well as any other significant model if more than one model was fit to the data. In the diagram, place the standardized or unstandardized (or both) coefficients along a given path. If using unstandardized coefficients, or leaving out any part of the LVM for ease of reading (e.g., the error variance terms), indicate this in the figure's caption. Nicol and Pexman (2010) give some examples of publication-ready diagrams that include IVs.

3.7 Exercises

- 3.1 Umstattd-Meyer, Janke, and Beaujean (2013) measured poor psychosocial health as a single factor model using three item facets from a depression questionnaire and a measure of social activity. The covariance matrix is given in Table 3.4.
 - 3.1.a Enter the covariance matrix into **R**.
 - 3.1.b Fit the model using the marker variable, standardized latent variable, and effects-coding methods of identification. For the marker variable method, use *Depression 1* as the marker variable. The resulting χ^2 and *df* should be identical for all methods.
- 3.2 The model for Exercise 3.1 was part of a larger SEM, part of which is shown in Figure 3.11. The full covariance matrix is given in Table 3.5.
 - 3.2.a Enter the covariance matrix into **R**.
 - 3.2.b Fit the SEM model to the data. Are both psychosocial and physical health predictive of personal mobility?

Table 3.4 Covariances for Exercise 3.1 ($n = 6053$).

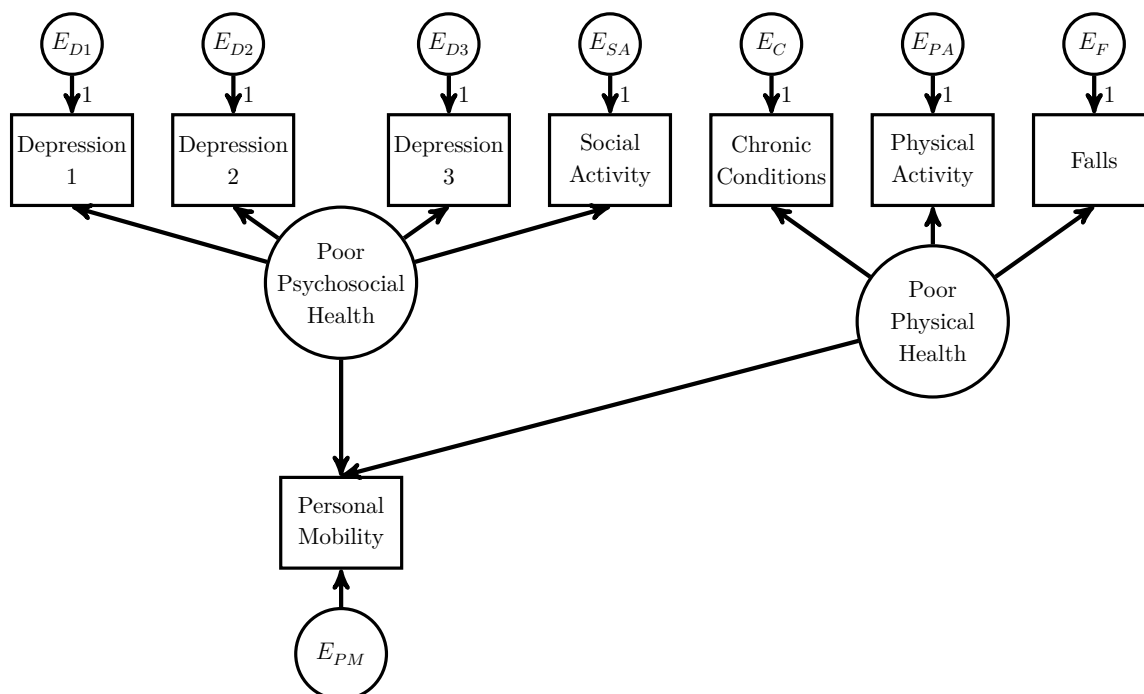
	D1	D2	D3	SA
Depression 1	0.77	0.38	0.39	−0.25
Depression 2	0.38	0.65	0.39	−0.32
Depression 3	0.39	0.39	0.62	−0.27
Social Activity	−0.25	−0.32	−0.27	6.09

Data taken from Umstattd-Meyer et al. (2013, pp. 4-5)

Table 3.5 Covariances for Exercise 3.2 ($n = 6053$).

	D1	D2	D3	SA	F	CC	PA	PM
Depression 1	0.77	0.38	0.39	-0.25	0.31	0.24	-3.16	-0.92
Depression 2	0.38	0.65	0.39	-0.32	0.29	0.25	-3.56	-0.88
Depression 3	0.39	0.39	0.62	-0.27	0.26	0.19	-2.63	-0.72
Social Activity	-0.25	-0.32	-0.27	6.09	-0.36	-0.18	6.09	0.88
Falls	0.31	0.29	0.26	-0.36	7.67	0.51	-3.12	-1.49
Chronic Conditions	0.24	0.25	0.19	-0.18	0.51	1.69	-4.58	-1.41
Physical Activity	-3.16	-3.56	-2.63	6.09	-3.12	-4.58	204.79	16.53
Personal Mobility	-0.92	-0.88	-0.72	0.88	-1.49	-1.41	16.53	7.24

Data taken from Umstattd-Meyer et al. (2013, pp. 4-5)

**Figure 3.11** Path model for Exercise 3.2. Figure taken from Umstattd-Meyer et al. (2013, p. 7) by permission of Oxford University Press and The Gerontological Society of America.

- 3.3 Graham (2008) showed how SEM can be used to analyze a variety of general linear models. His descriptive discriminant analysis (DDA) example uses formative latent variables. DDA is a procedure that describes the differences between multiple groups on multiple continuous descriptors—a multiple regression between two or more groups. In DDA, the latent variable of interest is called the *discriminant function*. A path model of the DDA is shown in Figure 3.12 and data for the model is given in Figure 3.6.
- 3.3.a Import the covariance matrix and mean vector shown in Figure 3.6 into **R**.
- 3.3.b Write **lavaan** syntax for Figure 3.12.
- 3.3.c Obtain the function coefficients (standardized coefficients for a & b), canonical correlation (standardized coefficient for c), and R^2 (standardized coefficient for d) values.

Table 3.6 Covariances for Exercise 3.3.

	DV ₁	DV ₂	Cat ₁
DV ₁	59.66	11.18	2.63
DV ₂	11.18	22.30	0.41
Cat ₁	2.63	0.41	1.00

$n = 288$. DV₁ and DV₂ are continuous variables and Cat₁ is a categorical variable with two groups coded as -1 and +1.

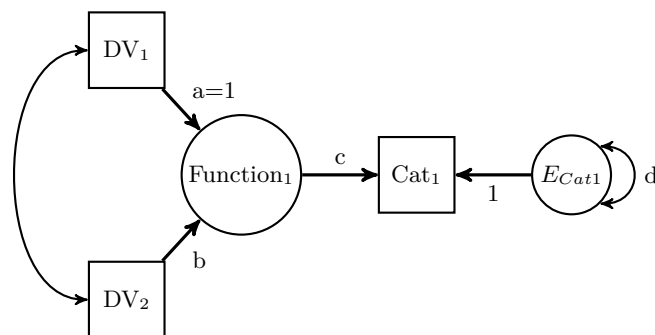


Figure 3.12 Path model of descriptive discriminant analysis.

3.8 References & Further Readings

- Beaujean, A. A. (2013). Factor analysis using **R**. *Practical Assessment, Research, and Evaluation*, 18(4), 1-11. Retrieved from <http://pareonline.net/pdf/v18n4.pdf>.
- Bollen, K. A. (2002). Latent variables in psychology and the social sciences. *Annual Review of Psychology*, 53, 605-634. doi: 10.1146/annurev.psych.53.100901.135239
- Boomsma, A., Hoyle, R. H., & Panter, A. T. (2012). The structural equation modeling research report. In R. H. Hoyle (Ed.), *Handbook of structural equation modeling*. New York, NY: Guilford.
- Brown, T. A. (2006). *Confirmatory factor analysis for applied research*. New York, NY: Guilford Press.
- Chou, C.-p., & Huh, J. (2012). Model modification in structural equation modeling. In R. A. Hoyle (Ed.), *Handbook of structural equation modeling* (pp. 232-246). New York, NY: Guilford.
- Edwards, J. R., & Bagozzi, R. P. (2000). On the nature and direction of relationships between constructs and measures. *Psychological Methods*, 5, 155-174. doi: 10.1037/1082-989X.5.2.155
- Graham, J. M. (2008). The general linear model as structural equation modeling. *Journal of Educational and Behavioral Statistics*, 33, 485-506. doi: 10.3102/1076998607306151
- Kenny, D. A., Kashy, D., & Bolger, N. (1988). Data analysis in social psychology. In D. Gilbert, S. Fiske, & G. Lindzey (Eds.), *Handbook of social psychology* (4th ed., pp. 233-265). New York, NY: McGraw-Hill.
- Little, T. D., Slegers, D. W., & Card, N. A. (2006). A non-arbitrary method of identifying and scaling latent variables in SEM and MACS models. *Structural Equation Modeling: A Multidisciplinary Journal*, 13, 59-72. doi: 10.1207/s15328007sem1301_3
- Loehlin, J. C. (2004). *Latent variable models: An introduction to factor, path, and structural equation analysis* (4th ed.). Mahwah, NJ: Erlbaum.
- McDonald, R. P., & Ho, M.-H. R. (2002). Principles and practice in reporting structural equation analyses. *Psychological Methods*, 7, 64-82. doi: 10.1037/1082-989X.7.1.64
- Nicol, A. A. M., & Pexman, P. M. (2010). *Displaying your findings: A practical guide for creating figures, posters, and presentations* (6th ed.). Washington, DC: American Psychological Association.
- Parkin, J., & Beaujean, A. A. (2012). The effects of Wechsler Intelligence Scale for Children-Fourth Edition cognitive abilities on math achievement. *Journal of School Psychology*, 50, 113-128. doi: 10.1016/j.jsp.2011.08.003
- Umstattd-Meyer, M. R., Janke, M. C., & Beaujean, A. A. (2013). Predictors of older adults' personal and community mobility: Using a comprehensive theoretical mobility framework. *The Gerontologist*, Advance online publication. doi: 10.1093/geront/gnt054
- Wechsler, D. (2003). *Wechsler intelligence scale for children* (4th ed.). San Antonio, TX: The Psychological Corporation.

4 | Latent Variable Models with Multiple Groups

Chapter Contents

4.1	Background	56
4.2	Invariance	56
4.2.1	Means and Intercepts	57
4.2.2	Types of Measurement Invariance	58
4.2.3	Evaluating Invariance	61
4.3	Group Equality Constraints	61
4.4	Example: Invariance	62
4.5	Using Labels for Parameter Constraints	70
4.6	Example: Genetically Informative Design	71
4.7	Summary	74
4.8	Writing the Results	75
4.9	Exercises	75
4.10	References & Further Readings	78

4.1 Background

There are typically two reasons to include multiple groups in a LVM model. The first reason is to examine if the LVM works the same across the groups. The second reason is when the estimation of certain parameters require data from two or more unique groups, as is done with genetically informative designs.

4.2 Invariance

The general question of invariance is one of whether or not, under different conditions of observing and studying phenomena, measurements yield measures of the same attributes. If there is no evidence indicating presence or absence of measurement invariance-the usual case-or there is evidence that such invariance does not obtain, then the basis for drawing scientific inference is severely lacking: *findings of differences between individuals and groups cannot be unambiguously interpreted.* (Horn & McArdle, 1992, p. 117, emphasis added)

A question that often arises when examining group differences on a variable is if any observed differences are due to differences in the groups themselves or differences in the instrument(s) used to measure the variable. One of the more useful ways of answering this question is by assessing invariance in LVs. To do so, however, requires I adjust the LVM I introduced in Chapter 3 to include both covariances and means, which is why these models are sometimes referred to as **mean and covariance structure** (MACS) models. A MACS path model is shown in Figure 4.1.

*Mean and
Covariance
Structure
(MACS)*

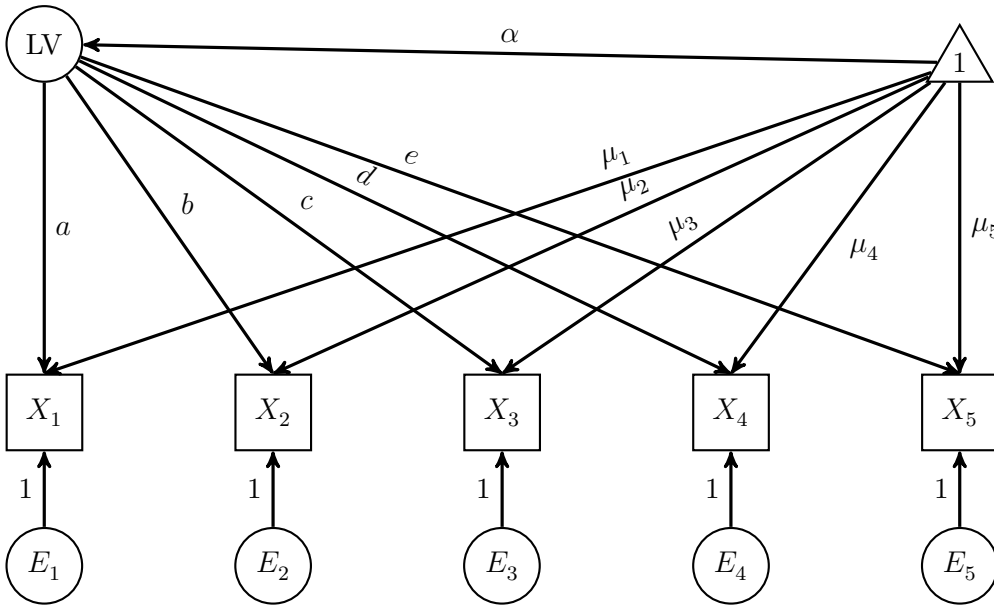


Figure 4.1 Example of a mean and covariance structure (MACS) path model. The indicator variables' intercepts are labeled with a μ and the latent variable's mean is α .

4.2.1 Means and Intercepts

When the means of the LV are included in a LVM, the intercepts of the MVs need to be included as well. Means and intercepts are measures of a variable's *location*.¹ Specifically, in a LVM the mean of an indicator variable is decomposed into two parts: (a) a common factor component (*mean*), and (b) a unique component (*intercept*). This is done in a path model by adding a constant term of value 1.0 (symbolized by a triangle) and drawing directional paths from it to all the latent and manifest variables. Thus, the tracing rules discussed in Section 2.1.3 need to be expanded to account for these additions. The additional rules are given in Figure 4.2.

- The mean of an indicator variable is found by tracing all paths from it to the constant, and then adding all values. The paths:
 - can only go through directional arrows, and
 - can only go backwards along an arrow.

Figure 4.2 Additional tracing rules when including means in the model. For the other tracing rules, see Figure 2.5.

¹When a variable is exogenous, its location is usually referred to as the *mean*. When a variable is endogenous, its location is usually referred to as an *intercept*. The difference is that the mean is unconditional, while the intercept is conditional. For simplicity, I use *means* for all the LV location parameters throughout this book.

As an example, in Figure 4.1 the path for the mean of the manifest variable X_1 , \bar{x}_1 , is

$$a \times \alpha + \mu_1 = \bar{x}$$

The $a \times \alpha$ path is the common factor component (i.e., mean) and μ_1 path is the unique component (i.e., intercept).

4.2.1.1 Identification with Means and Intercepts in the Model

When means and intercepts are included in a path model, each MV's observed mean needs to be included in the input. The formula to calculate the number of non-redundant pieces of information in a dataset with means is given in Equation (4.1).

$$\text{non-redundant information in a dataset with means} = \frac{p(p+1)}{2} + m = \frac{p(p+3)}{2} \quad (4.1)$$

where p is the number of manifest variables.

For MACS models to be identified, all the free location (means/intercepts) and scale (variances) parameters have to be estimated uniquely. Thus, I need to add to the identification rules-of-thumb I presented in Section 3.2.1 to account for means and intercepts. There are three ways to do this that map onto one of the three methods of LV scaling I presented in Section 3.2.1.2.

1. *Standardized latent variable.* For this method, in addition to fixing the latent variance to 1.0 (to fix the LV's scales), it also requires fixing the latent means to 0.0 (to fix the LV's location). This allows the estimation of all the indicator variables' intercepts, although they will be equal to the indicator variables' means.
2. *Marker variable.* For this method, the LV's scale is set by fixing one indicator variable's loading to 1.0 for each LV. To set the LV's mean, the same indicator variable's intercepts is fixed to 0.0. This sets the LVs to have scales and locations directly related to the marker variable.
3. *Effects-coding.* To set a LV's scale with this method, the loadings of its indicator variables need to be constrained to sum to the number of its indicators. To set the LV's location, constrain the intercepts of a LV's indicator variables to sum to 0.0. This method forces the mean and variance of the LV to be the weighted average of all of its indicator variables' means and loadings.

4.2.2 Types of Measurement Invariance

Measurement invariance is not an all-or-nothing concept. Instead, there are degrees of invariance, usually described as a hierarchical sequence starting with the weakest form and working up to the strictest form. I describe the different levels of invariance below as well as present them in Table 4.1.

*Measurement
Invariance*

*Configural
Invariance*

1. **Configural.** This is the most basic level, and just indicates that LVMs have the same structure in all the groups. That is, the groups have the same number of LVs, formed by the same number of indicator variables, all of which have the same pattern of constrained and estimated parameters. There is no requirement that any estimated coeffi-

cients are equal across groups, but the price for these weak assumptions is that there is no reason to believe that the LVs are measuring the same construct in each group. Thus, this level of invariance does not warrant any between-group comparisons of the constructs the LVs represent.

2. **Weak.** This level adds that, for a given indicator, the loadings are the same across groups. Latent variances are allowed to vary among groups, though.² With weak invariance, there is a weak argument that the LVs are equivalent between groups, but, at most, this condition only allows a comparison of the latent variances and covariances. *Weak Invariance*
3. **Strong.** This level of invariance adds the constraint that, for a given indicator variable, the intercepts are the same among groups. When constraining the intercepts, the latent means are allowed to vary among groups. With this level of invariance, individuals at the same level of a given LV have the same expected value on the indicator variables, irrespective of group membership. Moreover, strong invariance allows for comparisons of the LV. Specifically, latent means, variances, and covariances can all be compared among groups. If there are group differences on strongly-invariant LVs, it likely indicates that there are real difference in these variables, as opposed to the difference being in how the LVs are measured. *Strong Invariance*
4. **Strict.** This level adds that, for a given indicator, the error variances are constrained to be equal across groups. Usually this level of invariance is not required for making cross-group comparisons of the LVs. Because the error terms in a LVM are comprised of random error variance as well as indicator-specific variance, there is not necessarily the expectation that they would be the same among the groups. Thus, this form of invariance is usually tested, in conjunction with evaluating invariance of the latent variances (level 5), when the area of interest is the reliability of the construct the LV represents. If the latent variances are invariant among the groups, then evaluating strict invariance is really an investigation of construct reliability invariance. *Strict Invariance*
5. **Homogeneity of Latent Variable Variances.** The variance of a LV represents its dispersion, and thus represents how much variability there is in the construct that the LV represents varies within groups. If there is homogeneity of the latent variances, this indicates the groups used equivalent ranges of the construct for the indicator variables' values. If this does not hold, however, this indicates that the group with the smaller amount of latent variance is using a narrower range of the construct than the group with the larger amount. Evaluating homogeneity of latent *covarinaces* can be done, too, but there is usually not much to be gained from such an analysis.
6. **Homogeneity of Factor Means.** This level of invariance indicates no differences among groups in the average level of the construct the LV represents. Like the more traditional analyses of mean differences (e.g., ANOVA), if there is more than one group then the test of a latent variable's mean differences usually begins with an omnibus test that constrains the means to be equal across all groups. If this model does not fit the data well, then subsequent tests may be conducted to isolate specific group differences.

²Since the latent variances are constrained to be the same at this level, weak invariance really just shows that the loadings are only *proportionally* equivalent. This is akin to predicting weight from height in two groups. In the first group, height is measured in feet, but in the second group it is measured in inches.

Table 4.1 Types of Measurement Invariance.

	Type of Invariance	Constraints	Between-Groups Comparisons Allowed
1	Configural	Same model. No parameter constraints.	None
2	Weak	1 + all loadings constrained to be equal between groups (but can vary within a group). Latent (co)variances allowed to vary between groups.	Latent (co)variances [weak evidence]
3	Strong	2 + all intercepts are constrained to be equal between groups (but can vary within a group). Latent means allowed to vary between groups.	Latent means, latent (co)variances [strong evidence]
4	Strict	3 + error variances are constrained to be the same between groups (but can vary within a group).	

Models with larger numbers are nested within the models with smaller numbers.

Structural Invariance

According to Vandenberg and Lance (2000), technically only the first four of the steps deal with aspects of measurement invariance because they are concerned with the relationships between indicator variables and LVs between groups. The last two steps deal with aspects of **structural invariance** because they are concerned with properties of the LVs themselves.

4.2.2.1 Partial Invariance

Partial Invariance

When a given level of invariance is untenable for all the indicator variables (usually levels 2-4), a follow-up analysis is usually done to determine what specific indicator(s) contribute to the misfit. **Partial invariance** is when an invariance constraint is not warranted for only a few of the parameter estimates within an invariance level, but is warranted for all the others. If partial invariance exists at a given level for a model, there are a variety of ways to proceed:

1. Leave the non-invariant indicator variables in the model, but not constrain them to be invariant across groups, arguing that the invariant indicators are sufficient to establish comparability of the constructs.
2. Argue that the differences between indicator variables are small enough that they would not make a substantive difference and proceed with invariance constraints in place.
3. Remove the indicator variables that are not fully invariant, and then re-run the invariance assessment.
4. Conclude that because there is not full invariance, the indicator variables must be measuring different constructs across the groups and, therefore, not use the indicators.

Usually, a mix of options 1 and 2 are used in practice, although 3 is useful in some circumstances. Option 4 becomes the best alternative when the number of indicator variables not exhibiting invariance is too high.

4.2.3 Evaluating Invariance

Because invariance involves a hierarchy of models, testing for invariance requires a multi-step procedure, testing more restrictive models at each step.³ As with assessing the fit of a model for a single group, there are multiple ways to examine if the increasingly restrictive models in Section 4.2.2 are consistent with the data. One approach is the *statistical approach*, which examines the change in χ^2 values ($\Delta\chi^2$) across nested models (see Appendix A for more on nested models).⁴ If, as the models grow more restrictive, the $\Delta\chi^2$ values do not significantly change (i.e., the p -value for $\Delta\chi^2$ is $>$ some pre-set type 1 error level), this is taken as evidence that the more restrictive model (i.e., higher-level of invariance) fits data as well as the less restrictive model; thus, the more restrictive model should be favored over the less restrictive one since it is more parsimonious (i.e., estimates fewer parameters).

An alternative approach to evaluating invariance is to use a *modeling approach*. As with the statistical approach, increasingly-constrained models are evaluated against each other, only this approach uses approximate fit indices (AFI) to determine what model to use (see Appendix A for types of AFIs). For this approach, if a model with a given set of invariance constraints has an adequate level of fit on the AFIs, then the invariance constraints are reasonable additions to the LVM. The question you are probably asking is, *What fit indices do I use for the modeling approach to evaluate invariance?* Meade, Johnson, and Braddy (2008) did a Monte Carlo study (see Chapter 8 for more on Monte Carlo studies) and suggested the comparative fit index (CFI) and McDonald's noncentrality fit index (MFI) were relatively robust. Thus, the modeling approach would recommend that invariance be based on two criteria: (a) the multi-group factor model with constraints exhibits an adequate fit to the data according to the AFIs; and (b) the change in the CFI and MFI values from one model to a more restrictive (nested) model is negligible.

See Meade et al's (2008) article for more information on negligible value changes.

4.3 Group Equality Constraints

Before working through an invariance example in **R**, I briefly discuss general parameter constraints in LVMs. For some LVMs, there is a need to constrain a parameter estimate to be equal to a specific value, to be greater/less than a specific value, or to be equal to another parameter estimate (which includes the same parameter estimate across groups). There are two ways to constrain parameters in **lavaan** (and almost any other LV program). The first way to constrain parameters is by using parameter labels, which I discuss in Section 4.5. The second way is to use *group equality constraints*, which impose equality constraints on a whole set of parameter estimates (e.g., all intercepts) among groups using the `group.equal` argument in the `cfa()` or `sem()` functions. These take the general form of `group.equal="m"`, where `m` is one

³The alternative would be to conduct omnibus tests for equality of the indicators' covariances and means among the groups as the initial step. If that model fit the data well, then there would not be a need for any further invariance investigation.

⁴The df for $\Delta\chi^2$ are the difference between the nested models' df .

Table 4.2 Keywords for the `group.equal` Argument to Specify Equality Constraints Among Groups.

Keyword	Constraints
<code>intercepts</code>	Intercepts of the manifest variables
<code>means</code>	Means/Intercepts of the latent variables
<code>residuals</code>	Residual variances of the manifest variables
<code>residual.covariances</code>	Residual covariances of the manifest variables
<code>lv.variances</code>	(Residual) variances of the latent variables
<code>lv.covariances</code>	(Residual) covariances of the latent variables
<code>regressions</code>	All regression coefficients

If the analysis uses data from multiple groups, but there is no `group.equal` argument, the same model is used for both groups, but all parameters are freely estimated in each group (i.e., configural invariance).

or more of the keywords given in Table 4.2. If one or more of the constrained values needs to be released, use the `group.partial` argument with the name of the parameter(s) to be freed. Section 4.4 contains an example using both of these arguments.

4.4 Example: Invariance

This example comes from Beaujean, Freeman, Youngstrom, and Carlson (2012). They examined if the structure of the Wechsler intelligence Scale for Children-Third Edition (WISC-III; Wechsler, 1991) was the same in children with and without manic symptoms. They used eight subtests from the WISC-III, and their model had two LVs (see Figure 4.3), each of which had four indicator variables. The LVs were allowed to correlate with each other. The data (covariances and means) from the two groups are shown in Table 4.3.

Since the purpose of this analysis is to compare the groups, I enter the covariances into separate **R** objects.⁵

```
# variable names
wisc3.names <- c("Info", "Sim", "Vocab", "Comp", "PicComp", "PicArr", "BlkDsgn", "ObjAsmb")
# group with manic symptoms
## covariances
manic.cov <- c(9.364, 7.777, 12.461, 6.422, 8.756, 10.112, 5.669, 7.445, 6.797, 8.123, 3.048,
4.922, 4.513, 4.116, 6.200, 3.505, 4.880, 4.899, 5.178, 5.114, 15.603, 3.690, 5.440, 5.220,
3.151, 3.587, 6.219, 11.223, 3.640, 4.641, 4.877, 3.568, 3.819, 5.811, 6.501, 9.797)
manic.cov <- lower2full(manic.cov)
## means
manic.means <- c(10.09, 12.07, 10.25, 9.96, 10.90, 11.24, 10.30, 10.44)
## label the covariances and means
colnames(manic.cov) <- rownames(manic.cov) <- wisc3.names
```

⁵If I used raw data instead of summary statistics for the input, I would need to have a variable in the dataset that contained group membership. Then, I would use the `group="group.name"` argument to refer to the name of the group membership variable, e.g.,

```
cfa(model, data = my.data, group = "group.name")
```

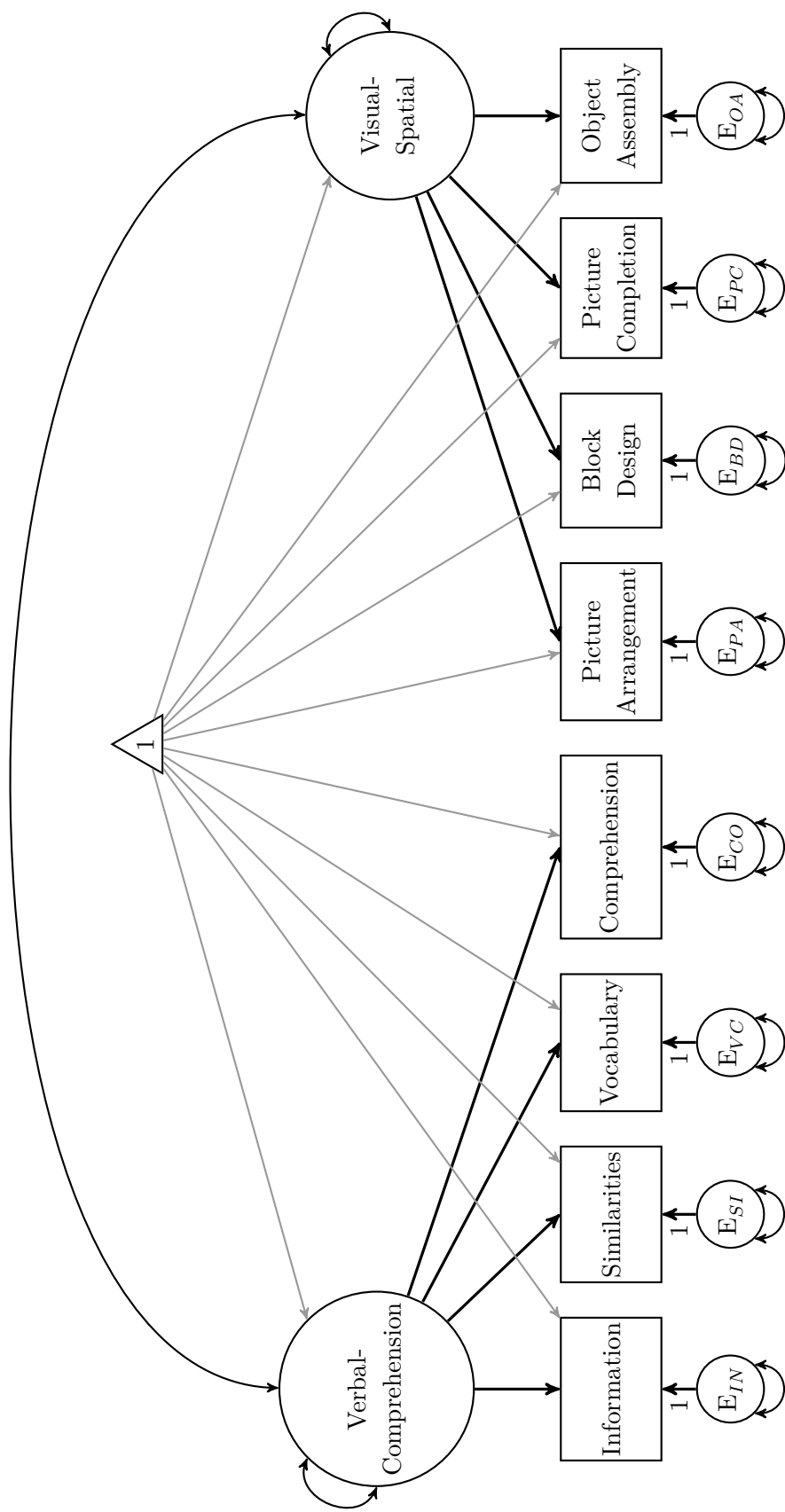


Figure 4.3 Latent variable model of the Wechsler Intelligence Scale for Children-Third Edition subtests based on Beaujean et al. (2012).

```
names(manic.means) <- wisc3.names

# norming group
## covariances
norming.cov <- c(9.610, 5.844, 8.410, 6.324, 6.264, 9.000, 4.405, 4.457, 5.046, 8.410, 4.464,
4.547, 4.512, 3.712, 10.240, 3.478, 2.967, 2.970, 2.871, 3.802, 10.890, 5.270, 4.930,
4.080, 3.254, 5.222, 3.590, 11.560, 4.297, 4.594, 4.356, 3.158, 4.963, 3.594, 6.620, 10.890)
norming.cov <- lower2full(norming.cov)
## means
norming.means <- c(10.10, 10.30, 9.80, 10.10, 10.10, 10.10, 9.90, 10.20)
## label the covariances and means
colnames(norming.cov) <- rownames(norming.cov) <- wisc3.names
names(norming.means) <- wisc3.names
```

Beaujean et al. (2012) examined model fit using the χ^2 , CFI, MFI, root mean square error of approximation (RMSEA), and standardized root mean square residual (SRMR) (see Appendix A for more information on these fit measures). Since I use the same fit indices to evaluate every model, I saved their names in an **R** object so I do not have to re-type them every time I use the `fitMeasures()` function.

```
# fit indices
fit.indices <- c("chisq", "df", "cfi", "rmsea", "srmr", "mfi")
```

Before examining invariance, I fit the path model in Figure 4.3 for each group, separately. To tell **lavaan** to use the means in an analysis with a single group, input a vector of means using the `sample.mean` argument and set the `meanstructure` argument to `TRUE`.

```
# LVM model for WISC-III data
wisc3.model<-'
VC =~ Info + Sim + Vocab + Comp
VS =~ PicComp + PicArr + BlkDsgn + ObjAsmb
VC~~VS
'

# group with manic symptoms
manic.fit <- cfa(wisc3.model, sample.cov=manic.cov, sample.nobs=81,
sample.mean=manic.means, meanstructure=TRUE)
fitMeasures(manic.fit, fit.indices)

##  chisq    df    cfi  rmsea  srmr   mfi
## 29.188 19.000 0.971 0.081 0.047 0.939

# norming group
norming.fit <- cfa(wisc3.model, sample.cov=norming.cov, sample.nobs=200,
sample.mean=norming.means, , meanstructure=TRUE)
fitMeasures(norming.fit, fit.indices)

##  chisq    df    cfi  rmsea  srmr   mfi
## 24.211 19.000 0.992 0.037 0.029 0.987
```

To fit the factor model for both groups simultaneously using the summary statistics as input, I must combine both covariance matrices into a single *list*; likewise for the means and sample sizes. In all the lists, the same name needs to be used for each group. In **R**, a `list()` is an object that contains other **R** objects. Unlike a `data.frame()`, `list()` objects do not have to be the same dimensions or even the same type of object. To reference a list object directly,

```
list()
[[1]]
```

Table 4.3 Covariances and Means for Wechsler Intelligence Scale for Children-Third Edition Subtests.

	Info	Sim	Vocab	Comp	PicComp	PicArr	BlkDsgn	ObjAsmb
Information	9.364	7.777	6.422	5.669	3.048	3.505	3.690	3.640
Similarities	7.777	12.461	8.756	7.445	4.922	4.880	5.440	4.641
Vocabulary	6.422	8.756	10.112	6.797	4.513	4.899	5.220	4.877
Comprehension	5.669	7.445	6.797	8.123	4.116	5.178	3.151	3.568
Picture Completion	3.048	4.922	4.513	4.116	6.200	5.114	3.587	3.819
Picture Arrangement	3.505	4.880	4.899	5.178	5.114	15.603	6.219	5.811
Block Design	3.690	5.440	5.220	3.151	3.587	6.219	11.223	6.501
Object Assembly	3.640	4.641	4.877	3.568	3.819	5.811	6.501	9.797
Subtest Mean	10.090	12.070	10.250	9.960	10.900	11.240	10.300	10.440

(a) Youth with Manic Symptoms ($n = 81$). Data taken from Beaujean et al. (2012, p. 5).

	Info	Sim	Vocab	Comp	PicComp	PicArr	BlkDsgn	ObjAsmb
Information	9.610	5.844	6.324	4.405	4.464	3.478	5.270	4.297
Similarities	5.844	8.410	6.264	4.457	4.547	2.967	4.930	4.594
Vocabulary	6.324	6.264	9.000	5.046	4.512	2.970	4.080	4.356
Comprehension	4.405	4.457	5.046	8.410	3.712	2.871	3.254	3.158
Picture Completion	4.464	4.547	4.512	3.712	10.240	3.802	5.222	4.963
Picture Arrangement	3.478	2.967	2.970	2.871	3.802	10.890	3.590	3.594
Block Design	5.270	4.930	4.080	3.254	5.222	3.590	11.560	6.620
Object Assembly	4.297	4.594	4.356	3.158	4.963	3.594	6.620	10.890
Subtest Mean	10.100	10.300	9.800	10.100	10.100	10.100	9.900	10.200

(b) WISC-III Norming Sample ($n = 200$). Data taken from Wechsler (1991).

use a double square bracket `[[]]` operator. If I name the objects when creating the list, then the `$` operator can be used to reference the list's object as well. Working with lists is made easier with an example, so below I show how I combined the two groups' covariances matrices into a single list object.

```
# combine the two covariance matrices into a single list object
combined.cov <- list(manic = manic.cov, norming = norming.cov)
# access the covariance matrix for group with manic symptoms
combined.cov[[1]]
```

```
##          Info  Sim Vocab Comp PicComp PicArr BlkDsgn ObjAsmb
## Info      9.4  7.8   6.4  5.7    3.0    3.5    3.7    3.6
## Sim       7.8 12.5   8.8  7.4    4.9    4.9    5.4    4.6
## Vocab     6.4  8.8  10.1  6.8    4.5    4.9    5.2    4.9
## Comp      5.7  7.4   6.8  8.1    4.1    5.2    3.2    3.6
## PicComp   3.0  4.9   4.5  4.1    6.2    5.1    3.6    3.8
## PicArr    3.5  4.9   4.9  5.2    5.1   15.6    6.2    5.8
## BlkDsgn   3.7  5.4   5.2  3.2    3.6    6.2   11.2    6.5
## ObjAsmb   3.6  4.6   4.9  3.6    3.8    5.8    6.5    9.8
```

```
# access the norming group's covariance matrix
combined.cov$norming
```

```
##          Info Sim Vocab Comp PicComp PicArr BlkDsgn ObjAsmb
## Info      9.6 5.8   6.3  4.4    4.5    3.5    5.3    4.3
## Sim       5.8 8.4   6.3  4.5    4.5    3.0    4.9    4.6
## Vocab     6.3 6.3   9.0  5.0    4.5    3.0    4.1    4.4
## Comp      4.4 4.5   5.0  8.4    3.7    2.9    3.3    3.2
## PicComp   4.5 4.5   4.5  3.7   10.2    3.8    5.2    5.0
## PicArr    3.5 3.0   3.0  2.9    3.8   10.9    3.6    3.6
## BlkDsgn   5.3 4.9   4.1  3.3    5.2    3.6   11.6    6.6
## ObjAsmb   4.3 4.6   4.4  3.2    5.0    3.6    6.6   10.9
```

```
# combine the sample sizes and means into lists
combined.n <- list(manic = 81, norming = 200)
combined.means <- list(manic = manic.means, norming = norming.means)
```

Fitting the configural invariance model is very similar to fitting a single-group LVM in *lavaan*. The only difference is that for the configural invariance model the specified covariance, n , and mean objects are lists.

```
# configural invariance
configural.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
  sample.mean=combined.means)
fitMeasures(configural.fit, fit.indices)
```

```
##  chisq      df      cfi  rmsea  srmr   mfi
## 53.399 38.000  0.985  0.054  0.038  0.973
```

To fit the weak invariance model, I constrain all the loadings between groups to be the same. I do this using the `group.equal` argument.

```
# weak invariance
weak.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings"))
fitMeasures(weak.fit, fit.indices)

##   chisq    df    cfi  rmsea  srmr   mfi
## 66.012 44.000 0.979 0.060 0.061 0.962
```

Next, I fit a strong invariance model by constraining the indicator variables' intercepts to be the same between the groups. In *lavaan*, doing this automatically sets the means of the latent variables in the first group to 0.0 and free the means in all the other groups. This is shown in the *Intercepts:* section of the output (not shown here for space considerations).

```
# strong invariance
strong.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts"))
fitMeasures(strong.fit, fit.indices)

##   chisq    df    cfi  rmsea  srmr   mfi
## 109.107 50.000 0.942 0.092 0.064 0.900
```

Beaujean et al. (2012) reported that this model did not fit the data well. Consequently, they examined the modification indices to see what constraints to release to make the model fit the data better. This is done in *lavaan* using the `modindices()` function (see Section 3.3 for more information on this function's output). Instead of examining the modification indices for *all* the parameter estimates, I only examine those involving the intercepts. I do this by saving the `modindices()` output into an **R** object, selecting the modification indices (column *mi*) associated with ~1 in the operation (*op*) column, and then sorting those values from largest to smallest using the `order()` function with the `decreasing=TRUE` argument.

```
modindices()
order()
```

```
# extract modification indices
strong.mi <- modindices(strong.fit)
# select only values concerning an intercept or mean
strong.mi <- strong.mi[strong.mi$op == "~1", ]
# sort the results from largest to smallest
strong.mi[order(strong.mi$mi, decreasing = TRUE), ]

##      lhs op rhs group    mi    epc sepc.lv sepc.all sepc.nox
## 1   Sim ~1      2 28.17 0.76 0.76 0.214 0.214
## 2   Sim ~1      2 28.17 -0.76 -0.76 -0.254 -0.254
## 3  Comp ~1      2 7.17 0.34 0.34 0.116 0.116
## 4  Comp ~1      1 7.17 -0.34 -0.34 -0.125 -0.125

..<Output Omitted>..

## 19   VC ~1      2 0.00 0.00 0.00 0.000 0.000
## 20   VS ~1      2 0.00 0.00 0.00 0.000 0.000
```

It appears that the *Similarities* subtest is one place where a problem lies, so I remove the invariance constraint from its intercept using the `group.partial` argument.


```
# strong invariance 2: allow intercepts for Similarities subtest to be freely estimated
strong.fit2 <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts"),
group.partial=c("Sim~1"))
fitMeasures(strong.fit2, fit.indices)
```

```
## chisq    df    cfi rmsea  srmr   mfi
## 75.943 49.000 0.974 0.063 0.058 0.953
```

This model appears to fit the data fairly well, so, next, I fit a partial strict invariance model. Since I had to release the invariance constraint for the Similarities subtest's intercept, I also release the constraint for its error variance.

```
# partial strict invariance
strict.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts", "residuals"),
group.partial=c("Sim~1", "Sim~~Sim"))
fitMeasures(strict.fit, fit.indices)
```

```
## chisq    df    cfi rmsea  srmr   mfi
## 94.71 56.00 0.96 0.07 0.07 0.93
```

Beaujean et al. (2012) reported that there was a problem with this model, too. To address this, they released the invariance constraint for the Picture Completion, Comprehension, and Picture Arrangement subtests' error variance; I do the same.

```
# partial strict invariance 2: allow residual variances for Picture Completion, Comprehension, and
# Picture Arrangement subtests to be freely estimated
strict.fit2 <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts", "residuals"),
group.partial=c("Sim~1", "Sim~~Sim", "PicComp~~PicComp", "Comp~~Comp", "PicArr~~PicArr"))
fitMeasures(strict.fit2, fit.indices)
```

```
## chisq    df    cfi rmsea  srmr   mfi
## 76.619 53.000 0.977 0.056 0.058 0.959
```

This model appears to fit the data relatively well. For completeness, I finish the invariance assessment by fitting models constraining the latent variances, covariances, and means to be equal.

```
# latent variances
factor.var.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts", "residuals",
"lv.variances"), group.partial=c("Sim~1", "Sim~~Sim", "PicComp~~PicComp",
"Comp~~Comp", "PicArr~~PicArr"))
fitMeasures(factor.var.fit, fit.indices)
```

```
## chisq    df    cfi rmsea  srmr   mfi
## 79.528 55.000 0.976 0.056 0.070 0.957
```

```
# latent covariances
factor.covar.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts", "residuals",
"lv.variances", "lv.covariances"), group.partial=c("Sim~1", "Sim~~Sim",
"PicComp~~PicComp", "Comp~~Comp", "PicArr~~PicArr"))
fitMeasures(factor.covar.fit, fit.indices)
```

```
## chisq      df      cfi  rmsea  srmr    mfi
## 79.850 56.000  0.977  0.055  0.069  0.958

# latent means
factor.means.fit <- cfa(wisc3.model, sample.cov=combined.cov, sample.nobs=combined.n,
sample.mean=combined.means, group.equal=c("loadings", "intercepts", "residuals",
"lv.variances", "lv.covariances", "means"), group.partial=c("Sim~1", "Sim~~Sim",
"PicComp~~PicComp", "Comp~~Comp", "PicArr~~PicArr"))
fitMeasures(factor.means.fit, fit.indices)

## chisq      df      cfi  rmsea  srmr    mfi
## 83.439 58.000  0.975  0.056  0.073  0.956
```

Below I give the lavaan summary output for the final model with all its constraints. The constrained parameters have the same *unstandardized* coefficient values across groups.

```
summary(factor.means.fit, standardized = TRUE)

## lavaan (0.5-16) converged normally after 67 iterations

..<Output Omitted>..

##
## Group 1 [manic]:
##
##              Estimate Std.err Z-value P(>|z|) Std.lv Std.all
## Latent variables:
## VC =~
##   Info          1.000          2.393  0.778
##   Sim           1.099  0.072  15.185  0.000  2.631  0.841
##   Vocab         1.100  0.072  15.318  0.000  2.633  0.864
##   Comp          0.865  0.069  12.522  0.000  2.069  0.795
## VS =~
##   PicComp        1.000          2.030  0.766
##   PicArr         0.906  0.118  7.669  0.000  1.839  0.496
##   BlkDsgn        1.213  0.120  10.134  0.000  2.463  0.729
##   ObjAsmb        1.171  0.115  10.173  0.000  2.377  0.733
##
## Covariances:
## VC ~~
##   VS            3.684  0.503  7.327  0.000  0.758  0.758
##
## Intercepts:
##   Info          10.097  0.184  54.997  0.000  10.097  3.281
##   Sim           11.903  0.261  45.649  0.000  11.903  3.804
##   Vocab          9.930  0.182  54.599  0.000  9.930  3.257
##   Comp          10.010  0.170  58.775  0.000  10.010  3.846
##   PicComp       10.396  0.175  59.301  0.000  10.396  3.923
##   PicArr        10.394  0.208  49.872  0.000  10.394  2.805
##   BlkDsgn       10.015  0.202  49.692  0.000  10.015  2.964
##   ObjAsmb       10.269  0.193  53.091  0.000  10.269  3.167
##   VC            0.000          0.000  0.000
##   VS            0.000          0.000  0.000
##
## Variances:
##   Info          3.743  0.382          3.743  0.395
##   Sim           2.868  0.595          2.868  0.293
##   Vocab         2.360  0.301          2.360  0.254
```

```

##      Comp      2.493    0.475      2.493    0.368
##      PicComp    2.903    0.613      2.903    0.413
##      PicArr    10.351    1.724     10.351    0.754
##      BlkDsgn    5.348    0.604      5.348    0.469
##      ObjAsmb    4.864    0.554      4.864    0.463
##      VC         5.729    0.765      1.000    1.000
##      VS         4.120    0.685      1.000    1.000
##
##
##
## Group 2 [norming]:
##
##      Estimate Std.err Z-value P(>|z|) Std.lv Std.all
## Latent variables:
## VC =~
##   Info      1.000
##   Sim       1.099    0.072   15.185    0.000    2.631    0.863
##   Vocab     1.100    0.072   15.318    0.000    2.633    0.864
##   Comp      0.865    0.069   12.522    0.000    2.069    0.685
## VS =~
##   PicComp    1.000
##   PicArr     0.906    0.118    7.669    0.000    1.839    0.538
##   BlkDsgn    1.213    0.120   10.134    0.000    2.463    0.729
##   ObjAsmb    1.171    0.115   10.173    0.000    2.377    0.733
##
## Covariances:
## VC ~~
##   VS         3.684    0.503    7.327    0.000    0.758    0.758
##
## Intercepts:
##   Info      10.097    0.184   54.997    0.000   10.097    3.281
##   Sim       10.368    0.194   53.320    0.000   10.368    3.400
##   Vocab     9.930    0.182   54.599    0.000    9.930    3.257
##   Comp      10.010    0.170   58.775    0.000   10.010    3.315
##   PicComp   10.396    0.175   59.301    0.000   10.396    3.328
##   PicArr    10.394    0.208   49.872    0.000   10.394    3.038
##   BlkDsgn   10.015    0.202   49.692    0.000   10.015    2.964
##   ObjAsmb   10.269    0.193   53.091    0.000   10.269    3.167
##   VC         0.000
##   VS         0.000
##
## Variances:
##   Info      3.743    0.382      3.743    0.395
##   Sim       2.374    0.353      2.374    0.255
##   Vocab     2.360    0.301      2.360    0.254
##   Comp      4.839    0.535      4.839    0.531
##   PicComp   5.640    0.667      5.640    0.578
##   PicArr    8.320    0.912      8.320    0.711
##   BlkDsgn   5.348    0.604      5.348    0.469
##   ObjAsmb   4.864    0.554      4.864    0.463
##   VC         5.729    0.765      1.000    1.000
##   VS         4.120    0.685      1.000    1.000

```

4.5 Using Labels for Parameter Constraints

In Section 4.3, I discussed how to constrain parameters using `lavaan`'s `group.equal` argument. The alternative way to constrain parameters is to use variable labels. This can take one of three forms. First, to constrain a parameter to be a specific value (i.e., fix the parameter

estimate), premultiply the parameter by the desired value. For example, to constrain the loading of *V2*:

```
F1 =~ V1 + 3*V2 + V3
```

Second, to constrain a parameter to be greater/less than a value, premultiply the label by the parameter, then in a separate line give the inequality. For example, to constrain the variance of *V1* to be greater than 0:

```
V1 ~~ a*V1
a > 0
```

Third, to estimate a parameter, but constrain it to be the same value among multiple groups, use the same label in all the groups. To do this, use the concatenate function, *c()*, to apply multiple labels.⁶ For example, to constrain the loading of *V2* to be the same value between two groups:

```
F1 =~ V1 + c(a,a)*V2 + V3
```

4.6 Example: Genetically Informative Design

In Section 4.1, I stated there were two purposes for including multiple groups in a LVM. The first was to examine invariance. The second purpose is when the estimation of a parameter requires data from two or more unique groups. This is commonly done with genetically informative designs, as is common in the field of **behavior genetics**. Behavior genetic (BG) models examine genetic and environmental influences on physical or behavioral traits (i.e., a phenotype). The research designs in this field use natural experiments to study these influences, such as twins reared in different environments and children in adopted families.

Behavior Genetics

In BG models, the genetic influence on a phenotype is decomposed into: (a) additive effects of alleles at various loci (*A*) (this is sometimes called narrow heritability); and (b) non-additive effects (*D*), such as allele dominance. The combination of *A* and *D* is called the broad heritability. Likewise, environmental influence on the phenotype is decomposed into: (a) effects due to a *shared* environment (*C*) (i.e., between-family effects), such as being raised by the same parents in the same house; and (b) effects due to a *non-shared* environment (*E*) (i.e., within-family effects), such as having different peers or attending different schools. These non-shared effects include random environmental events (e.g., getting into automobile accident) as well as random measurement events (i.e., measurement error). For readers interested in such models, Plomin, DeFries, McClearn, and McGuffin (2008) provide an excellent introductory text.

Figure 4.4 contains a path model for a BG analysis of a single phenotype, *P*, measured in both identical/monozygotic (MZ) and fraternal/dizygotic (DZ) twins raised together. The model has seven latent variables, but there are only two indicator variables: the phenotype measured in twin 1 (*P*₁) and in twin 2 (*P*₂). This is obviously an underidentified model, so I need to place constraints on the parameter estimates to obtain unique estimates. For the LVs

⁶A second way to constrain parameters between groups within the model is to specify *m* separate models for your *m* groups, and give the parameters to be constrained the same label. For an example, see my answer to Exercise 4.1.b.

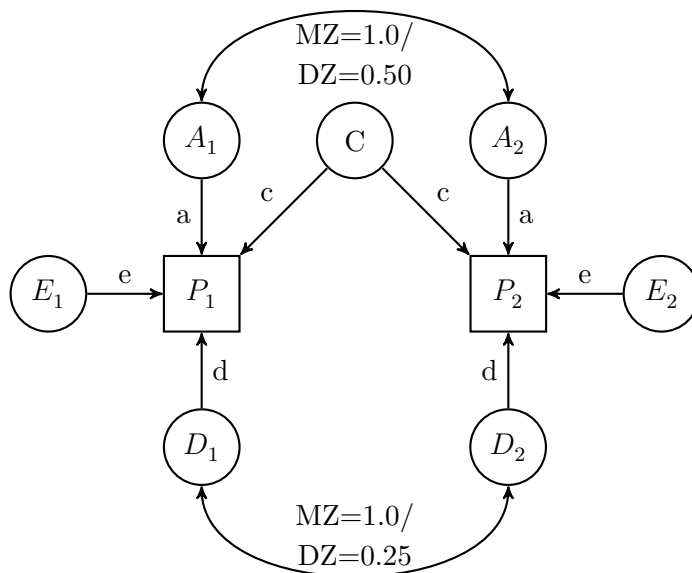


Figure 4.4 ACDE model for a single phenotype for monozygotic (MZ) and dizygotic (DZ) twins raised together. The subscripts in the latent variables refer to sibling.

representing additive genetic effects (A), I set the covariance to 1.00 for the MZ twins and to 0.50 for the DZ twins. Likewise, for the non-additive genetic effects (D), I set the covariance to 1.00 for the MZ twins and to 0.25 for the DZ twins. For all twin types, the covariance between the non-shared environments is set to 0.00 and the covariance between the shared environments is set to 1.00 (this is the same as having a single shared environment LV). The constraint values come from genetic theory.

For the parameter estimates, I standardized the LVs and constrained a given loading to be equal for both siblings. This reduces the number of parameters to estimate to four: a , b , c , and d . As it turns out, with just MZ and DZ twins reared together in the sample, c and d are confounded, so either c or d can be estimated within the same model. The c or d parameters, however, do not necessarily need to be estimated if the hypothesis is that only additive genetic components or random environmental events are influencing the phenotype. Consequently, with twin designs it is typical to test the the following series of models that postulate different genetic and environmental components influence a phenotype: ADE, ACE, AE, and CE. Since there are two indicator variables for all these models, there is $2 \times 3/2 = 3$ pieces of unique information—the variances for twin one and twin two as well as the covariance between them. I have two groups of twins (MZ and DZ), so this doubles the amount of unique information to six. Thus, for, say, the ADE model, the df are $6 - 3$.

To estimate the c and d parameters within the same model would require additional data (e.g., twins separated at birth, relatives of twins).

For a numerical example, Neale and Maes (1992) provide body mass index (BMI) co-variances for young female MZ and DZ twins, which are shown in Table 4.4. As with the invariance example in Section 4.4, I am using summary statistics for the data in this example. Thus, I need to enter the covariance matrices and sample sizes for the MZ and DZ twin

Table 4.4 Body Mass Index Covariances for Monozygotic ($n = 534$) and Dizygotic ($n = 328$) Twins Reared Together.

		Twin 1	Twin 2
MZ	Twin 1	0.725	0.589
	Twin 2	0.589	0.792
DZ	Twin 1	0.779	0.246
	Twin 2	0.246	0.837

MZ: Monozygotic; DZ: Dizygotic. Data taken from Neale and Maes (1992, p. 115).

groups separately, and then combine them into list objects. I do not need the means for this model, so I do not include them.

```
# MZ twins
MZ <- lower2full(c(0.725, 0.589, 0.792))
rownames(MZ) <- colnames(MZ) <- c("P1", "P2")
# DZ twins
DZ <- lower2full(c(0.779, 0.246, 0.837))
rownames(DZ) <- colnames(DZ) <- c("P1", "P2")

# combine the covariances and sample sizes
bmi.cov <- list(MZ = MZ, DZ = DZ)
bmi.n <- list(MZ = 534, DZ = 328)
```

Next, I write the syntax for the models. I only show the syntax for the ADE model, leaving the other models as an exercise (Exercise 4.2). As *lavaan* estimates the error variance by default, it automatically constrains the path coefficient from the error variance to the MV (path e in Figure 4.4) to 1.0. Thus, I constrained the variances of the E LV to be equal across groups.

```
# ADE model
bmi.ade.model<- '
# build the factor model with group constraints
A1 =~ NA*P1 + c(a,a)*P1 + c(.5,.5)*P1
A2 =~ NA*P2 + c(a,a)*P2 + c(.5,.5)*P2
D1 =~ NA*P1 + c(d,d)*P1
D2 =~ NA*P2 + c(d,d)*P2
# constrain the factor variances
A1 ~~ 1*A1
A2 ~~ 1*A2
D1 ~~ 1*D1
D2 ~~ 1*D2
P1~~c(e2,e2)*P1
P2~~c(e2,e2)*P2
# constrain the factor covariances
A1 ~~ c(1,.5)*A2
A1 ~~ 0*D1 + 0*D2
A2 ~~ 0*D1 + 0*D2
D1 ~~ c(1,.25)*D2
'
```

In most LVMs, error is a single-indicator factor. Thus, either the path coefficient *or* the error variance needs be estimated (see Section 3.2.1.1).

Once I have specified the model, obtaining the parameter estimates is the same as with the other models. Since the parameter estimates are constrained to be the same in both groups, I only show the output for the DZ twins.

```
bmi.ade.fit <- cfa(bmi.ade.model, sample.cov = bmi.cov, sample.nobs = bmi.n)
summary(bmi.ade.fit, standardized = TRUE)
```

```
## lavaan (0.5-16) converged normally after 21 iterations
##
```

```
..<Output Omitted>..
```

```
## Group 2 [DZ]:
```

```
##
```

```
##           Estimate Std.err Z-value P(>|z|) Std.lv Std.all
```

```
## Latent variables:
```

```
##   A1 =~
```

```
##     P1      (a)    0.562    0.139    4.053    0.000    0.562    0.636
```

```
##   A2 =~
```

```
##     P2      (a)    0.562    0.139    4.053    0.000    0.562    0.636
```

```
##   D1 =~
```

```
##     P1      (d)    0.543    0.140    3.874    0.000    0.543    0.615
```

```
##   D2 =~
```

```
##     P2      (d)    0.543    0.140    3.874    0.000    0.543    0.615
```

```
##
```

```
## Covariances:
```

```
##   A1 ~~
```

```
##     A2           0.500           0.500    0.500
```

```
##     D1           0.000           0.000    0.000
```

```
##     D2           0.000           0.000    0.000
```

```
##   A2 ~~
```

```
##     D1           0.000           0.000    0.000
```

```
##     D2           0.000           0.000    0.000
```

```
##   D1 ~~
```

```
##     D2           0.250           0.250    0.250
```

```
##
```

```
## Variances:
```

```
##     A1           1.000           1.000    1.000
```

```
##     A2           1.000           1.000    1.000
```

```
##     D1           1.000           1.000    1.000
```

```
##     D2           1.000           1.000    1.000
```

```
##     P1      (e2)    0.170    0.010    0.170    0.218
```

```
##     P2      (e2)    0.170    0.010    0.170    0.218
```

The results for the three estimated parameters using the fully standardized results (the *Std.all* column) are: $a = 0.636$, $e^2 = 0.218$, and $d = 0.615$ (c was constrained to be 0). Using the tracing rules (see Section 2.1.3), results from this model indicate that non-shared environmental influences account for $e^2 = 21.8\%$ of the total variation in BMI. Narrow heritability accounts for $a^2 = 40.4\%$ of the total BMI variation and $\frac{0.404}{0.404+0.378} \times 100 = 51.7\%$ of the broad heritability variance.

4.7 Summary

In this chapter, I discussed estimating LVM with more than one group. Such analyses are useful in two situations. The first situation is when the question of interest revolves around the comparability of constructs between groups (i.e, measurement invariance). This situation requires the use of means in the LVM, which is why they are referred to as mean and covariance structure (MACS) models. The second situation is when the parameters of a LVM require data from two or more unique groups, such as with genetically informative designs.

Both situations require the use of parameter constraints, and I showed two different methods of using parameter constraints: group equality constraints (i.e., impose equality constraints on a whole set of parameter estimates) and using parameter labels. I finished the discussion of each situations by working through an example.

4.8 Writing the Results

When reporting results from a multiple group analysis, follow the guidelines in Section 3.6. In addition, when assessing invariance, document all the decisions made for each level of the invariance investigation. This includes how the baseline model (i.e., each group's initial LVM) was derived and the criteria used to determine whether a given level of invariance fit the data. In addition, be explicit enough in describing the constraints used for each level of invariance (i.e., what parameters are freely estimated and what ones are not) so that the readers could replicate findings with another set of analyses.

A table comparing the different models is mandatory when examining invariance. The table's columns should report: (a) χ^2 , (b) degrees of freedom, (c) values of alternative fit indexes, and (d) the difference in items *a-c* from the prior (less constrained) model. In the table's rows, describe the invariance model in language that most readers can understand, such as *Baseline model*, *Strict Invariance (equal loadings and intercepts)*. It is helpful if the table includes the groups' sample sizes, the estimation method used (e.g., maximum likelihood), and the computer program used. Boomsma et al. (2012, p. 346) write: "The general goal [of the table] is the following: A reader who finds just the table from an invariance manuscript should be able to understand the context of the analysis, the order of the tests, and where, if at all, invariance breaks down."

In a second table, present the parameter estimates for the final model, which includes the unstandardized loadings, their standard errors, the standardized loadings, the intercepts, the latent means, and the latent (co)variances. A path model is optional, but can go a long way to facilitate the readers' understanding of the LVM.

4.9 Exercises

- 4.1 Beaujean and Sheng (in press) compared the different editions of the Wechsler Intelligence Scale for Children (WISC). Table 4.5 gives the selected WISC subtest correlations, means, and standard deviations for the 7-year-old age group.
 - 4.1.a Enter the correlations, means, and standard deviations for all four editions. Then, combine them into named list objects.
 - 4.1.b Using edition as the grouping variable, test for invariance. (*Hint*: Beaujean and Sheng had to free some parameter estimates from being invariant as well as allow some residuals to covary for the model to fit the data decently.)
- 4.2 Fit the ACE, CE, and AE models to the BMI covariance matrix, `bmi.cov`, from Section 4.6.

Table 4.5 Correlations, Standard Deviations, and Means of WISC data for Exercise 4.1. Data taken from Beaujean and Sheng (in press).

	Co	A	S	V	DS	PC	BD	Cd
Comprehension	1.00	0.43	0.45	0.61	0.34	0.33	0.43	0.18
Arithmetic	0.31	1.00	0.43	0.50	0.47	0.25	0.42	0.22
Similarities	0.36	0.40	1.00	0.59	0.34	0.36	0.50	0.18
Vocabulary	0.51	0.46	0.45	1.00	0.35	0.37	0.43	0.27
Digit Span	0.29	0.40	0.33	0.43	1.00	0.18	0.32	0.18
Picture Completion	0.39	0.29	0.27	0.36	0.33	1.00	0.42	0.13
Block Design	0.32	0.27	0.29	0.33	0.24	0.28	1.00	0.26
Coding	0.22	0.32	0.15	0.22	0.27	0.12	0.26	1.00
WISC SD	2.69	1.50	2.36	6.06	1.85	2.18	5.97	9.94
WISC Mean	7.83	5.50	5.67	21.50	7.67	8.00	6.50	34.83
WISC-R SD	3.63	1.77	3.26	5.07	2.86	4.25	8.82	8.05
WISC-R Mean	11.00	7.83	8.33	19.67	8.67	13.67	12.00	39.00

(a) Correlations, Standard Deviations, and Means for WISC ($n = 200$) and WISC-R ($n = 200$). WISC correlations are in the lower diagonal; WISC-R correlations are in the upper diagonal.

	Co	A	S	V	DS	PC	BD	Cd
Comprehension	1.00	0.46	0.58	0.63	0.27	0.45	0.33	0.15
Arithmetic	0.33	1.00	0.55	0.43	0.51	0.38	0.52	0.27
Similarities	0.46	0.46	1.00	0.73	0.37	0.37	0.49	0.16
Vocabulary	0.61	0.41	0.64	1.00	0.33	0.43	0.41	0.09
Digit Span	0.22	0.39	0.36	0.30	1.00	0.13	0.29	0.12
Picture Completion	0.23	0.30	0.33	0.31	0.25	1.00	0.43	0.25
Block Design	0.22	0.36	0.37	0.35	0.30	0.47	1.00	0.23
Coding	0.13	0.22	0.18	0.19	0.09	0.24	0.20	1.00
WISC-III SD	3.52	2.19	3.19	4.76	2.53	3.85	10.25	10.51
WISC-III Mean	12.50	12.83	10.00	17.00	10.50	13.67	18.67	43.67
WISC-IV SD	4.93	4.10	5.20	6.54	2.72	5.35	9.36	10.44
WISC-IV Mean	15.17	15.00	11.83	21.67	12.17	17.83	18.67	45.83

(b) Correlations, Standard Deviations, and Means for WISC-III ($n=200$) and WISC-IV ($n=200$). WISC-III correlations are in the lower diagonal; WISC-IV correlations are in the upper diagonal.

4.3 Little, Slegers, and Card (2006) presented an example of multiple group analysis using all three LV identification methods discussed in Section 4.2.1.1. Their model is shown in Figure 4.5, and the data they used are given in Table 4.6.

4.3.a Enter the correlations, standard deviations, and means for both groups.

4.3.b Fit configural, weak, strong invariance models using all three scaling methods (marker variable, standardized LV, and effects-coding). For the marker variable method, use indicators 1 and 5 as the marker variables.

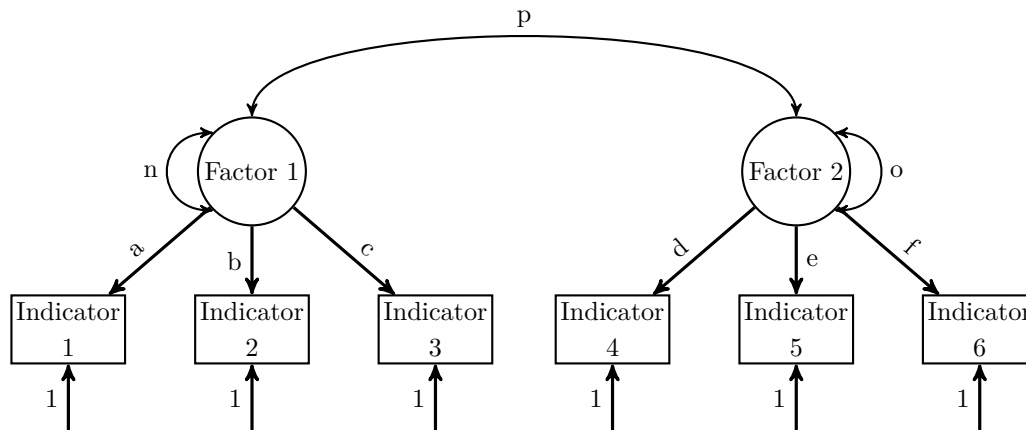


Figure 4.5 Two-factor model with means and intercepts not shown. Model based on description in Little et al. (2006).

Table 4.6 Correlations, Standard Deviations, and Means for Exercise 4.3.

	Ind 1	Ind 2	Ind 3	Ind 4	Ind 5	Ind 6
Indicator 1	1.000	0.813	0.850	−0.188	−0.289	−0.293
Indicator 2	0.759	1.000	0.835	−0.155	−0.250	−0.210
Indicator 3	0.762	0.787	1.000	−0.215	−0.338	−0.306
Indicator 4	0.028	0.010	−0.058	1.000	0.784	0.800
Indicator 5	−0.061	−0.061	−0.141	0.785	1.000	0.832
Indicator 6	−0.022	−0.052	−0.102	0.816	0.816	1.000
Group One SD	0.668	0.685	0.707	0.714	0.663	0.653
Group One Mean	3.135	2.991	3.069	1.701	1.527	1.545
Group Two SD	0.703	0.718	0.762	0.650	0.602	0.614
Group Two Mean	3.073	2.847	2.979	1.717	1.580	1.550

Correlations for group one are given in the lower diagonal, and those for group two are given in the upper diagonal. Group one: $n = 380$, Group two: $n = 379$. Data taken from Little et al. (2006, pp. 71-72).

4.10 References & Further Readings

- Beaujean, A. A., Freeman, M. J., Youngstrom, E., & Carlson, G. (2012). The structure of cognitive abilities in youths with manic symptoms: A factorial invariance study. *Assessment*, *19*, 462-471. doi: 10.1177/1073191111399037
- Beaujean, A. A., & Sheng, Y. (in press). Assessing the Flynn effect in the Wechsler scales. *Journal of Individual Differences*.
- Boomsma, A., Hoyle, R. H., & Panter, A. T. (2012). The structural equation modeling research report. In R. H. Hoyle (Ed.), *Handbook of structural equation modeling*. New York, NY: Guilford.
- Horn, J. L., & McArdle, J. J. (1992). A practical and theoretical guide to measurement invariance in aging research. *Experimental Aging Research*, *18*, 117-144.
- Little, T. D. (1997). Mean and covariance structures (MACS) analyses of cross-cultural data: Practical and theoretical issues. *Multivariate Behavioral Research*, *32*, 53-76. doi: 10.1207/s15327906mbr3201_3
- Little, T. D., Card, N. A., Slegers, D. W., & Ledford, E. C. (2007). Representing contextual effects in multiple-group MACS models. In T. D. Little, J. A. Bovaird, & N. A. Card (Eds.), *Modeling ecological and contextual effects in longitudinal studies* (pp. 121-147). Mahwah, NJ: Lawrence Erlbaum Associates.
- Little, T. D., Slegers, D. W., & Card, N. A. (2006). A non-arbitrary method of identifying and scaling latent variables in SEM and MACS models. *Structural Equation Modeling: A Multidisciplinary Journal*, *13*, 59-72. doi: 10.1207/s15328007sem1301_3
- Meade, A. W., Johnson, E. C., & Braddy, P. W. (2008). Power and sensitivity of alternative fit indices in tests of measurement invariance. *Journal of Applied Psychology*, *93*, 568-592. doi: 10.1037/0021-9010.93.3.568
- Millsap, R. E. (2011). *Statistical approaches to measurement invariance*. New York, NY: Routledge.
- Neale, M. C., & Maes, H. H. M. (1992). *Methodology for genetic studies of twins and families*. Dordrecht, The Netherlands: Kluwer Academic Publishers.
- Plomin, R., DeFries, J. C., McClearn, G. E., & McGuffin, P. (2008). *Behavioral genetics* (5th ed.). New York: Worth.
- Rutter, M. (2007). Proceeding from observed correlation to causal inference: The use of natural experiments. *Perspectives on Psychological Science*, *2*, 377-395. doi: 10.1111/j.1745-6916.2007.00050.x
- Steenkamp, J.-B. E. M., & Baumgartner, H. (1998). Assessing measurement invariance in cross-national consumer research. *Journal of Consumer Research*, *25*, 78-107. doi: 10.1086/209528
- Vandenberg, R. J., & Lance, C. E. (2000). A review and synthesis of the measurement invariance literature: Suggestions, practices, and recommendations for organizational research. *Organizational Research Methods*, *3*, 4-70. doi: 10.1177/109442810031002
- Wechsler, D. (1991). *Wechsler intelligence scale for children* (3rd ed.). San Antonio, TX: The Psychological Corporation.

5 | Models with Multiple Time Periods

Chapter Contents

5.1	Background	79
5.1.1	Latent Curve Models in <code>lavaan</code>	80
5.2	Example: Latent Curve Model	80
5.3	Latent Curve Model Extensions	84
5.3.1	Adding Covariates	84
5.3.2	Multi-Group Models	87
5.3.3	Alternative Ways to Specify the Latent Intercept and Slope	87
5.4	Summary	88
5.5	Writing the Results	88
5.6	Exercises	89
5.7	References & Further Readings	92

5.1 Background

This far, all of the models I have considered used data measured at a single time period. LVMs are versatile, though, and can handle data that has been measured at multiple time periods (i.e., repeated measures). There are a variety of ways to analyze such data, such as growth models, multilevel models, and random regression/mixed-effects models. When using a LVM, though, the model is typically called a latent growth curve model or simply a **latent curve model** (LCM).

Latent Curve Model

In LCMs, each respondent in the sample has his or her own trajectory of change over time on the measured variable(s). Thus, characteristics of the trajectory (i.e., slope, intercept) can vary across individuals. To allow this variation, the parameters are modeled as LVs. I call these variables *latent intercept* and *latent slope*, so as not to confuse them with the manifest intercepts I discussed in Section 4.2.1. Like any other LV, the latent intercept and latent slope can be exogenous or endogenous. Figure 5.1 contains a path model of a LCM with data across four time periods.

LCMs provide a great amount of flexibility in examining change over time. Moreover, more traditional forms of modeling growth (e.g., repeated measures ANOVA), as well as more modern forms of modeling growth (e.g., multilevel model), can be parametrized as a LCM. What sets apart LCM from other methods of analyzing change is that LCMs are explicit in modeling the latent slope and latent intercept’s covariances, as well as modeling the structure of the error variances. Moreover, like all LVMs, LCMs can examine how different models fit the data.

In the LVM models I have examined thus far, the loadings were usually estimated, unless they were constrained for identification or to examine invariance. In LCM, typically all the loadings are fixed. For the latent intercept, the loadings are fixed to 1.0, which indicates the latent intercept has equal influence on the indicator variables across all waves of data collection. The latent slope's loadings can have a variety of values, increasing from one time point to another, although they are somewhat dependent on the time between data collections and the nature of the hypothesized growth. The best values to start with are 0.0 for the first time point and then have the values increase by 1.0 for the subsequent loadings (i.e., 0.0, 1.0, 2.0, 3.0), which reflects an equal amount of time passing between data collection time periods. Starting with 0.0 gives the mean of the latent intercept the interpretation of being the mean value of the outcome variable at the first time point.

LCMs not only need to account for the means of the manifest variables, but they need to account for mean changes as well. Thus, they need to include the manifest variables' intercepts as well as the latent means (see Section 4.2.1). Usually, the manifest intercepts are constrained to be 0.0, which makes the manifest variables' means modeled entirely by the latent means.¹ The residual (co)variances and latent (co)variances are modeled as usual, unless there is a need to constrain their values.

Bollen and Curran (2006) give three questions to ask when initially creating a LCM:

1. What is the average trajectory for *all* respondents? Stated differently: (a) what is the initial level of the variable (i.e., mean); (b) does the variable change over time; and, if so, (c) what is the form of the change (e.g., linear, quadratic)?
2. Is the average trajectory sufficient, or is there enough variability between observations to warrant allowing distinct trajectories? In other words, does the latent intercept or latent slope need to have a variance term or can the variance be constrained to 0.0? These latent slope and latent intercept variances are sometimes called **random effects**.
3. If distinct trajectories are needed (i.e., the latent slope or latent intercept has variability), are there variables that can predict them? That is, do other variables need to be added to the LCM to explain the latent intercept or latent slope variability?

Random Effects

5.1.1 Latent Curve Models in lavaan

growth()

The easiest way to specify a LCM in `lavaan` is to use the `growth()` function. The `growth()` function is very similar to the `sem()` function, but it assumes there are means and intercepts in the model, with the observed intercepts fixed to zero and the latent means freely estimated.

5.2 Example: Latent Curve Model

The data are crime rates in various New York and Pennsylvania communities, measured at four equidistant time periods across eight months in 1995.² Summary statistics are given in

¹An alternative parameterization is to set the latent means equal to 0.0 and estimate the intercepts instead.

²The outcome variable is actually the logarithm of the crime rates. The authors transformed the variables to have them better approximate a normal distribution.

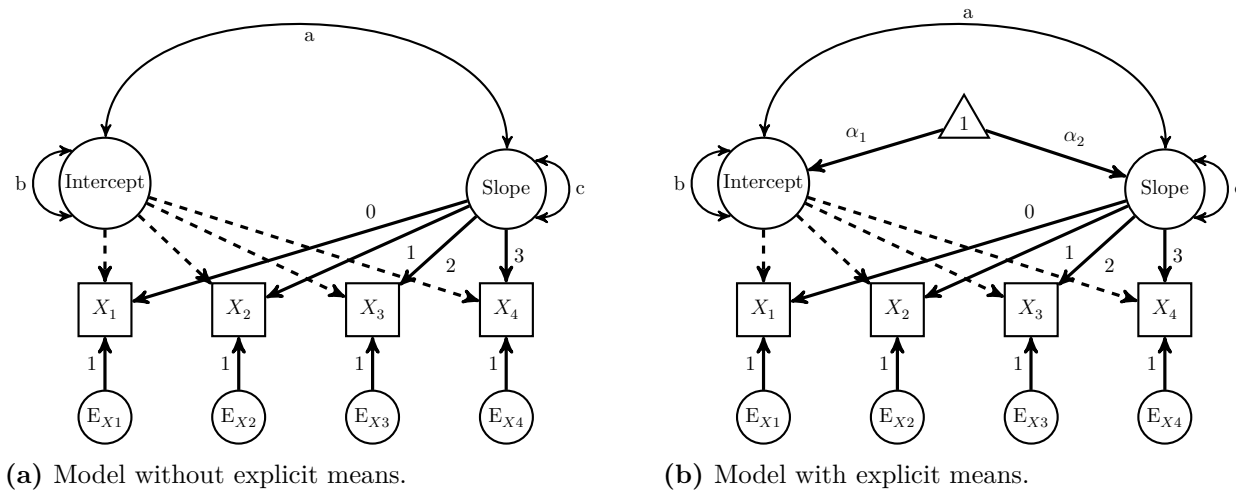


Figure 5.1 Example latent curve model with data collected at four time periods. Dashed paths have loadings equal to 1.0.

Table 5.1. I input the covariance matrix and the mean vector into objects named `crime.cov` and `crime.mean`, respectively. I do not show the syntax for space considerations.

An analysis of LCMs typically requires fitting multiple models. Figure 5.2 shows the `lavaan` syntax for a series of increasingly less restrictive LCMs. The results from the model specified in Figure 5.2a, `crime.model1`, are given below. This model only contains an average latent intercept and constrains the residual variances to be the same across all four time periods. Such a model suggests there is no change over time, and that the average crime rate at time one is sufficient for all communities and all time periods.

```
crime.fit1 <- growth(crime.model1, sample.cov=crime.cov, sample.mean=crime.mean,
sample.nobs=952)
summary(crime.fit1)
```

Table 5.1 Covariances and Means of New York and Pennsylvania Crime Data ($n = 952$).

	Time 1	Time 2	Time 3	Time 4
Time 1	0.63	0.50	0.48	0.47
Time 2	0.50	0.60	0.48	0.48
Time 3	0.48	0.48	0.58	0.51
Time 4	0.47	0.48	0.51	0.67
Mean	5.17	5.32	5.40	5.52

Data taken from Bollen and Curran (2006), and available at: <http://www.ats.ucla.edu/stat/examples/lcm/>

```
crime.model1 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
i~~0*i
# residual variances
Time1~~r*Time1
Time2~~r*Time2
Time3~~r*Time3
Time4~~r*Time4
'
```

(a) Model for a mean latent intercept and constrained residual variances.

```
crime.model3 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
s~~0*s
s~~0*i
# residual variances
Time1~~r*Time1
Time2~~r*Time2
Time3~~r*Time3
Time4~~r*Time4
'
```

(c) Model with a mean latent intercept that is allowed to vary, mean latent slope, and constrained residual variances.

```
crime.model5 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
'
```

(e) Unconstrained model.

```
crime.model2 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# residual variances
Time1~~r*Time1
Time2~~r*Time2
Time3~~r*Time3
Time4~~r*Time4
'
```

(b) Model for a mean latent intercept that is allowed to vary, and constrained residual variances.

```
crime.model4 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
# residual variances
Time1~~r*Time1
Time2~~r*Time2
Time3~~r*Time3
Time4~~r*Time4
'
```

(d) Model for a mean latent intercept that is allowed to vary, mean latent slope that is allowed to vary, and constrained residual variances.

Figure 5.2 lavaan syntax for different latent curve models.

```
## lavaan (0.5-16) converged normally after 13 iterations
```

```
..<Output Omitted>..
```

```
##
##               Estimate Std.err Z-value P(>|z|)
## Latent variables:
##   i =~
##     Time1           1.000
##     Time2           1.000
##     Time3           1.000
##     Time4           1.000
##
## Intercepts:
##     Time1           0.000
##     Time2           0.000
```

```
##      Time3      0.000
##      Time4      0.000
##      i          5.351    0.013  413.817    0.000
##
## Variances:
##      i          0.000
##      Time1      (r)    0.637    0.015
##      Time2      (r)    0.637    0.015
##      Time3      (r)    0.637    0.015
##      Time4      (r)    0.637    0.015
```

The results show that the mean for the latent intercept, i , is 5.351, indicating that, on average, the communities have a crime rate of 5.351 at all time periods. The amount of unexplained variability (i.e., residual or error variance) is 0.637, indicating that this single-latent intercept model explained 36.327 % of the variability in crime rates.

I do not show the output for the other four LCMs in Figure 5.2 to save space, but interpret their results. The second model, `crime.model2` (Figure 5.2b), allows the latent intercept to vary. Although it still posits there is no change over time, it examines if any of the residual variance is due to between-community variability. It does this by allowing the latent intercept to have a variance term associated with it. The residual variance for this model is 0.157, down from 0.637 in the first model. The variance for the latent intercept is 0.48, which, when taken with the knowledge about the residual variance (i.e., within-community variability), indicates that 75.4% of the total variance in crime rate is explained by differences between communities.

$$75.4\% = \frac{0.48}{0.48 + 0.157} \times 100$$

The third LCM, `crime.model3` (Figure 5.2c), allows the latent slope to vary. This model still posits there is no *systematic* change over time, but there are *random* changes beyond what the variability at time one (i.e., latent intercept variance) can explain. With this LCM, the residual variance has now shrunk to 0.11, down from 0.157 in the second model.

The fourth LCM, `crime.model4` (Figure 5.2d), finally allows for systematic change in the crime rate over time as well as variability in this average change, i.e., each county can deviate from the average change. In addition, since both the latent intercept and latent slope can vary, this model allows the latent intercept and slope to covary.

The mean for the latent slope, s , is 0.113, indicating that, on average, the crime rate increased 0.113 units between each time period. The covariance between the intercept and the slope, -0.023, suggests that there is less crime growth for those communities that had higher crime rates at time point one than for those communities with lower crime rates. The residual variance is now 0.106. While this is substantially lower than the initial estimate of 0.637, within rounding error it is the same as the residual variance in the third model. Moreover, since the 95% confidence interval for the residual variance (0.10, 0.113) does not contain 0.0, this indicates that there is more outcome variance to explain.

The last model, `crime.model5` (Figure 5.2e), allows the residual variances to vary across time periods. The unexplained variance for this model ranges from 0.085 to 0.138.

Table 5.2 Covariances and Means of Crime Data with Time-Invariant Covariates ($n = 952$).

	Time 1	Time 2	Time 3	Time 4	State	Poverty	StPov
Time 1	0.63	0.50	0.48	0.47	−0.06	1.87	1.16
Time 2	0.50	0.60	0.48	0.48	−0.07	1.80	1.11
Time 3	0.48	0.48	0.58	0.51	−0.08	1.64	1.04
Time 4	0.47	0.48	0.51	0.67	−0.09	1.57	0.93
State	−0.06	−0.07	−0.08	−0.09	0.24	−0.04	−0.41
Poverty	1.87	1.80	1.64	1.57	−0.04	64.04	42.59
StPov	1.16	1.11	1.04	0.93	−0.41	42.59	43.23
Variable Means	5.17	5.32	5.40	5.52	0.62	−1.67	−1.08

Data taken from Bollen and Curran (2006), and available at: <http://www.ats.ucla.edu/stat/examples/lcm/> *State* was dichotomously coded (0=New York, 1=Pennsylvania). *Poverty* is mean centered. *StPov* is the interaction between State and Poverty.

5.3 Latent Curve Model Extensions

5.3.1 Adding Covariates

Time-Invariant
Covariate

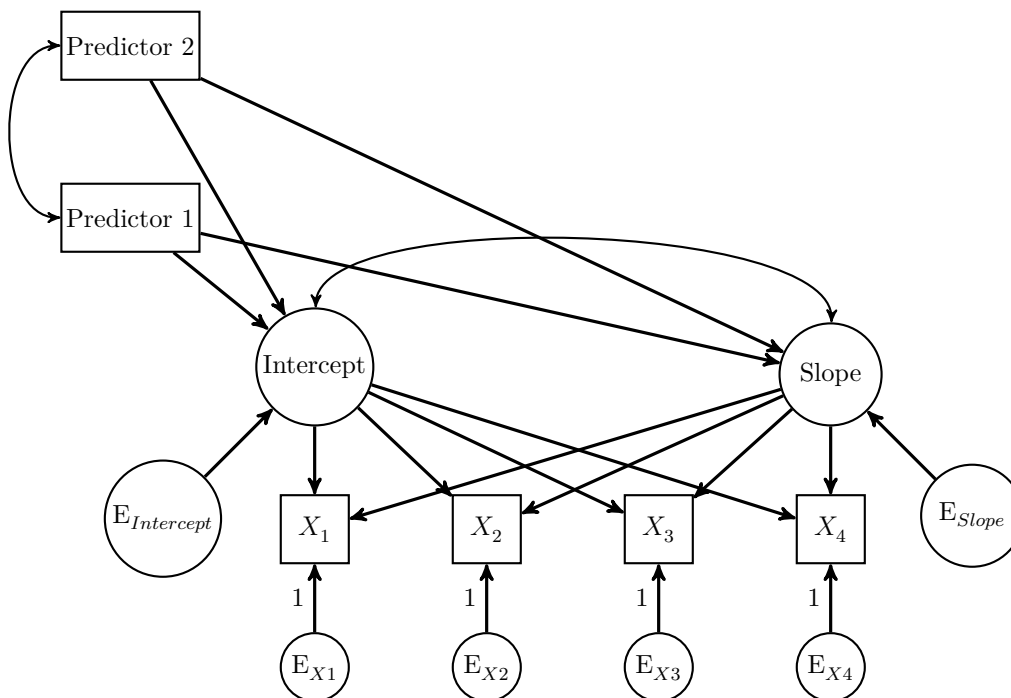
Time-Varying
Covariate

There are two kinds of covariates to add to a LCM. The first is **time-invariant**, meaning that the variables’ values are stable over all the data collection time periods. Examples of time-invariant variables are race, sex, and cognitive ability. The second type of covariate is a variable whose values can vary with time (i.e., **time-varying**). Some examples are attitudes, weight, and age. Path models of both types of covariates are shown in Figure 5.3. In this section, I extend the models I introduced in Section 5.2 to include two time-invariant predictors. Some of the sources listed in Section 5.7 gives examples with time-varying covariates.

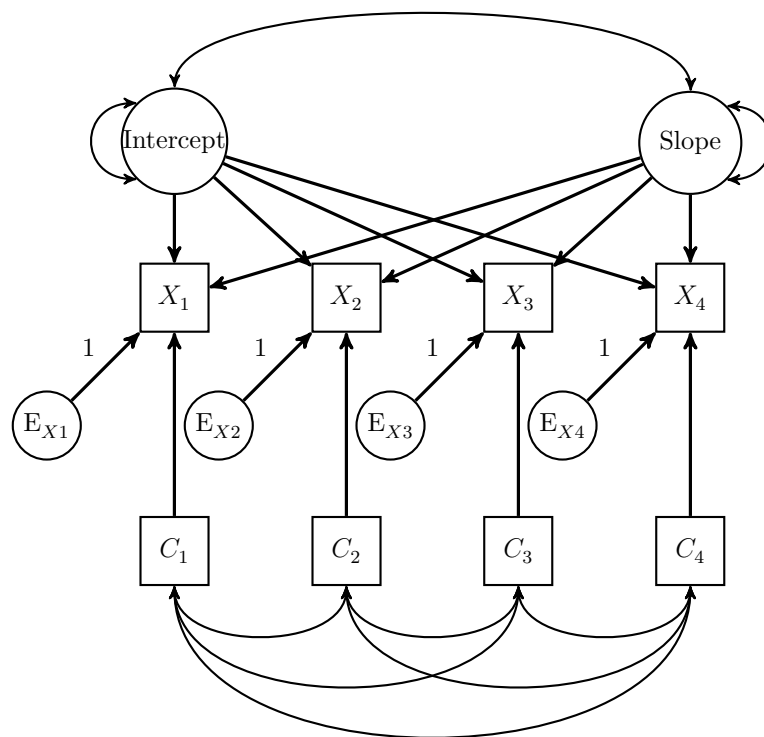
At the end of Section 5.2, I fit a model, `crime.model15`, that had a random latent intercept, random latent slope, and residual variances that change at each time point. While this explained a fair amount of the within-community variability in crime rates, there was still a non-negligible amount of unexplained variance. Moreover, the model allowed the latent intercept and latent slope to vary between communities, but I did not examine any variables that might explain either variable’s variance. One way to address these issues is to add covariates to the model.

There are three other predictors in the dataset: (a) state (Pennsylvania and Delaware) (*State*), (b) percent of the community living in poverty (*Poverty*), and (c) the interaction between state and poverty (*StPov*). The revised covariances and means are shown in Table 5.2, and the `lavaan` syntax for models with the covariates are in Figure 5.4. For each model, I used the revised data, which I named `crime2.cov` (covariances) and `crime2.mean` (means).

For the syntax in Figure 5.4, the group membership variable (*State*) is dummy-coded (Penn=0, Del=1). Using this coding mechanism allows me to examine whether the mean of the latent intercept or mean of the latent slope differs across the two states. The residuals are allowed to vary across time, but not across states. Thus, the latent intercept and latent



(a) Time-invariant covariates (multiple indicators multiple independent causes [MIMIC]).



(b) Time-varying covariates.

Figure 5.3 Latent curve models with manifest covariates (predictor variables). Except for the latent intercept and latent slope in (b), the exogenous variables' variances are not shown.

```
crime.model6 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
# regression
i + s ~ State
'
```

(a) State as a predictor.

```
crime.model7 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
# regression
s + i ~ State + Poverty
'
```

(b) State and poverty as predictors.

```
crime.model8 <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
# regression
s + i ~ State + Poverty + StPov
'
```

(c) State, poverty, and their interaction as predictors.

Figure 5.4 lavaan syntax for latent curve models with covariates.

slope are forced to be equally good predictors of crime rate in both states. This is a **multiple indicators multiple independent causes** (MIMIC) model. MIMIC models examine the relationship between exogenous manifest variables (the independent causal indicators) and endogenous latent variables. Often, MIMIC models are used to control for demographic characteristics (e.g., sex, race) or as an alternative to assessing structural invariance (see Section 4.2), especially when there are multiple grouping variables or small sample sizes. They are less thorough because they assume the covariance matrices are invariant (i.e., strong invariance).

For the model shown in Figure 5.4a, state is used to predict the latent intercept and latent mean. The mean of the latent intercept is 5.337 and the regression coefficient from it to state is -0.251. Thus, the initial average total crime rate in New York is 5.337, while in Pennsylvania it is 5.086. Likewise, the mean of the latent slope is 0.143 and the regression coefficient from it to state is -0.047. Thus, the average change in total crime rate per time period in New York is 0.143, while in Pennsylvania it is 0.10. Thus, New York not only starts with a higher average crime rate, it also, on average, increases faster than Pennsylvania.

$$5.086 = 5.337 + -0.251$$

$$0.10 = 0.143 + -0.047$$

In addition to having error variances and R^2 values for each of the MVs, having covariates predict the latent intercept and latent slope forces there to be error variances and R^2 values for them as well. The R^2 for the latent intercept and latent slope are 0.029 and 0.036, respectively, indicating that state does not explain much of the between-community variability in the initial crime rate or changes in crime rate.

Adding a quantitative covariate (Figure 5.4b) is similar to adding a categorical one, only now the interpretation is not just differences between states, but also differences as the poverty rate increases/decreases. Thus, the regression coefficient from poverty to the latent intercept of 0.029 indicates that, controlling for the state where the community is located, communities

with higher poverty rates started with higher crime rates at the first time point than communities with lower poverty rates. The regression coefficient from poverty to the latent slope of -0.002 indicates that, controlling for state where the community is located, as the community poverty rate increases, the crime rate slightly decreases.

The last model (Figure 5.4c) examines if there is an interaction between the time-invariant predictors. Examining interactions between time-invariant predictors is similar to examining an interaction in a multiple regression. The focus shifts from the predictors' main effects to how the relationship of one predictor with the outcome changes as a function of another predictor (i.e., statistical moderation). In this case, however, the outcomes are the latent intercept and latent slope.

`lavaan` does not currently allow for the creation of interaction terms within the model specification, so I made the interaction in the initial dataset. In Table 5.2, the interaction variable is named *StPov*.

Use the `*` operation to make an interaction variable, e.g.,
`int <- var1 * var2`

The interaction coefficient for the latent intercept is -0.012, indicating there is a small, but non-negligible, effect of poverty on the mean differences for the latent intercept values between New York and Pennsylvania. In other words, the mean difference in crime rate between states at time one differs, albeit slightly, for higher poverty vs. lower poverty counties. The interaction coefficient for the latent slope is -0.001 (95% CI: -0.005-0.002), which is minimal. Thus, the differences in the average change in crime rates between states do not appear to be a function of poverty level.

5.3.2 Multi-Group Models

In Section 5.3.1, I fit a MIMIC model to the data by adding a group-membership covariate (*State*) to the LCM. This allowed me to determine if there was a difference in the average latent intercept or latent slope between groups. I stated that this was not a thorough way to test for invariance, because the model assumes the covariance matrices are at least strongly invariant. Sometimes, the question of interest is between-group differences that go beyond the means of the latent intercept or latent slope. With LCM, these differences are usually: (a) growth trajectory (e.g., linear vs. quadratic); (b) variance of the intercept and slope, as well as their covariance; or (c) residual variances.

The same procedures I explained in Chapter 4 can also be used for a more thorough assessment of invariance in LCMs. Here, instead of using group membership as a covariate (as I did in Table 5.2), the data are entered for each group separately (or there is a group membership variable if the raw data is used for the analysis). Then, a series of more restrictive models is tested. I leave fitting a multi-group LCM as an exercise (Exercise 5.1).

5.3.3 Alternative Ways to Specify the Latent Intercept and Slope

In the example I have used in this chapter, I coded time so that the latent intercept reflected the initial time data was collected and the latent slope reflected a one-unit change in time for each data collection point. There is nothing that requires the LCM to have either definitions for the latent intercept or latent slope.

In Figure 5.5, I present some alternative ways of coding time. The first three examples involve changing the definition of the latent intercept. In Figure 5.5a, the *last* point of data collection is the latent intercept. In Figure 5.5b, the latent intercept is some point in the middle of the data collection, while in Figure 5.5c the latent intercept is the exact middle of the time collection. Such coding would be useful if, say, the research design involved a treatment that was administered after the data collection began.

The last four examples involve changing the latent slope. In Figure 5.5d, the time periods are coded to be two units (e.g., data was collected every two months). In Figure 5.5e, the time periods are coded to be one unit, but there is a follow-up data collection point four units after the major part of the study has ended. The manifest variable's change does not have to be linear, though. Modeling non-linear change, though, requires the inclusion of multiple latent slopes. For example, Figure 5.5f shows syntax for quadratic growth. The first latent slope accounts for the linear trend while the second accounts for the quadratic trend. As another example, Figure 5.5g shows syntax for a *piecewise* linear trend (sometimes called a spline model) across six time periods, with the latent slope changes at time four (i.e., the knot).

While the examples in Figure 5.5 show either changes in latent intercept or latent slope, they can be combined. In fact, it is possible to code the latent intercept and latent slope to have just about any definition of interest. The difficulty is finding the correct values to use to constrain the latent intercept's and latent slope's loadings. Bollen and Curran (2006) present some general methods for finding such values.

5.4 Summary

In this chapter, I presented how to model longitudinal data using a LVM framework, specifically the use of latent curve models (LCMs). I explained how to fit a traditional LCM, and then explained how to expand this model to include covariates, introduced multiple indicators multiple independent causes (MIMIC) models, and showed alternative methods of coding the latent intercept and latent slope. I worked through an example in **R** using `lavaan`'s `growth()` function.

5.5 Writing the Results

Writing on LCMs is a little different than other LVMs, as the focus is on the means of the latent intercept and latent slope, as well as their variability. While many of the guidelines in Section 3.6 are still applicable, with LCM it is very important to describe the indicator variables thoroughly. This includes not only reporting their means and variances, but their possible range of values as well as the range in the sample. Moreover, if the indicator variables come from a psychological instrument, then report the scores' reliabilities as well.

When describing the LCM, be sure to describe what time period was used to define the latent intercept, as well as why it was used, when interpreting it. Likewise, describe the method and units used to code time (e.g., equally spaced intervals of two months), and if more than one slope was included (i.e., for non-linear change). If the model includes other

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ -3*Time1 + -2*Time2 + -1*Time3 + 0*Time4
'
```

(a) Model with the intercept coded to be the *last* data collection period.

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ -1.5*Time1 + -0.5*Time2 + 0.5*Time3 + 1.5*Time4
'
```

(c) Model with the intercept coded to be the *exact middle* of data collection.

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 1*Time2 + 2*Time3 + 6*Time4
'
```

(e) Model with a follow-up time period distant (four units) from the end of regular data collection.

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ -2*Time1 + -1*Time2 + 0*Time3 + 1*Time4
'
```

(b) Model with the intercept coded to be the *third* data collection period.

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope
s =~ 0*Time1 + 2*Time2 + 4*Time3 + 6*Time4
'
```

(d) Model using two units as the time between data collection periods.

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4
# slope 1
s1 =~ 0*Time1 + 1*Time2 + 2*Time3 + 3*Time4
# slope 2
s2 =~ 0*Time1 + 1*Time2 + 4*Time3 + 9*Time4
'
```

(f) Model with quadratic growth. The second latent slope's loadings are the square of the first latent slope's loadings.

```
model <- '
# intercept
i =~ 1*Time1 + 1*Time2 + 1*Time3 + 1*Time4 + 1*Time5 + 1*Time6
# slope 1
s1 =~ -3*Time1 + -2*Time2 + -1*Time3 + 0*Time4 + 0*Time5 + 0*Time6
# slope 2
s2 =~ 0*Time1 + 0*Time2 + 0*Time3 + 0*Time4 + 1*Time5 + 2*Time6
'
```

(g) Model with piecewise linear slopes.

Figure 5.5 lavaan syntax for alternative ways of coding time in a latent curve model.

covariates (see Section 5.3.1), describe them thoroughly, including the coding scheme for any categorical variables.

Path models are a very helpful tool to describe a LCM. As the loadings are typically all constrained, be sure to include all their values in the model as well as the means and variances of the latent intercept and latent slope(s).

5.6 Exercises

5.1 Cohen, Cohen, West, and Aiken (2003) present data for a LCM using five time periods and two groups. The covariances and means are given in Table 5.3.

Table 5.3 Covariances and Means of Longitudinal Data for Two Groups for Exercise 5.1.

Group 0 ($n = 30$)					
	T1	T2	T3	T4	T5
Time Period 1	3.59	3.11	2.91	3.22	2.88
Time Period 2	3.11	3.10	2.80	3.05	2.63
Time Period 3	2.91	2.80	2.82	2.86	2.62
Time Period 4	3.22	3.05	2.86	3.30	2.82
Time Period 5	2.88	2.63	2.62	2.82	2.71
Time Period Mean	11.97	11.72	12.03	11.96	12.10

Group 1 ($n = 30$)					
	T1	T2	T3	T4	T5
Time Period 1	3.42	3.03	2.62	2.95	2.89
Time Period 2	3.03	3.18	2.73	2.97	2.91
Time Period 3	2.62	2.73	2.69	2.59	2.67
Time Period 4	2.95	2.97	2.59	3.02	2.83
Time Period 5	2.89	2.91	2.67	2.83	3.25
Time Period Mean	9.80	12.00	13.94	15.96	18.10

Data taken from Cohen et al. (2003).

5.1.a Import the data. Remember to create separate covariance matrices and mean vectors for both groups, and then combine them as a `list` object.

5.1.b Create the syntax for the following series of models:

1. Latent intercept with residuals constrained to be the same across time periods. Constrain the means of the latent intercept to be the same between groups.
2. Add a variance term to the latent intercept in model 1 and constrain the variance to be the same between groups.
3. Add a latent slope to model 2. Constrain the latent slope's mean to be 0.0, but estimate its variance and the latent slope and latent intercept's covariance. Constrain the latent variables' variances and covariance to be the same between groups.
4. Remove the constraint in model 3 that the latent slope's mean is 0.0, but constrain it to be the same between groups.
5. Remove the constraint in model 4 that the mean of the latent intercept is the same between groups (i.e., intercept \times group interaction).
6. Remove the constraint in model 5 that the mean of the latent slope is the same between groups (i.e., slope \times group interaction).

5.1.c Fit the models created in Exercise 5.1.b to the data.

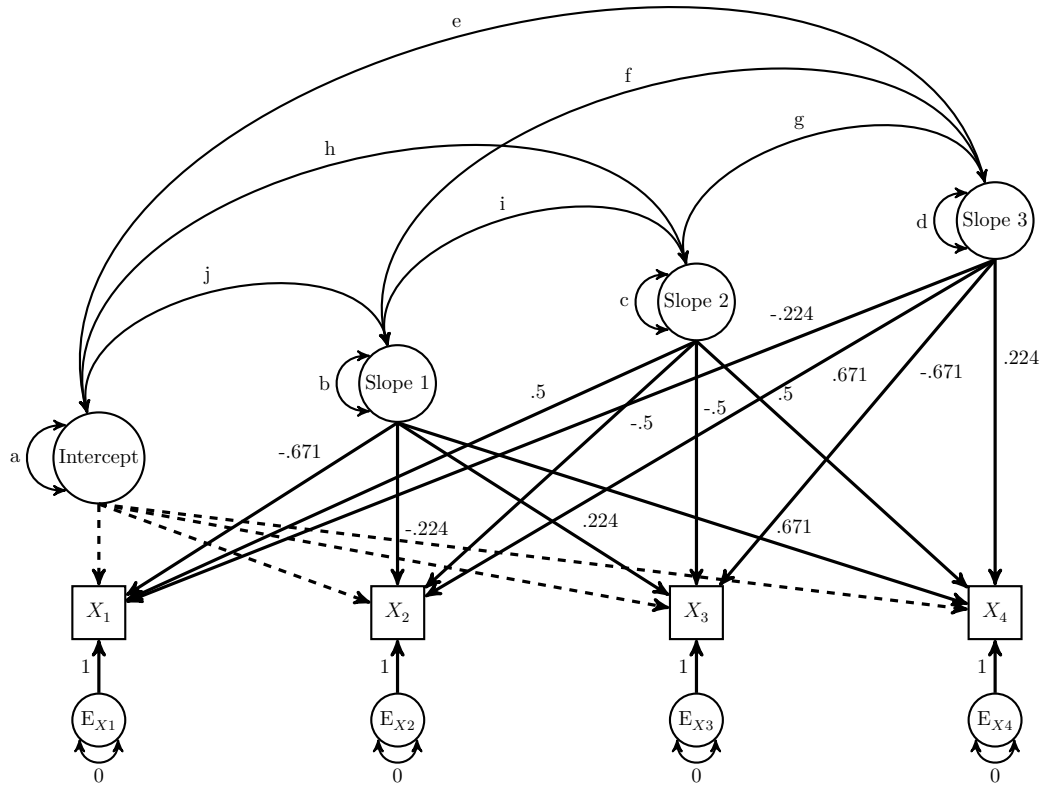


Figure 5.6 Repeated measures ANOVA specified as a latent curve model for Exercise 5.2. Latent means not shown for space considerations, but are parameters k - n , for *Intercept-Slope 3*, respectively. See Voelkle (2007) for more information on the derivation of the loading values. Dashed paths have loadings equal to 1.0.

5.2 Voelkle (2007) described how LCM can be constrained to become more traditional analyses of multiple time periods, such as a repeated-measures ANOVA. His data are scores on a learning task, where performance has been assessed on four consecutive equidistant time periods (x_1 to x_4) on 17 female and 18 male participants. His data `voelke.dat` are available as a comma-delimited file on the book's website.

5.2.a Use the `voelke.dat` data to do a repeated measures analysis of variance (ANOVA) in **R**. (Hint: Before doing a typical repeated measures ANOVA, you need to reshape the data to be in *long* format. I suggest using the `reshape()` function.)

`reshape()`

5.2.b Specify and fit a LCM for the repeated measures ANOVA (path model shown in Figure 5.6). Show that the results from the LCM are equivalent to those from Exercise 5.2.a.

5.2.c Specify and fit an unconstrained LCM for the `voelke.dat` data.

5.7 References & Further Readings

- Aiken, L. S., & West, S. G. (1991). *Multiple regression: Testing and interpreting interactions*. Newbury Park, CA: Sage.
- Bollen, K. A., & Curran, P. J. (2006). *Latent curve models: A structural equation perspective*. Hoboken, NJ: Wiley.
- Cohen, J., Cohen, P., West, S. G., & Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences* (3rd ed.). Mahwah, NJ: Lawrence Erlbaum.
- Muthén, B. O. (1989). Latent variable modeling in heterogeneous populations. *Psychometrika*, *54*, 557-585. doi: 10.1007/BF02296397
- Muthén, B. O., & Curran, P. J. (1997). General longitudinal modeling of individual differences in experimental designs: A latent variable framework for analysis and power estimation. *Psychological Methods*, *2*, 371-402. doi: 10.1037/1082-989X.2.4.371
- Preacher, K. J., Wichman, A. L., MacCallum, R. C., & Briggs, N. E. (2008). *Latent growth curve modeling*. Thousand Oaks, CA: Sage.
- Voelkle, M. C. (2007). Latent growth curve modeling as an integrative approach to the analysis of change. *Psychology Science*, *49*, 375-414.

6 | Models with Dichotomous Indicator Variables

Chapter Contents

6.1	Background	93
6.1.1	Item Factor Analysis	96
6.1.2	Parameterizing the Model	99
6.1.3	Item Response Theory	100
6.1.4	Relationship Between Item Response and Factor Analysis Models . .	104
6.2	Example: Dichotomous Indicator Variables	104
6.2.1	Item Response Theory Parameter Estimation	105
6.2.2	Item Factor Analysis Parameter Estimation	107
6.3	Summary	109
6.4	Writing the Results	110
6.5	Exercises	111
6.6	References & Further Readings	112

6.1 Background








































Thus far, the only LVMs I have covered have assumed that the indicator variables were continuous. When the indicators are categorical, the LVM needs some modification to account for the fact that the outcomes can only attain a small, finite number of values. If adjustments are not made and, instead, the parameters are estimated under the assumption that the indicator variables are continuous, the estimated covariances for the indicator variables tend to be spuriously high. If the number of possible response categories is > 4 and the distribution of the responses approximately follows a normal distribution (i.e., symmetrical with more responses in the middle than the extremes), then treating the data as being continuous will typically not substantially alter the parameter estimates, a robust estimator should be used (see Section A.1.1.1). Consequently, this chapter only deals with the most extreme case of categorical variables: those with only two response options—otherwise known as dichotomous variables.

Traditionally, LVMs with categorical outcomes were used for item-level analyses of psychometric instruments (e.g., extraversion scale, test of math knowledge), so they assumed that the indicator variables were unidimensional (i.e., only measure one LV) and locally independent. Multidimensional (i.e., more than one LV) models with dichotomous items are becoming more well-developed and used, but are beyond the scope of this chapter. Some of the references in Section 6.5 give more information about multidimensional models. Local independence is a complex topic. The gist of the assumption is that after controlling for differences in the LV, the items are unrelated to each other (i.e., no correlated errors).

When discussing dichotomous indicator variables, I use the terms *item* and *indicator variable* interchangeably.

Directions: The problems in this section are designed to measure reasoning ability without depending on verbal skills or educational background. Each problem consists of two groups of figures, labeled 1 and 2. These are followed by five lettered answer figures. For each problem you are to decide what characteristic it is that each of the figures in group 1 has and none of the figures in group 2 has. Then select the lettered answer figure that has this characteristic and blacken the space beneath the appropriate letter on the answer sheet.

Now look at the sample problems and their explanations below.

	I	2	A	B	C	D	E
I.	   	   	    				
II.	   	   	    				
III.	   	   	    				

(In sample problem I all the figures in group 1 are rectangles but none of the figures in group 2 are rectangles, so B is the answer. In sample problem II all the figures in group 1 include a dot, so D is the answer. The figures in group 1 of sample problem III are all white figures, so A is the answer.)

As soon as you understand the directions start to work on the problems in this section, marking your answers on your answer sheet.

Figure 6.1 Example LSAT *Figure Classification* items. Taken from Reese and Cotter (1994, p. 57). Used with permission.

The material in this chapter can be a bit difficult to understand, so instead of presenting an example towards the end of the chapter, I use a common example throughout to aid my explanations. The data for the example come from Bock and Lieberman (1970), and consist of 1000 responses to five items on the *Figure Classification* portion of the original version of the Law School Admission Test (LSAT). Some example *Figure Classification* items are shown in Figure 6.1, while the item response patterns are shown in Figure 6.2.

Historically, there have been two approaches to handling dichotomous data with a LVM. One approach is through item factor analysis (IFA) models, which take a similar approach to the data as has been discussed in the previous chapters. The second approach is through item response theory (IRT) models, which have been used for many decades with large-scale assessments, such as educational and licensing examination. There are a variety of IRT models, but I only discuss the two that map directly onto the IFA models. Although at one time IRT and IFA were considered different methods (or at least they were used that way), we now know that the main difference between them are the statistics' scales.

The situation of using dichotomous indicator variables for LVM is the same as with multiple regression when the outcome is dichotomous. With regression, the traditional linear

In Section 6.1.4, I show how to convert the estimates' scales.

Item 1	Item 2	Item 3	Item 4	Item 5	Frequency
0	0	0	0	0	3
0	0	0	0	1	6
0	0	0	1	0	2
0	0	0	1	1	11
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	3
0	0	1	1	1	4
0	1	0	0	0	1
0	1	0	0	1	8
0	1	0	1	1	16
0	1	1	0	1	3
0	1	1	1	0	2
0	1	1	1	1	15
1	0	0	0	0	10
1	0	0	0	1	29
1	0	0	1	0	14
1	0	0	1	1	81
1	0	1	0	0	3
1	0	1	0	1	28
1	0	1	1	0	15
1	0	1	1	1	80
1	1	0	0	0	16
1	1	0	0	1	56
1	1	0	1	0	21
1	1	0	1	1	173
1	1	1	0	0	11
1	1	1	0	1	61
1	1	1	1	0	28
1	1	1	1	1	298
0.92	0.71	0.55	0.76	0.87	

Numbers on the bottom row are the proportions of respondents who answered the item correctly.

Figure 6.2 Response patterns on five *Figure Classification* items ($n = 1000$).

The transformation is sometimes called a *link*.

model does not work with a dichotomous outcome, so the variable has to be transformed. The two common transformations are a logit (for logistic regression) or an inverse cumulative normal (for probit regression). These same two transformations work for LVMs with dichotomous indicator variables as well.

Item Factor Analysis (IFA)

Underlying Variable

An underlying variable is really a LV. I call it an underlying variable to minimize confusion with the LV of major interest.

6.1.1 Item Factor Analysis

Item factor analysis (IFA) models treat the responses to the dichotomous indicator variables as if they are very coarse (i.e., incomplete) representations of a continuous **underlying variable**, which is typically assumed to follow a normal distribution. The setup for these types of LVMs is as follows. Let y_c represent an observed dichotomous response (i.e., what is recoded in the data) and y represent the underlying continuous variable that y_c represents. I assume y follows a normal distribution with a mean of μ_y and variance of σ_y^2 . If higher scores on y reflect higher levels of the LV, then y_c and y are related through Equation (6.1).

$$y_c = \begin{cases} 0, & \text{if } y \leq \tau_c \\ 1, & \text{otherwise} \end{cases} \tag{6.1}$$

The τ_c term in Equation (6.1) is called the *threshold*. The threshold is the level of the LV needed to move from one level of the dichotomous response to another (i.e., move from incorrect to correct). If I make the very common assumption that $\mu_y = 0$ and $\sigma_y^2 = 1$ (i.e., y follows a standard normal distribution), then the threshold is the value of the standard normal distribution that makes the proportion of the distribution’s mass to the left of the threshold equal the proportion of incorrect (or 0) responses in the data.

`pnorm()`

The `pnorm()` function defaults to using a standard normal distribution.

Examining the *Figure Classification* items, the observed responses, y_c , are coded as 0 (incorrect) or 1 (correct). The threshold for item one, τ_1 , is -1.433, which is shown graphically in Figure 6.3. This threshold value indicates that the proportion of examinees who missed the item is approximately 0.076 (cf. Figure 6.2). I estimated the proportion using the `pnorm()` function (see Section 1.1.5.6), which returns the cumulative probability from a normal distribution.

```
pnorm(-1.43)

## [1] 0.0764
```

The value of τ_1 indicates that respondents with levels of y less than -1.433 (i.e., 1.433 standard deviations *below* the mean) will, on average, give an incorrect response, whereas individuals with levels of y greater than -1.433 will, on average, correctly answer the item.

Because the underlying variables are assumed to be standard normal, their covariances are correlations.

Tetrachoric Correlation

A similar approach is taken when there is more than one dichotomous indicator variable. Only now, not only are there thresholds to estimate, but correlations as well. Estimating the correlation requires the 2×2 table of response proportions for the items. Assuming both variables are coarse representations of normally distributed underlying variables, the resulting correlation is a **tetrachoric correlation**. This tetrachoric correlation approximates what the Pearson correlation would be between two dichotomous variables if they were recorded on their true continuous scale. Tetrachoric correlations cannot be calculated by hand with a

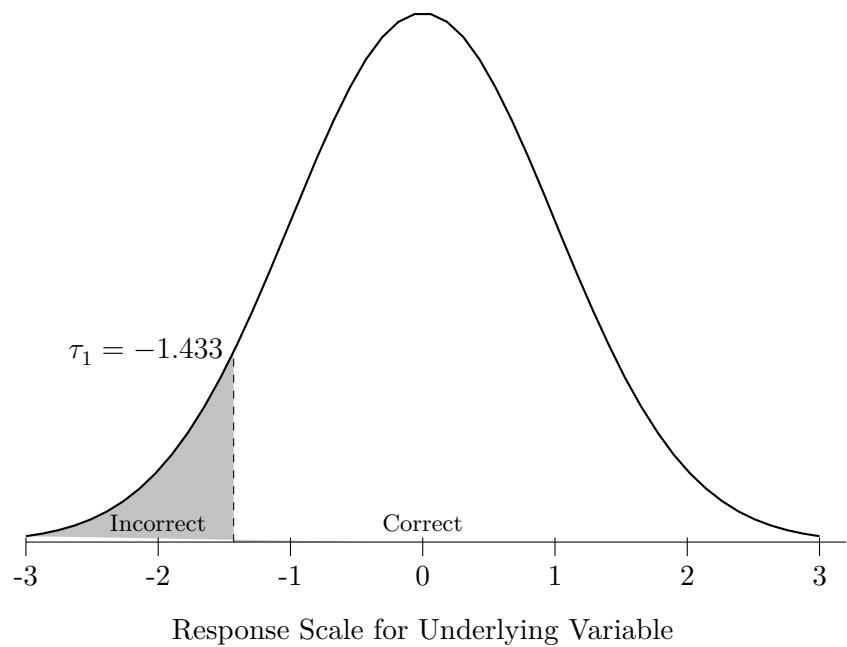


Figure 6.3 Underlying variable response distribution for the first *Figure Classification* item. τ_1 marks the threshold between observed responses (i.e., incorrect vs. correct).

simple formula. What is important to know is that the matrix of the tetrachoric correlations are what IFA programs use to estimate factor loadings.

Returning to the *Figure Classification* items, a 2×2 table of responses for items one and two is shown in Table 6.1. The bivariate distribution for this data, along with the individual item’s distributions (i.e., marginal distributions) and thresholds, is presented in Figure 6.4. Both representations of the data show that the majority of the respondents correctly answered both items. Very few respondents missed both items, however, at least in comparison to the number who correctly answered item one and incorrectly answered item two. Thus, the tetrachoric correlation between the items of 0.17 is not very large. For there to be a large tetrachoric correlation, the proportions who correctly answered both items and incorrectly answered both items must be much larger than the proportions who incorrectly answered only one of the items.

Table 6.1 Response Proportions and Frequencies for *Figure Classification* Items One and Two.

		Item 1	
		Incorrect	Correct
Item 2	Incorrect	0.031(31)	0.045(45)
	Correct	0.26(260)	0.664(664)

Numbers in parentheses are frequency counts.

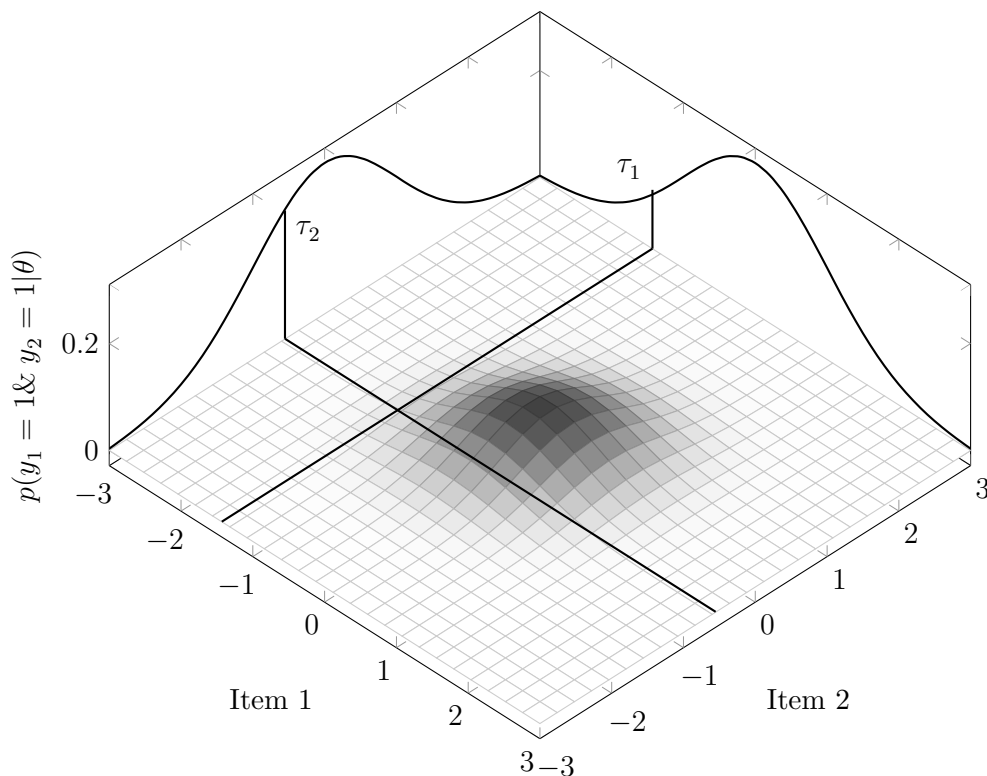


Figure 6.4 Bivariate and marginal underlying response distributions for *Figure Classification* items one and two (darker colors indicate higher density). The bivariate underlying response distribution, with a correlation of 0.17, represents the distribution of interest. The τ_1 and τ_2 parameters denote the thresholds for items 1 and 2, respectively, which are -1.433 and -0.55.

6.1.1.1 Parameter Estimation

Estimating parameters for IFA models typically involve using summary statistics (e.g., tetrachoric correlations, thresholds), so are sometimes called *limited information* methods. It may be tempting to use the tetrachoric correlations as the sample covariance matrix and then estimate the model parameters using traditional LVM estimation, pretending the correlations are covariances from continuous variables. This approach does not work well, as the standard errors will be wrong. Instead, limited information methods need to use a least squares estimator, which also necessitates the parameters are estimated using the cumulative normal distribution.

Some least squares estimators commonly used for IFA are generalized least squares (GLS), unweighted least squares (ULS), weighted least squares (WLS), or diagonally weighted least squares (DWLS). More recently, LV programs have begun to add corrected version of the DWLS estimator that produces standard errors and a χ^2 fit statistic that are robust to the fact that the indicator variables do not follow a normal distribution (see Appendix A for more on the χ^2 statistic). Corrections that adjust for the means (WLSM) and corrections that adjust for the means and variances (WLSMV) are two of the better options for LVMs with dichotomous indicator variables, with WLSMV usually performing the best. Some references in Section 6.5 explain the differences between these estimators in greater detail.

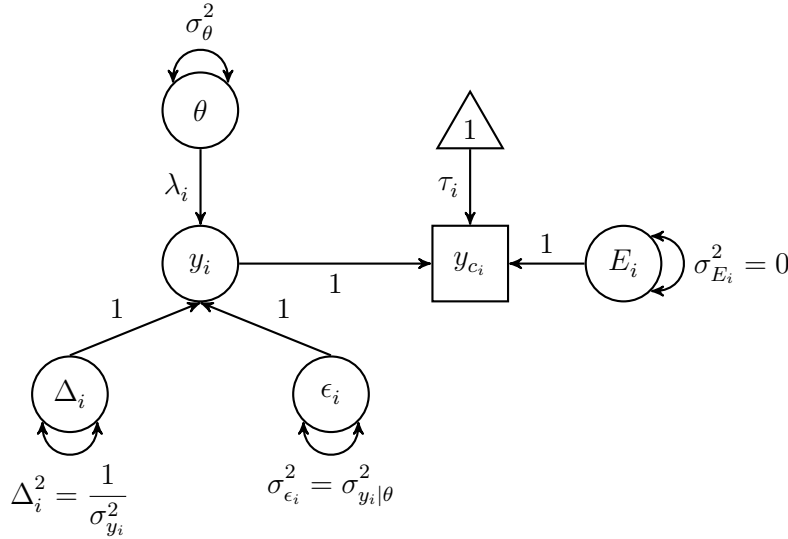


Figure 6.5 Conceptual diagram for the underlying variable representation of a item factor analysis model with a single dichotomous indicator, i .

6.1.2 Parameterizing the Model

There is not a proper path model for LVMs with dichotomous indicator variables. Nonetheless, Figure 6.5 contains a *conceptual* diagram for a LVM with a single dichotomous indicator variable. In this model, y_{ci} is the observed response to indicator variable i , τ_i is that item's threshold, y_i is the underlying variable for that item, λ_i is the factor loading, θ is the LV (which is same for any item on the test containing item i , so it has no subscript), and $\sigma_{y_i}^2$ is the variance of y_i . $\sigma_{E_i}^2$ is the error variance of y_{ci} (i.e., what y_i does not explain of y_{ci}), but from Equation (6.1) I know that y_c is a deterministic function of y , meaning it is measured perfectly and $\sigma_{E_i}^2 = 0$. Δ_i is a scaling factor for σ_{y_i} , defined to be $\frac{1}{\sigma_{y_i}}$ (i.e., the reciprocal of the underlying variable's standard deviation). Finally, $\sigma_{\epsilon_i}^2$ is the error variance of y_i (i.e., what θ does not explain of y_i), and is defined to be $\sigma_{\epsilon_i}^2 = \frac{1}{\Delta_i^2} - \lambda^2 \sigma_{\theta}^2$.

A problem with the model in Figure 6.5 is that it is underidentified. To make the model identified, I have to constrain some parameters. There are two sets of parameter constraints (i.e., parameterizations) for IFA models: marginal and conditional. They are mathematically equivalent (i.e., result in the same model fit), but estimate different aspects of the model.

6.1.2.1 Marginal Parameterization

The **marginal parameterization** (also called the delta or standardized parameterization) is currently the only way `lavaan` handles dichotomous data, although there are plans to implement other parameterizations. It gets its name from the fact that the marginal (unconditional) distribution of y_i is standardized. It is the most common parameterization for IFA models.

*Marginal
Parameterization*

In this parameterization, $\sigma_{y_i}^2$ is constrained to be 1.0, which makes $\Delta_i^2 = 1.0$, too. $\sigma_{\epsilon_i}^2$ is not estimated directly, but using the definition of $\sigma_{\epsilon_i}^2$, and substituting $\sigma_{y_i}^2 = \Delta^2 = 1.0$,

returns the results in Equation (6.2).

$$\sigma_{\epsilon_i}^2 = \frac{1}{\Delta^2} - \lambda^2 \sigma_\theta^2 = \frac{1}{1} - \lambda^2 \sigma_\theta^2 = 1 - \lambda_i^2 \sigma_\theta^2 \quad (6.2)$$

6.1.2.2 Conditional Parameterization

Conditional Parameterization

In the **conditional parameterization** (also called the theta or unstandardized parameterization), the underlying variable's error variance, $\sigma_{\epsilon_i}^2$, is constrained to be 1.0. It gets its name from the fact that $\sigma_{\epsilon_i}^2$, which is conditional on θ (i.e., $\sigma_{\epsilon_i}^2 = \sigma_{y_i|\theta}^2$), is standardized. The thresholds are also constrained, but typically this is done by finding their values from a prior run of the model using the marginal parameterization.

Using the definition of $\sigma_{\epsilon_i}^2$ and setting it equal 1.0 returns the results in Equation (6.3).

$$\sigma_{\epsilon_i}^2 = 1.0 = \frac{1}{\Delta^2} - \lambda^2 \sigma_\theta^2 \quad (6.3)$$

A little algebraic manipulation of Equation (6.3) produces Equation (6.4).

$$\Delta = \frac{1}{\sqrt{\lambda^2 \sigma_\theta^2 + 1}} \quad (6.4)$$

Substituting the definition of Δ , $\frac{1}{\sigma_{y_i}}$, into Equation (6.4) produces Equation (6.5).

$$\sigma_{y_i} = \sqrt{\lambda^2 \sigma_\theta^2 + 1} \quad (6.5)$$

6.1.2.3 Scaling the Latent Variable

As with LVs using continuous indicator variables, the LV in IFA models must be scaled, too (see Section 3.2.1.2). This can be done using either a marker variable or by standardizing the LV. With dichotomous indicator variables, the marker variable method requires constraining one item's threshold, τ , to be 0.0 and loading, λ , to be 1.0. In return for these constraints, the LV's mean and variance are estimated. To use the standardized LV method, constrain the LV's mean to 0.0 and variance to 1.0, which produces estimates for all the loadings and thresholds. The standardized LV method is the most common approach used by IRT programs and many IFA programs as well.

6.1.3 Item Response Theory

Item Response Theory (IRT)

Item Characteristic Curve

Item response theory (IRT) models have largely been used with large-scale assessment, such as those done in education and for professional licensure. Thus, the terminology tends to focus on item responses and the parameter estimates are scaled such that, when placed in an appropriate function, they give the probability of endorsing a given response (e.g., correct response) based on a respondent's level on the LV. IRT analyses have a longstanding tradition of using graphical representations of their output, specifically the **item characteristic curve** (ICC).

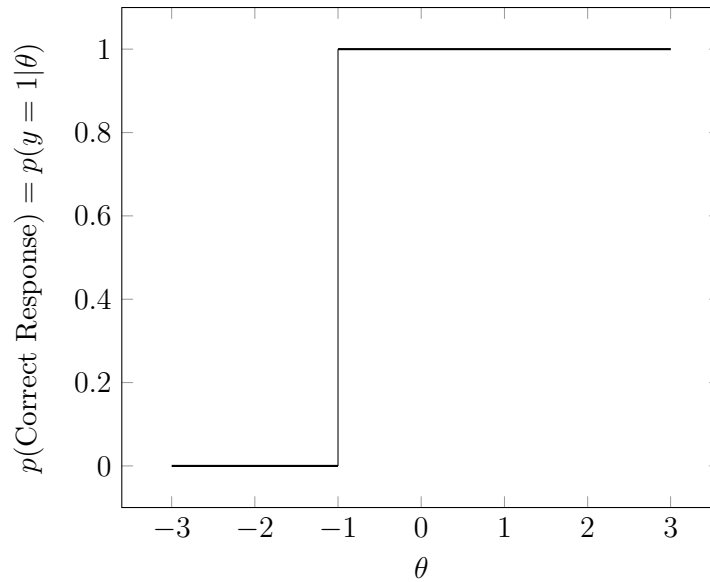


Figure 6.6 Item characteristic curve of a Guttman scale item whose location, b , is -1. The item works the best at differentiating respondents whose ability (θ) is -1.0.

The ICC graphically shows the probability endorsing a given response to an item as a function of the LV. There are two requirements for the function creating a ICC.

1. The LV, θ , is unbounded; thus, the ICC works for any level of the LV.
2. The outcome is in the probability metric, p , so is bounded between 0.0 and 1.0.

One of the first ICC functions was given by Louis Guttman (1950) and is shown in Equation (6.6).

$$p(y = 1|\theta) = \begin{cases} 1, & \text{if } \theta > b \\ 0, & \text{if } \theta < b \end{cases} \quad (6.6)$$

where θ is the LV, and b is the **item location** (i.e., the place on the latent variable's scale where the item best functions). According to this function, the probability of responding correctly to an item, $p(y = 1)$, conditioned on the respondent's level of the LV, θ , is 1.0 if the level of θ is greater than the item's location, but is 0.0 if the level of θ is less than the item's location. An example of an ICC based on Equation (6.6) is given in Figure 6.6.

Item Location

Notice that θ and b are *directly* compared in Equation (6.6). For this comparison to have any meaning, the items and the LV must use the same scale. Thus, these models estimate both item (e.g., b) and person (e.g., θ) parameters and do so by placing b and θ on the same metric.

The major problem with Guttman's function is that it assumes items are perfect measures of their constructs. Anyone who has ever designed a test before knows that this assumption is typically not tenable. Subsequently, IRT scholars have used two alternative functions instead: cumulative normal distribution and cumulative logistic distribution.

The scale of the cumulative normal distribution (sometimes called a *normal ogive*) is in probability units, thus it is bounded by 0.0 and 1.0. Moreover, it monotonically increases, meaning that as the magnitude of the input increases, the magnitude of the output never

ogive is synonymous with a cumulative distribution function.

decreases.¹The major problem with this function is that it requires advanced integral calculus to be able to use it. With the computational speed of modern computers, this is not a major issue anymore, but it was in the early days of IRT development. Consequently, IRT scholars turned to a similarly behaving function: the cumulative logistic distribution. A plot of a cumulative logistic distribution is shown in Equation (6.7).

$$L(Z) = \frac{\exp(Z)}{1 + \exp(Z)} = \frac{1}{\exp(-Z) + 1} \quad (6.7)$$

Euler's number is approximately 2.718 and is the base of the natural logarithm.

where $\exp()$ is the exponentiate function with Euler's number as the base.

Like the cumulative normal distribution, the cumulative logistic distribution's metric is in probability units and is monotonically increasing. In fact, the results from the cumulative normal and cumulative logistic distributions can be made almost indistinguishable by multiplying the logistic function's kernel (i.e., the exponentiated value) by 1.7.² The cumulative logistic distribution has an advantage, however, as it is much simpler to use.

For more information on IRT models with more than two outcomes, see Section 6.5.

The two parameter logistic (2PL) IRT model, shown in Equation (6.8), is one of the most commonly used IRT models for dichotomous items, and generalizes relatively well for items with more than two outcomes. It gets its name because it estimates two item parameters (location and discrimination) and uses the cumulative logistic distribution function.

$$p(y_{ij} = 1|\theta_i) = \frac{1}{\exp(-Da_j[\theta_i - b_j]) + 1} \quad (6.8)$$

where y_{ij} is the i th person's response to item j , θ_i is the i th person's level on the latent variable, b_j is the j th item's location, a_j is the j th item's discrimination, and D is the 1.7 constant.

Item Discrimination

In the 2PL model, the item's location (b) takes on the additional meaning of being the level of the LV needed to have a 0.50 probability of correctly answering the item (i.e., scoring 1.0 vs. 0.0). The **item discrimination** (a) is a measure of how well an item represents the LV.

The 2PL IRT estimates for the five *Figure Classification* items are shown in the second and third column of Table 6.2. All the items are rather easy, as the location estimates are < 0.0 , which is the mean of the LV. Moreover, all the items measure the construct of interest relatively similarly, as the discrimination estimates are fairly homogeneous. The 2PL ICCs for these items are shown in Figure 6.7.

To speed estimation time, most current IRT program re-parameterize Equation (6.8) into a slope-intercept format. The re-parameterized version of Equation (6.8) is given in Equation (6.9).

You can think of α and β estimates as being an unstandardized factor loadings and intercepts.

¹For items that have a negative relationship with the LV, the function monotonically decreases (i.e., as the magnitude of the input increases, the magnitude of the output never increases).

²Camilli (1994) gives a brief explanation of how the value 1.7 was derived.

Table 6.2 Item Response Theory Parameter Estimates for the *Figure Classification* Items.

Item	b	a	β	α
1	-3.36	0.83	2.77	0.83
2	-1.37	0.72	0.99	0.72
3	-0.28	0.89	0.25	0.89
4	-1.86	0.69	1.28	0.69
5	-3.12	0.66	2.05	0.66

b : location; a : discrimination; β : intercept; α : slope.

$$f(\alpha_j \theta_i + \beta_j) \quad (6.9)$$

where α_j is the j th item's slope, β_j is the j th item's intercept, and $f()$ is the cumulative logistic or normal distribution.

For the two-parameter model, the slope is equivalent to the discrimination parameter, but the intercept is not equivalent to the location. To convert the intercept back into a location parameter, use the formula given in the bottom of Table 6.3. The IRT slope and intercept estimates for the six *Figure Classification* items are given in the last two columns of Table 6.2.

6.1.3.1 Parameter Estimation

IRT software usually estimate the model parameters by using all the individual respondents' item responses (or response patterns). These are called *full information* estimates. Typically, this is done using a maximum likelihood estimator. The field of parameter estimation with IRT models has a relatively large literature base. The interested reader should consult the the IRT sources listed in Section 6.5.

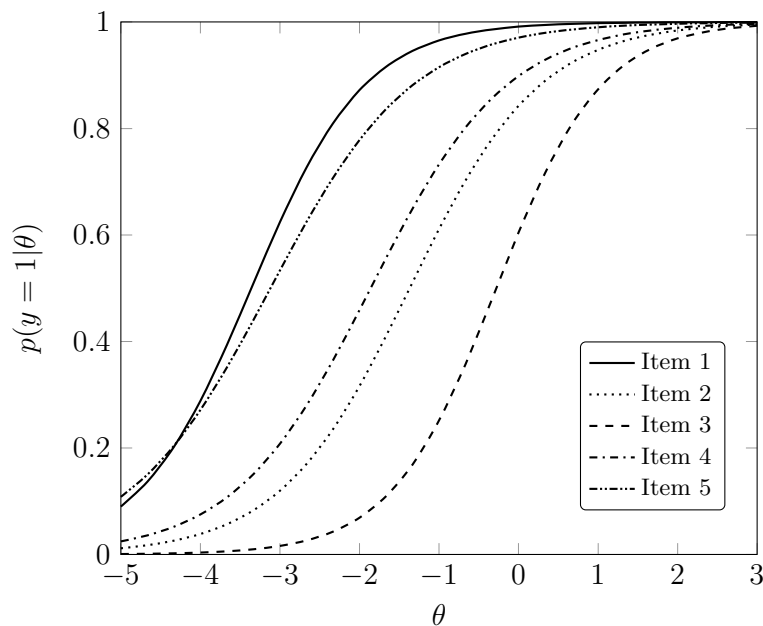


Figure 6.7 Item characteristic curves for the *Figure Classification* items.

Table 6.3 Conversion Formulae for Different Item Factor Analysis Parameterizations to Item Response Theory Parameters.

		Latent Variable (θ) Identification	
		Marker Variable	Standardized
Parameterization	Marginal	$\alpha = \frac{\lambda\sqrt{\sigma_\theta^2}}{\sqrt{1 - \lambda^2\sigma_\theta^2}}$	$\alpha = \frac{\lambda^2}{\sqrt{1 - \lambda^2}}$
		$\beta = \frac{-(\tau - \lambda\mu_\theta)}{\sqrt{1 - \lambda^2\sigma_\theta^2}}$	$\beta = \frac{-\tau}{\sqrt{1 - \lambda^2}}$
	Conditional	$\alpha = \lambda\sqrt{\sigma_\theta^2}$	$\alpha = \lambda$
		$\beta = -(\tau - \lambda\mu_\theta)$	$\beta = -\tau$

Some programs estimate $-\tau$ instead of τ . For those estimates, use τ instead of $-\tau$ in the conversion formulae. To obtain b , use the conversion: $b = -\beta/\alpha$

IRT parameter estimates can be estimated using either the cumulative logistic or cumulative normal distributions, although the cumulative logistic is more efficient so it is the most common. When using the cumulative logistic distribution, some programs include the $D = 1.7$ constant and some do not, but most are explicit in stating whether or not they use it.

6.1.4 Relationship Between Item Response and Factor Analysis Models

Parameter estimates from IRT and IFA models can be converted to each others' metrics. I present two tables to facilitate this conversion. Table 6.3 presents formulae to calculate IRT slope and intercept from IFA estimates, using either model parameterization and either LV scaling discussed in Section 6.1.2. Table 6.4 presents formulae to convert IFA parameter estimates to IRT discrimination and location parameters, and vice versa, for logistic or normal ogive scales. When converting from parameters estimated from a logistic distribution to a normal distribution (or vice versa), the resulting coefficients will be much rougher than when converting estimates using the same distribution.

6.2 Example: Dichotomous Indicator Variables

For the example, I demonstrate how I analyzed the LSAT *Figure Classification* data that I used throughout the chapter. The data are available in the `psych` package and the dataset is named `lsat6`. The items are named *Q1-Q5*.

```
library(psych)
head(lsat6)

##      Q1 Q2 Q3 Q4 Q5
## [1,]  0  0  0  0  0
## [2,]  0  0  0  0  0
```

Table 6.4 Item Response and Item Factor Analysis Conversion Formulae.

	a and λ	b^1 and τ
IFA to IRT	$a = \left(\frac{\lambda}{\sqrt{1 - \lambda^2}} \right) D$	$b = \frac{\tau}{\lambda}$
IRT to IFA	$\lambda = \frac{a/D}{\sqrt{1 + (a/D)^2}}$	$\tau = \frac{b(a/D)}{\sqrt{1 + (a/D)^2}}$

The conversions assume θ is standardized for all item parameter estimates. Some programs estimate $-\tau$ instead of τ . For those estimates, use $-\tau$ instead of τ in the conversion formulae.

¹ To obtain β , use the conversion: $\beta = -\alpha \times b$.

$$D = \begin{cases} 1.7, & \text{if IRT parameters estimated using the logistic distribution} \\ 1, & \text{otherwise} \end{cases}$$

```
## [3,] 0 0 0 0 0
## [4,] 0 0 0 0 1
## [5,] 0 0 0 0 1
## [6,] 0 0 0 0 1
```

To estimate the tetrachoric correlations among the items, use the `psych` package's `tetrachoric()` function. The function returns both the tetrachoric correlations and the thresholds.

`tetrachoric()`

```
# Tetrachoric correlations
tetrachoric(lsat6)
```

```
## Call: tetrachoric(x = lsat6)
## tetrachoric correlation
##      Q1  Q2  Q3  Q4  Q5
## Q1 1.00
## Q2 0.17 1.00
## Q3 0.23 0.19 1.00
## Q4 0.11 0.11 0.19 1.00
## Q5 0.07 0.17 0.11 0.20 1.00
##
## with tau of
##      Q1  Q2  Q3  Q4  Q5
## -1.43 -0.55 -0.13 -0.72 -1.13
```

6.2.1 Item Response Theory Parameter Estimation

There are multiple packages in R that estimate IRT parameters.³ I use the `ltm()` function in the `ltm` package for estimation with the logistic distribution and the `irt.fa()` function in the `psych` package for estimation with the normal distribution.

The `ltm()` function estimates parameters for the 2PL model using maximum likelihood. By default, it estimates β and α , but using the `IRT.param=TRUE` argument returns b and a instead. Its main argument is the model to estimate, which is the dataset's name followed by the tilde and `z1`.

`ltm()`

`z1` represents one standardized latent variable.

³See <http://cran.cc.uoc.gr/web/views/Psychometrics.html>

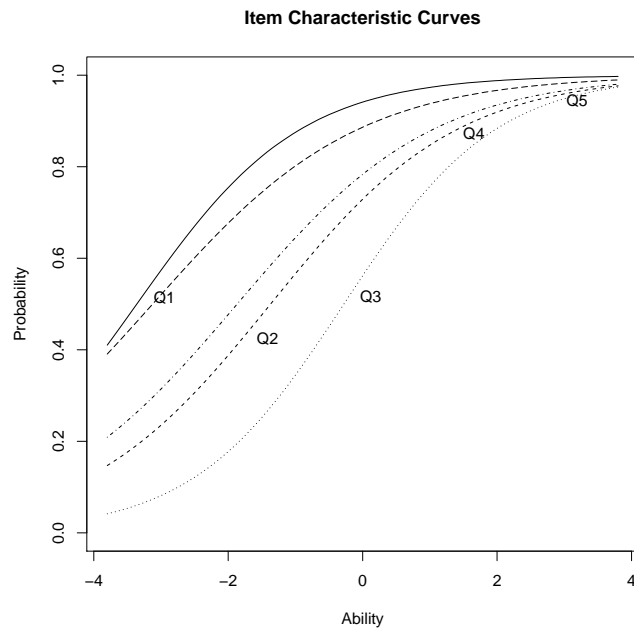


Figure 6.8 Item characteristic curves for `lsat6` data using the `ltm` package's `plot()` function.

```
library(ltm)
# discrimination and difficulty
lsat.IRT <- ltm(lsats6 ~ z1, IRT.param = TRUE)
# slope and intercept
lsat.SI <- ltm(lsats6 ~ z1, IRT.param = FALSE)
```

To return the parameter estimates, use the `coef()` function. For the slope-intercept output, $z1 = \alpha$ and $(\text{Intercept}) = \beta$.

```
# discrimination and location
coef(lsat.IRT)
```

```
##      Dffclt Dscrmn
## Q1  -3.36  0.825
## Q2  -1.37  0.723
## Q3  -0.28  0.890
## Q4  -1.87  0.689
## Q5  -3.12  0.657
```

```
# slope and intercept
coef(lsat.SI)
```

```
##      (Intercept)      z1
## Q1           2.773 0.825
## Q2           0.990 0.723
## Q3           0.249 0.890
## Q4           1.285 0.689
## Q5           2.054 0.657
```

The `plot()` function plots the ICCs using an object created by the `ltm()` function. The ICCs for the *Figure Classification* items are shown in Figure 6.8.

```
# item characteristic curves
plot(lsat.IRT)
```

The `irt.fa()` function takes the item data as the main argument. By default it returns some item plots, so to turn those off I used the `plot=FALSE` argument.

```
lsat.NO <- irt.fa(lsat6, plot = FALSE)
```

To obtain the IRT parameter estimates I extract them from the results. I do this by appending `$irt` onto the name of the object where I stored the results.

```
lsat.NO$irt

## $difficulty
## $difficulty[[1]]
##      Q1      Q2      Q3      Q4      Q5
## -1.550 -0.602 -0.152 -0.771 -1.189
##
##
## $discrimination
##      MR1
## Q1 0.413
## Q2 0.445
## Q3 0.555
## Q4 0.398
## Q5 0.337
```

Even though the output's labels say discrimination and *difficulty*, the resulting coefficients are really slopes and intercepts, with the intercept being multiplied by -1 to make them the same sign as the location estimates.

If I append `$fa` onto the name of the `irt.fa()` function's output object, **R** returns the loadings.

```
lsat.NO$fa

## Factor Analysis using method = minres
## Call: fa(r = r, nfactors = nfactors, n.obs = n.obs)
## Standardized loadings (pattern matrix) based upon correlation matrix
##      MR1  h2  u2
## Q1 0.38 0.15 0.85
## Q2 0.41 0.17 0.83
## Q3 0.49 0.24 0.76
## Q4 0.37 0.14 0.86
## Q5 0.32 0.10 0.90
##
##
..<Output Omitted>..
```

6.2.2 Item Factor Analysis Parameter Estimation

lavaan estimates λ and τ from the *marginal* underlying variable approach. The `|` model specification syntax defines a threshold. The basic threshold specification is: `latent variable | label*t1`.

See Table 2.2 for **lavaan** model specification syntax.

To make sure **lavaan** knows what variables are dichotomous, I name all the variables in the model that are dichotomous within the `ordered` argument. When using dichotomous indicator variables, **lavaan** automatically uses the *WLSMV* estimator, with the alternative

being robust ULS estimation (ULSMV).⁴ As I want the LV to be standardized, I use the `std.lv=TRUE` argument.

To convert the IFA results to IRT parameters, I can either make the calculations after obtaining the IFA results or have `lavaan` do the calculations as part of the parameter estimation by labeling the loadings and thresholds, and then defining the parameters of interest using the `:=` syntax and the conversion formulae in Table 6.3 and Table 6.4. I do not show the `lavaan` output for space considerations, but in Table 6.5, I compare the results from the different estimation methods and scales. Within a distribution, the results are almost identical, and between distributions they are fairly close.

```
library(lavaan)
twoP.model<-'
# loadings
Theta =~ l1*Q1 + l2*Q2 + l3*Q3 + l4*Q4 + l5*Q5
# thresholds
Q1 | th1*t1
Q2 | th2*t1
Q3 | th3*t1
Q4 | th4*t1
Q5 | th5*t1
# convert loadings to slopes (normal)
alpha1.N := (l1)/sqrt(1-l1^2)
alpha2.N := (l2)/sqrt(1-l2^2)
alpha3.N := (l3)/sqrt(1-l3^2)
alpha4.N := (l4)/sqrt(1-l4^2)
alpha5.N := (l5)/sqrt(1-l5^2)
# convert thresholds to intercepts (normal)
beta1.N := (-th1)/sqrt(1-l1^2)
beta2.N := (-th2)/sqrt(1-l2^2)
beta3.N := (-th3)/sqrt(1-l3^2)
beta4.N := (-th4)/sqrt(1-l4^2)
beta5.N := (-th5)/sqrt(1-l5^2)
# convert intercepts to locations (normal)
loc1 := -beta1.N/alpha1.N
loc2 := -beta2.N/alpha2.N
loc3 := -beta3.N/alpha3.N
loc4 := -beta4.N/alpha4.N
loc5 := -beta5.N/alpha5.N
# convert loadings to slopes (logistic)
alpha1.L := (l1)/sqrt(1-l1^2)*1.7
alpha2.L := (l2)/sqrt(1-l2^2)*1.7
alpha3.L := (l3)/sqrt(1-l3^2)*1.7
alpha4.L := (l4)/sqrt(1-l4^2)*1.7
alpha5.L := (l5)/sqrt(1-l5^2)*1.7
# convert thresholds to locations (logistic)
loc1.L := th1/l1
loc2.L := th2/l2
loc3.L := th3/l3
loc4.L := th4/l4
loc5.L := th5/l5
# convert locations to intercepts (logistic)
beta1.L := (-alpha1.L)*loc1.L
beta2.L := (-alpha2.L)*loc2.L
```

⁴Pairwise maximum likelihood (PML) is another available estimator, although as of version 0.5.14, the estimator is “still under development.”

Table 6.5 Parameter Estimates from Item Response Theory and Item Factor Analysis Parameterizations of the `lsat6` Data.

Item	a	b	α	β	λ	τ
ltm ltm() (Logistic)						
Q1	0.83	-3.36	0.83	2.77	<i>0.39</i>	<i>-1.47</i>
Q2	0.72	-1.37	0.72	0.99	<i>0.36</i>	<i>-0.53</i>
Q3	0.89	-0.28	0.89	0.25	<i>0.41</i>	<i>-0.13</i>
Q4	0.69	-1.87	0.69	1.28	<i>0.35</i>	<i>-0.70</i>
Q5	0.66	-3.12	0.66	2.05	<i>0.34</i>	<i>-1.13</i>
psych irt.fa() (Normal)						
Q1	<i>0.41</i>	<i>-3.78</i>	0.41	-1.55	0.38	-1.43
Q2	<i>0.44</i>	<i>-1.36</i>	0.44	-0.60	0.41	-0.55
Q3	<i>0.56</i>	<i>-0.27</i>	0.56	-0.15	0.49	-0.13
Q4	<i>0.40</i>	<i>-1.93</i>	0.40	-0.77	0.37	-0.72
Q5	<i>0.34</i>	<i>-3.50</i>	0.34	-1.19	0.32	-1.13
lavaan cfa()						
Normal						
Q1	<i>0.42</i>	<i>-3.68</i>	<i>0.42</i>	<i>1.56</i>	0.39	-1.43
Q2	<i>0.43</i>	<i>-1.39</i>	<i>0.43</i>	<i>0.60</i>	0.40	-0.55
Q3	<i>0.53</i>	<i>-0.28</i>	<i>0.53</i>	<i>0.15</i>	0.47	-0.13
Q4	<i>0.41</i>	<i>-1.90</i>	<i>0.41</i>	<i>0.77</i>	0.38	-0.72
Q5	<i>0.36</i>	<i>-3.29</i>	<i>0.36</i>	<i>1.20</i>	0.34	-1.13
Logistic						
Q1	<i>0.72</i>	<i>-3.68</i>	<i>0.72</i>	<i>2.64</i>	—	—
Q2	<i>0.74</i>	<i>-1.39</i>	<i>0.74</i>	<i>1.02</i>	—	—
Q3	<i>0.91</i>	<i>-0.28</i>	<i>0.91</i>	<i>0.26</i>	—	—
Q4	<i>0.69</i>	<i>-1.90</i>	<i>0.69</i>	<i>1.31</i>	—	—
Q5	<i>0.62</i>	<i>-3.29</i>	<i>0.62</i>	<i>2.04</i>	—	—

Italicized coefficients were converted using formulae in Table 6.3 or Table 6.4. The β estimates from `irt.fa()` are $-1(\beta)$.

```

beta3.L := (-alpha3.L)*loc3.L
beta4.L := (-alpha4.L)*loc4.L
beta5.L := (-alpha5.L)*loc5.L
'

twoP.fit <- cfa(twoP.model, data=data.frame(lsat6), std.lv=TRUE,
ordered=c("Q1","Q2","Q3","Q4","Q5"))

# print IFA results as well as the specified conversions
summary(twoP.fit, standardized = TRUE)

```

6.3 Summary

In this chapter, I introduced item factor analysis (IFA) and item response theory (IRT) models—LVMs that use dichotomous indicator variables. The IRT model I showed was the

two-parameter logistic (2PL) model, which estimates location and discrimination (or intercept and slope) parameters. I showed the basic idea behind the IFA model (i.e., underlying variables), and then showed two different parameterizations of the model: marginal and conditional. I then gave formulae for how to convert the IFA parameter estimates to the IRT scale, and vice versa. I concluded the chapter by demonstrating how to estimate IRT and IFA parameters in **R** using the `ltm`, `psych`, and `lavaan` packages, and then empirically showed the equivalency of the parameter estimates.

One question that may arise from the reader is: If the IRT and IFA models return such similar results, then which one is the best to choose? Forero and Maydeu-Olivares (2009) did an extensive simulation study comparing the methods and found that the IRT model (with its maximum likelihood estimator) is better when sample sizes are small ($n \leq 200$). Otherwise, the difference between them is negligible, so they suggest choosing the most efficient (i.e., quickest) estimator. The most efficient estimator is often the robust weighted least-squares (WLSMV), which is used with IFA.

In the same study, Forero and Maydeu-Olivares (2009) indicated situations that make a factor analysis of dichotomous variables likely to fail (e.g., unable to estimate the parameters or return impossible values for the estimates). Some of the situations to avoid are: (a) having a small number of indicator variables for a LV; (b) having indicator variables with low loadings (i.e., $\lambda < 0.40$) or slopes ($\alpha < 0.74$); (c) having indicator variables with minimal difference in response (i.e., everyone gets the items correct or incorrect); and (d) having a small sample size ($n < 200$).

6.4 Writing the Results

When writing the results from a LVM using dichotomous response, you should include much of the same information that is reported when analyzing a LVM with continuous indicator variables (see Section 3.6). The parameter estimates and standard errors should be presented in a table, where the first column is the item name (e.g., item 1, items 2) or item content (e.g., $4 \times 5 = ?$), and the items are either sorted by the order they were presented to the respondents or by loading/discrimination/slope magnitude.

Either in the body of the text or in a table of parameter estimates, report the following information: (a) sample size used to estimate the parameters; (b) the item response format (e.g., multiple choice, true/false); (c) the estimation method (e.g., maximum likelihood, robust weighted least squares) and distribution (e.g., logistic, normal); (d) the IFA parameterization (e.g., marginal, conditional) or IRT parameterization (e.g., 2PL); (e) the program used to estimate the parameters (if using **R**, state which packages); (f) any additional constraints added to the model; and (g) interpretations of the parameter estimates.

There should be some discussion of model fit (see Appendix A). If only one model was used then an in-text description is fine, but if more than one model is fit then present the results in a table. The R^2 estimates for dichotomous indicators are not the typical R^2 estimates (they are pseudo- R^2 estimates), so standardized loadings are a better effect size measure. The standardized loadings should be included either in a table or in a path model-type diagram;

Table 6.6 *British Social Attitudes Survey* Items Concerning Abortion for Exercise 6.1.

Item Number	Item Stem
1	The woman decides on her own that she does not wish to have a child.
2	The couple agree that they do not wish to have a child.
3	The woman is not married and does not wish to marry the man.
4	The couple cannot afford any more children.

including the thresholds in either is optional. An alternative to a table or path model is to present the ICCs. If there were not very many items analyzed (≤ 10), all the ICCs can be placed on a single figure (e.g., Figure 6.7) for a typical range of θ , usually $-3 \leq \theta \leq +3$. If there were many items analyzed, then group the items (e.g., ones that worked poorly vs. well, ones that were easy vs. difficult) and show multiple figures with the grouped ICCs.

To increase replicability, the manuscript should present descriptive statistics for all the items. Include: (a) the proportion of the respondents who correctly responded to an item, (b) the tetrachoric correlations among the items, and (c) the **R** syntax used for the analysis in the body of the manuscript or in an appendix.

6.5 Exercises

- 6.1 Bartholomew, Steele, Galbraith, and Moustaki (2008) analyzed four items from the *British Social Attitudes Survey* concerning abortion. The item responses from 379 respondents are available in the **Abortion** data from the **ltm** package. The items are given in Table 6.6. For each items, respondents were to indicate *yes* (1) or *no* (0) on whether abortion should be allowed.
 - 6.1.a Enter the data into **R** and rename the items *I1-I4*.
 - 6.1.b Find the proportion who endorsed each item (i.e., the mean score).
 - 6.1.c Estimate item discrimination and location using the cumulative logistic and cumulative normal distributions.
 - 6.1.d Plot the item characteristic curves.
 - 6.1.e Estimate item slope and intercept using the cumulative logistic and cumulative normal distributions.
 - 6.1.f Conduct an item factor analysis with one latent variable.
- 6.2 Beaujean and Sheng (2010) conducted an IRT analysis of the ten-item vocabulary test from the *General Social Survey*. Data from the respondents with responses to all 10 items ($n = 2943$) from the 2000 decade group are available on the book's website as a space-delimited file (**gss2000.dat**), and the items are named *word.a-word.j*.
 - 6.2.a Import the data into **R** and name it **gss.data**.
 - 6.2.b Conduct an item-level confirmatory factor analysis with one latent variable.
 - 6.2.c Convert the factor loading and threshold estimates to item location and discrimination parameters. What are the easiest and most difficult items?
 - 6.2.d Compare your conversion results to those from estimating the parameters directly using **ltm()**.

6.6 References & Further Readings

- Baker, F. B., & Kim, S.-H. (2004). *Item response theory: Parameter estimation techniques* (2nd ed.). New York, NY: Marcel Dekker.
- Bartholomew, D. J., Knott, M., & Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach* (3rd ed.). New York, NY: John Wiley & Sons.
- Bartholomew, D. J., Steele, F., Galbraith, J., & Moustaki, I. (2008). *Analysis of multivariate social science data* (2nd ed.). Boca Raton, FL: Chapman and Hall/CRC.
- Beauducel, A., & Herzberg, P. Y. (2006). On the performance of maximum likelihood versus means and variance adjusted weighted least squares estimation in CFA. *Structural Equation Modeling: A Multidisciplinary Journal*, 13, 186-203. doi: 10.1207/s15328007sem1302_2
- Beaujean, A. A., & Sheng, Y. (2010). Examining the Flynn effect in the general social survey vocabulary test using item response theory. *Personality and Individual Differences*, 48, 294-298. doi: 10.1016/j.paid.2009.10.019
- Bock, R. D., & Lieberman, M. (1970). Fitting a response model for n dichotomously scored items. *Psychometrika*, 35, 179-197. doi: 10.1007/BF02291262
- Camilli, G. (1994). Teacher's corner: Origin of the scaling constant $d = 1.7$ in item response theory. *Journal of Educational and Behavioral Statistics*, 19, 293-295. doi: 10.3102/10769986019003293
- Embretson, S. E., & Reise, S. P. (2000). *Item response theory for psychologists*. Mahwah, NJ: Erlbaum.
- Finney, S. J., & DiStefano, C. (2013). Non-normal and categorical data in structural equation modeling. In G. R. Hancock & R. O. Mueller (Eds.), *Structural equation modeling: A second course* (2nd ed.). Charlotte, NC: Information Age Publishing.
- Flora, D. B., & Curran, P. J. (2004). An empirical evaluation of alternative methods of estimation for confirmatory factor analysis with ordinal data. *Psychological Methods*, 9, 466-491. doi: 10.1037/1082-989X.9.4.466
- Forero, C. G., & Maydeu-Olivares, A. (2009). Estimation of IRT graded response models: Limited versus full information methods. *Psychological Methods*, 14, 275-299. doi: 10.1037/a0015825
- Guttman, L. (1950). The basis for scalogram analysis. In S. A. Stouffer, L. Guttman, E. A. Suchman, P. F. Lazarsfeld, S. A. Star, & J. A. Clausen (Eds.), *Measurement and prediction* (pp. 60-90). Princeton, NJ: Princeton University Press.
- Kamata, A., & Bauer, D. J. (2008). A note on the relation between factor analytic and item response theory models. *Structural Equation Modeling: A Multidisciplinary Journal*, 15, 136-153. doi: 10.1080/10705510701758406
- Reese, L. M., & Cotter, R. A. (1994). *A compendium of LSAT and LSAC-sponsored item types* (Tech. Rep. No. LSAC-RR-94-01). Newton, PA: Law School Admission Council.
- Rhemtulla, M., Brosseau-Liard, P. E., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods*, 17, 354-373. doi: 10.1037/a0029315

- Rizopoulos, D. (2006). ltm: An **R** package for latent variable modeling and item response theory analyses. *Journal of Statistical Software*, 17(5), 1-25. Retrieved from <http://www.jstatsoft.org/v17/i05/>.
- West, S. G., Finch, J. F., & Curran, P. J. (1995). Structural equation models with non-normal variables: Problems and remedies. In R. A. Hoyle (Ed.), *Structural equation modeling: Concepts, issues and applications* (pp. 56–75). Newbury Park, CA: Sage.
- Wirth, R. J., & Edwards, M. C. (2007). Item factor analysis: Current approaches and future directions. *Psychological Methods*, 12, 58-79. doi: 10.1037/1082-989x.12.1.58

7 | Models with Missing Data

Chapter Contents

7.1	Background	114
7.1.1	Types of Missing Data	114
7.2	Analyzing Data With Missing Values	117
7.2.1	Traditional Methods	118
7.2.2	Modern Methods	119
7.2.3	Auxiliary Variables	121
7.3	Example: Missing Data	121
7.4	Summary	128
7.5	Writing the Results	128
7.6	Exercises	128
7.7	References & Further Readings	130

7.1 Background

Missing data are almost inevitable in any research project, especially those involving humans. Depending on the pattern and amount of missing data, excluding these incomplete observations can seriously bias parameter estimates or diminish sample size. Consequently, missing data needs to be assessed and, if found, addressed in every study.

It will make the presentation of missing data types and methods for handling them easier if I first present a motivational example. In this example, I am interested in understanding the influence of an academic intervention on reading ability in elementary school students. I randomly assigned half of the students to receive the intervention and the other half to receive typical reading instruction. I conceptualized reading ability as a latent variable, and gathered data on three indicator variables: word reading, pseudoword reading, and reading comprehension. A path model of this example is shown in Figure 7.1.

7.1.1 Types of Missing Data

Little and Rubin (2002) posited three types of missing data: (a) missing completely at random (MCAR) (b) missing at random (MAR); and (c) not missing at random (NMAR). These three types of missing data are not mutually exclusive categories, though. In fact, Graham (2009) argued that pure versions of the typologies never really exist, so it is better to think of missing data as being on a continuum between them. Thus, the question is not what type of missing data is present in a dataset, but if the *pattern of missingness* precludes being able to use modern techniques to handle the missing data (see Section 7.2). A **missing data pattern** is the configuration of observed and missing values in a dataset.

7.1.1.1 Missing Completely at Random

Data are **missing completely at random** (MCAR) when the missing values on a given variable (i.e, its *missingness*) are unrelated to both that variable *and* any other variable in

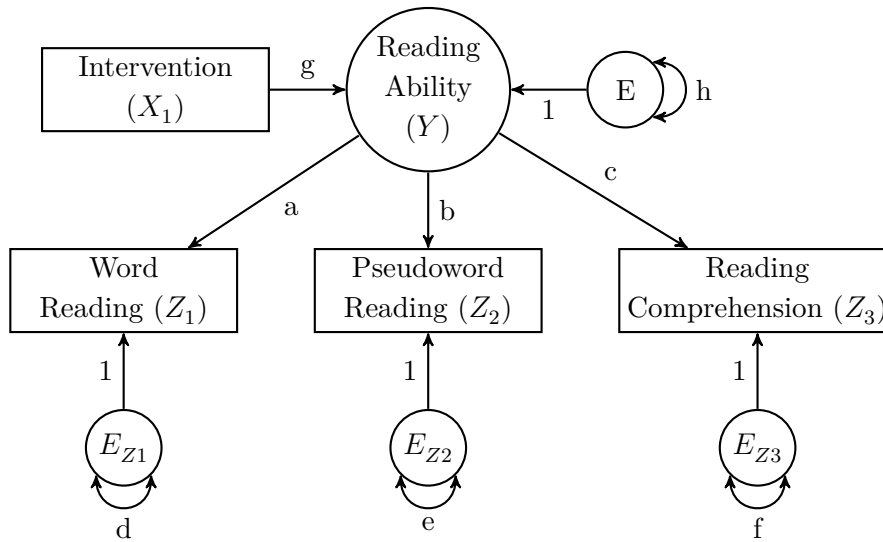


Figure 7.1 Path model of motivational example for missing data presentation.

the dataset.¹ The first of these conditions is sometimes called *missing at random* and the second is called *observed at random*. From a different perspective, data are MCAR if the observations with non-missing values represent a random subsample of the entire dataset.

*Missing
Completely at
Random (MCAR)*

Most (but not all) methods of handling data with values MCAR yield unbiased parameter estimates, i.e., estimates that are not systematically different from the population values. The main consequence of MCAR data is the loss of sample size, which can decrease statistical power and increase the standard error.

It does not violate the principle of MCAR if one variable's missingness is related to another variable's missingness. Such situations are not uncommon. For example, in longitudinal data collection designs with multiple cohorts, data might be collected by age of respondents (e.g., at ages 10-15 years) instead of every year the study has collected data. Another example is with psychological assessment and the number of variables to collect (usually multiple subtests from an instrument) is so time and financially costly that different cohorts within the sample are administered different parts of the test battery, with none of them taking the entire battery. This data missing solely because of a *planned missing design* is MCAR. I do not discuss designs with planned missing values, but give some references that discuss them more in Section 7.7.

In the motivational example, values on the reading comprehension test would be MCAR if they were unrelated to scores on the reading comprehension test itself (i.e., people with high levels of reading comprehension are just as likely to have missing values as people with low levels) *and* were not related to the pseudoword reading, word reading, or the intervention variables, either. Some reasons why data could be MCAR are, for example, the variable cod-

¹Technically, MCAR means that the missing values are unrelated to *any* variable, whether included in the dataset or not, but it is impossible to verify relationships empirically with variables not included in the dataset.

ing process had random input errors or a respondent got ill just before the test and did not complete the test later. The key is that there is no systematic reason why data are missing.

I can empirically evaluate if data are MCAR, and it is the only missing data mechanism that is testable. There are two approaches to doing this: univariate and multivariate. The univariate approach requires creating dummy codes for *each* variable that has missing values (e.g., 0 = missing, 1 = present), then doing one of two procedures. First, using each dummy-coded variable to represent group membership, calculate the effect size for the between-groups mean differences (e.g., the d or g effect sizes) for each dummy-coded variable, separately. If the effect size is small ($< 0.15 - 0.20$), this suggests that there is no systematic difference between responders and non-responders on other variables in the dataset, which would give some support for the data being MCAR. The second procedure involves doing a logistic regression, regressing each dummy-coded variable (separately) onto the other variables in the dataset. Having no large odds ratios (an effect size for logistic regression) indicates the data on the dummy-coded outcome are MCAR.

Section 7.7 has some references on interpreting odds ratios and using **R** for a logistic regression.

As an alternative to doing many univariate comparisons, Little (1988) proposed a multivariate test statistic for MCAR that is akin to simultaneously testing all the mean differences. Although it has not been tested extensively, what has been done has tended to support that the Type 1 error rate (i.e., saying the data is MCAR when it is not) is relatively stable across sample sizes, number of variables, and amount of missing data. The problem with both the univariate and multivariate methods, though, is that they only test for mean differences, not differences in (co)variances. Thus, they do not assess whether the relationships among the variables is the same for responders and non-responders. Fortunately, there has been some recent developments in this area, both theoretically and in **R**, which I discuss more in Section 7.3.

7.1.1.2 Missing At Random

Assuming values are MCAR is often unreasonable for many studies because missing values are usually related to other variables if enough pertinent information is collected (e.g., sex, education status, age). Thus, a second type of missing data is **missing at random** (MAR). Data are MAR if a given variable's missingness is unrelated to the variable itself, but is related to other variables in the dataset. Thus, MAR assumes that the variables that explain the missing data are included in the analysis either as predictors, outcomes, or auxiliary variables (more on auxiliary variables in Section 7.2.3). If handled appropriately, analysis of MAR data can yield unbiased parameter estimates.

In the motivational example, data on the reading comprehension variable would be MAR if students with high, average, or low reading comprehension scores were equally as likely to have missing responses, but students with low word reading scores had missing reading comprehension scores more often than students with average or high word reading scores. Within a word reading performance group, however, there is no relationship between missingness and reading comprehension scores. Thus, for analyses of MAR data to work properly, the variables related to the missingness need to be included in the analysis model.

Missing at Random (MAR)

I briefly mentioned MAR data in Section 7.1.1.1 as part of the MCAR definition. Thus, MCAR is really just a specific type of MAR.

Unlike MCAR, there is no test for values being MAR. Nonetheless, it can be useful to examine if any of the variables in the dataset are related to the missing responses. The logistic regression method discussed in Section 7.1.1.1 is a decent method for doing this.

7.1.1.3 Not Missing at Random

Data are **not missing at random** (NMAR) (sometimes called *non-ignorable* or *missing not at random*) if the missing values are not MCAR or MAR. More concretely, data are NMAR if a variable's missingness is related to that variable, even after including all possible covariates in the model. Values that are NMAR are a problem because they yield biased parameter estimates with traditional techniques and *could* yield biased results with modern techniques as well. Thus, the modern techniques I discuss in this chapter could work or it could require more complex techniques (e.g., instrumental variables, mixture modeling). The references in Section 7.7 describe some alternative techniques for data NMAR.

Not Missing At Random (NMAR)

A sensitivity analysis could indicate whether the modern methods are appropriate.

In the motivational example, if students with lower reading comprehension skills were more likely to have missing reading comprehension scores than students with higher skills, then the data on this variable would be NMAR. The problem, though, is that I do not know what the students' reading comprehension skills really are because I do not have their scores! Thus, NMAR is more of a conceptual consideration than an empirical one: Are students with low reading comprehension scores more likely to have missing data (e.g., absent when the test was administered, leave the study before taking this test)? This does not mean there are no data to examine, though. For example, I might observe that participants who have low scores on the reading portion of the state achievement test also have high rates of missing values on the reading comprehension test, which would provide some evidence that the data are likely NMAR.

7.2 Analyzing Data With Missing Values

If there are missing responses in a dataset, the best method to deal with them is a function of: (a) the type of missing data (see Section 7.1.1); (b) how much data are missing; and (c) the variables that have missing values.

For example, if the data are MCAR and only make up a small percentage (i.e., $< 3 - 5\%$) of the *entire* dataset, and the sample size is relatively large ($n > 200$), then listwise deletion (see Section 7.2.1) will likely not have a noticeable influence on the parameter estimates. As another example, say the variable of interest is the average value of a group of items on a questionnaire. The dataset contains all the item responses and a small percent of the item responses are missing. In such situations, it *may* be just as fine averaging the items that do have responses rather than using one of the modern methods listed below.

Another consideration is the roles the variables with missing values have in the model. Missing data on an endogenous variable pose different problems from missing data on an exogenous variable. When the data are MAR, then observations with missing values on the endogenous variable, but have values for all the exogenous variables, do not contribute any information to the outcome-predictor relationship. The information they provide is still useful, but they do not make a contribution to the path coefficient.

Consequently, there is not a one-size-fits-all method when dealing with missing values. Instead, it requires knowing how much data are missing on each variable, how the variables are to be used in the analysis, and a reasoned belief for the types of missing values.

7.2.1 Traditional Methods

Historically, missing data were addressed indirectly by one of two methods. First, observations with missing data were deleted either listwise or pairwise. With *listwise deletion* (sometimes called complete case analysis), observations (cases) with *any* missing values on variables used in a model are deleted before estimating any parameters. This is the default procedure for most statistical programs, including **R**. It provides unbiased parameter estimates only if data are MCAR. However, the price for removing entire observations is that the estimates' standard errors increase (i.e., are less precise) and statistical power decreases.

When initially examining a dataset, listwise deletion is fine to use. A more robust method typically should be used when conducting the final (reported) analysis.

With *pairwise deletion* (sometimes called available case analysis), the maximum amount of bivariate (pairwise) data available is retained for a single parameter estimate. This method is usually employed to estimate means and covariances, which are then used for other analyses (e.g., regression, latent variable model). In such instances, an observation (case) that has a missing value on one variable (A) would not be used to calculate the covariance between that variable and another variable (A and B), but would be included in all other covariance estimates (e.g., B and C).

There are some major problems with pairwise deletion. First, since (potentially) different observations are used to calculate each covariance, this can lead to a situation where two separate (non-overlapping) subsets of a dataset's observation are used for two covariance estimates. Moreover, the covariance matrices estimated using pairwise deletion are more likely to not return results or return impossible values (i.e., the matrix is singular/non-positive definite) than other techniques for handling missing data. Finally, while the parameter estimates from pairwise deletion may be unbiased if data are MCAR, it still leaves a big question unanswered: *What is the study's sample size?*

An American Psychological Association task force had this to say about listwise and pairwise methods for handling missing data:

Special issues arise in modeling when we have missing data. *The two popular methods for dealing with missing data that are found in basic statistics packages, listwise and pairwise deletion of missing values, are among the worst methods available for practical applications.* (Wilkinson & American Psychological Association Science Directorate Task Force on Statistical Inference, 1999, p. 598, emphasis added)

The second historical approach to handling missing data was *imputation*, which replaces a missing value with another plausible value, typically based on one of three strategies: (a) the mean/median of all present values of the variable; (b) regression-based predicted scores (using all the other variables in the dataset as predictors); or (c) use a pattern-matching technique to find another observation that has similar responses across all the other variables in

the dataset, and then replace the missing value with the matched observation's value (i.e., hot/cold-deck imputation). Mean imputation is never a good idea with any type of missing data. Regression and deck methods both sound reasonable, but because they only impute a single value for each missing response, they tend to produce underestimated variability and standard error estimates.

7.2.2 Modern Methods

There are two more modern alternatives to the more traditional techniques for handling missing values: full information maximum likelihood estimation and multiple imputation.

7.2.2.1 Full Information Maximum Likelihood

Full information maximum likelihood (FIML) is a type of maximum likelihood (ML) parameter estimation technique. ML estimation involves an iterative procedure to find parameter values that are the most likely for a model, given the variables' observed values. To use a ML estimator, there has to be an *a priori* assumption about the distribution of the variables, which is usually that they follow a multivariate normal distribution. I do not discuss the general steps involved in ML, as they are very well documented. Some of the sources in Section 7.7 discuss this in more depth.

Maximum likelihood estimation is the default for most LVM computer programs. This default ML estimator uses the sample's summary statistics (i.e., covariances and means) to find parameter estimates, irrespective of whether raw data or summary statistics were used as input. **Full information maximum likelihood**, however, finds the most likely parameter estimates for each observation, with the final parameter estimates being the ones that are the most likely across all observations. Conceptually, the difference between FIML and ML is the same as the difference between using full information and limited information estimation for categorical responses that I briefly discussed in Chapter 6. Thus, FIML can be used for item response theory parameter estimation, but cannot be used for typical item factor analysis models.²

*Full Information
Maximum
Likelihood
(FIML)*

Because FIML uses information from each observation, usually observations do not have to be removed if they are missing values. Moreover, unlike pairwise deletion, with FIML estimation observations with partial data can contribute to the estimation of *all* the parameters because FIML uses both observed data as well as the relationship among all the variables in the model.

7.2.2.2 Multiple Imputation

Unlike mean, regression, or deck imputation that impute a single value for a missing observation, **multiple imputation** (MI) creates multiple (m) datasets, each of which contain different plausible estimates of the missing values.

*Multiple
Imputation (MI)*

Multiple imputation involves a three-step process. The first step is to create the m datasets with imputed data. This is the most complex step, and differs by computer program as well

The three steps for MI are:

1. Imputation
2. Analysis
3. Pooling

²lavaan currently only allows listwise and pairwise deletion for categorical indicator variables with missing values.

as the type of data (e.g., categorical versus continuous, longitudinal versus cross-sectional). The gist of this step is that the non-missing data (Data_0) are used to estimate a mean vector ($\boldsymbol{\mu}_0$) and covariance matrix ($\boldsymbol{\Sigma}_0$). This initial mean vector and covariance matrix are then used to make *augmented* regression equations to predict the missing values. The augmentation in these regression equations is the addition of random error to the regression equations. The “new” dataset (Data_1) is used to make another mean vector and covariance matrix to which another random error term is added to each parameter estimate, which are then used to create a new mean vector ($\boldsymbol{\mu}_1$) and covariance matrix ($\boldsymbol{\Sigma}_1$). This procedure continues until the distribution of parameter estimates for the missing values has stabilized.³ The final result is m new datasets without any missing data.

If the missing values are from categorical variables, the above procedure do not work very well. One option in such cases is to use *chained equations* for the imputation step. The chained equations approach tailors the imputation model to the scale of the variable with missing values (e.g., a logistic regression model for a categorical variable, linear regression model for a continuous variable) so that a separate conditional model is specified for each variable with missing values. The specification of different models for *each* variable with missing values, as opposed to multivariate model for all variables simultaneously, is why this approach is so flexible in handling variables with different scale types. In **R**, the *mice* package (van Buuren & Groothuis-Oudshoorn, 2011), which stands for *multivariate imputation by chained equations*, does this type of imputation. I use the *mice* package for the example in Section 7.3.

The number of imputed datasets to create really depends on the amount of missing data, the type of variables that have missing values (e.g., categorical, continuous), and the patterns of the missing responses. As a rule of thumb, m should be at least 20, although 50 to 100 may be needed if the data has a lot of idiosyncrasies.

The second step in MI is the analysis of the different datasets, which involves estimating the model parameters in *all* of the m complete datasets, separately. The third step in MI is to pool the parameter estimates from the m datasets to calculate the final parameter estimates and their standard errors. The parameter estimates are simply the average of the m parameter estimates. The calculation of the standard errors is a little more complex, as each standard error is now comprised of two sources of variance: within dataset and between dataset. While most modern MI programs calculate the imputed standard error automatically, it is informative to understand the calculation.

Whenever combining variability measures, it is better to use variance than standard deviation. Taking the square root of the final value gives the standard deviation.

Within dataset variability (\bar{W}) is due to the regular sampling variability that is in all parameter estimates (i.e., typical standard error). For the MI datasets, \bar{W} is calculated by averaging the squared standard error across the m imputed datasets. Between dataset variability (B) is due to the missing data. It is the variability of the parameter estimates found among the m imputed datasets. The variability estimates are combined via Equation (7.1) to

³I have described a very simplified version of the *data augmentation* process for imputing data, but there are other procedures that take a somewhat different approach.

yield the total variability (T), the square root of which is the total standard error.

$$\sqrt{T} = \sqrt{\bar{W} + \left(1 + \frac{1}{m}\right) B} \quad (7.1)$$

7.2.3 Auxiliary Variables

An **auxiliary variable** (AV) is a variable that is not of interest in answering a research question, but is included in the model because it is either a potential cause or correlate of missingness in the variables of interest. Auxiliary variables can be used with both FIML and MI, but for them to work well they should be strongly correlated ($r \geq 0.50$) with the manifest variables of interest and with each other.

Auxiliary Variable

In the motivational example, say the average number of hours parents are at home a week is related to missing data on all three reading variables. Specifically, students who live with parents who are home more are less likely to have missing responses. The number of hours parents are home is not of interest to the study, but for the data to be MAR requires including this variable in the model.

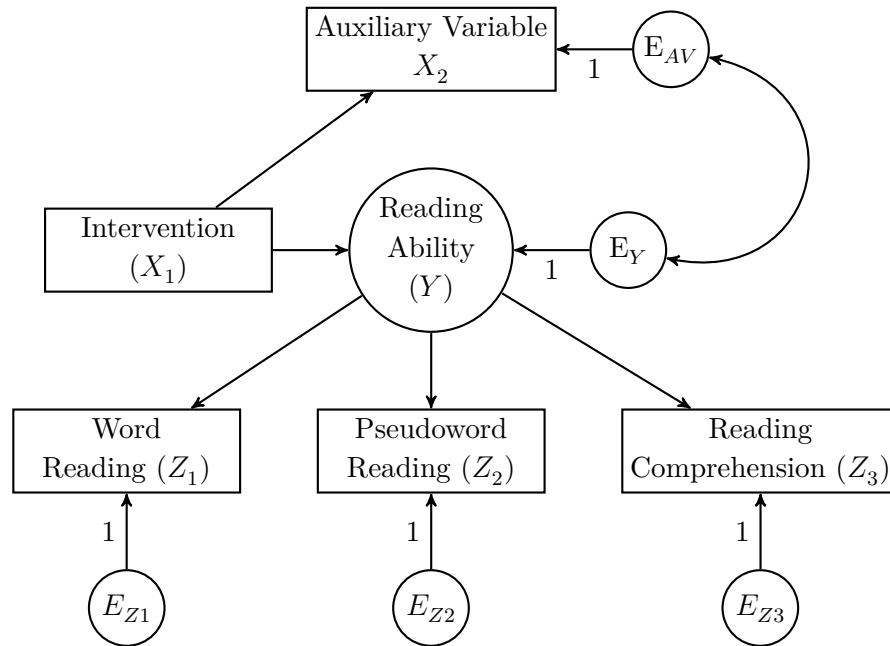
Some good candidates for AVs are the same variables used in the model, but measured at different waves of data collection. For example, if reading comprehension was measured at multiple time periods, but only the first and last waves were used in the analysis, then reading comprehension at, say, time periods 2, 3, 4, etc. would be good AV candidates. If many variables were collected, then another method to find AVs is to do an *exploratory* factor analysis (see Chapter 3) of all the study's variables and find the variables that have high loadings on the same LV that the variables of interest also have high loadings.

There are two ways to include an AV in a path model. When the model includes a regression term, then one option is to use the AV as an *extra endogenous variable* (see Figure 7.2a). For the extra endogenous variable model: (a) the AVs are predicted by all predictor variables (manifest or latent) in the regression models of interest; (b) the AVs' error variances covary with error variances from all regression models (for both manifest and latent variables); and (c) the AV error variances covary with each other. If the model does not include a regression term (or even if it does), then an alternative way to use an AV is as a *saturated correlate* (see Figure 7.2b). For the saturated correlates model: (a) the AVs covary with all exogenous manifest variables; (b) the AVs covary with all the endogenous manifest variables' error variances; and (c) the AVs covary with each other.

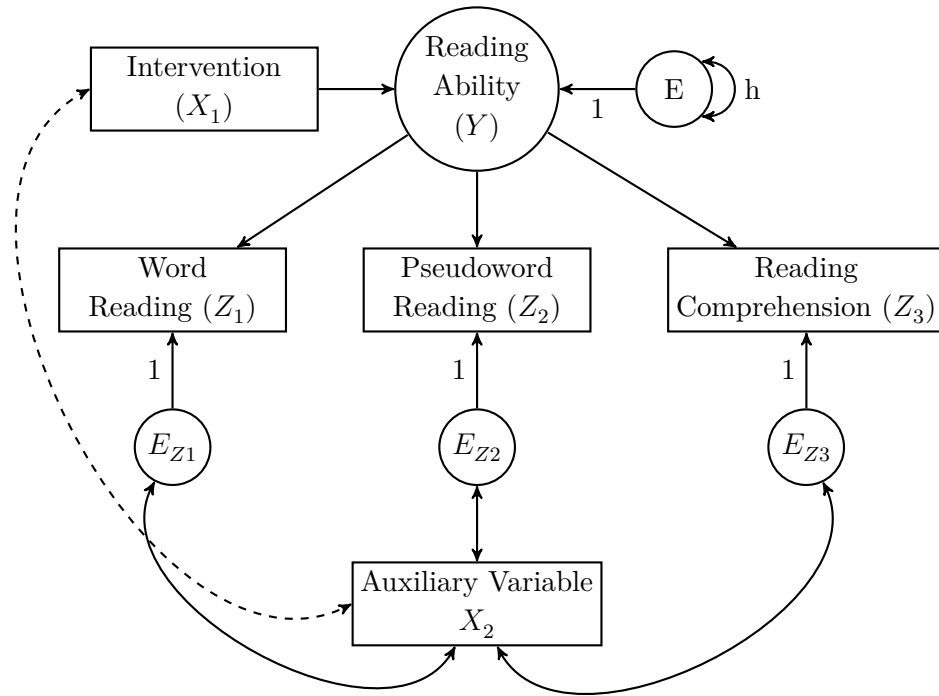
The *extra endogenous variable* method is sometimes called the *extra dependent variable* method.

7.3 Example: Missing Data

To demonstrate different ways of handling missing data and the difference in results, I did a very simple simulation study. Specifically, I simulated a single dataset from known population parameter values, then removed some data so that they be MCAR, MAR, or NMAR. In this section, I analyze the results using the different methods I discussed in this chapter and compare the results to the results from analyzing the complete data. Since the complete data results show what the parameter estimates should be, these comparisons give some indication of how well the various missing data methods work.



(a) Auxiliary variable in an *extra exogenous variable* model.



(b) Auxiliary variable in a *saturated correlates* model. Since there was random assignment to treatment groups, the dashed line should be 0.0.

Figure 7.2 Path models using an auxiliary variable in the motivational example (cf. Figure 7.1).

In the motivational example introduced in Section 7.1, the outcome is general reading ability, which is a latent variable with three indicator variables: (a) word reading (Z_1), (b) pseudoword reading (Z_2), and (c) reading comprehension (Z_3). The predictor variable (X_1) is a dichotomously-coded variable indicating whether the student received the treatment ($X_1 = 1$) or not ($X_1 = 0$), which was done by random assignment. In addition, I added an auxiliary variable to the dataset: average number of hours parents are home a week (X_2).

I simulated $n = 300$ observations to represent a random sample from the population. I then made three datasets with missing data have missing values only on Z_1 , Z_2 and Z_3 . In all datasets, approximately 30% of the values were missing on variables Z_1 , Z_2 , and Z_3 . No values were missing on X_1 or X_2 .

I explicitly show how to analyze the data with values MCAR data. For the MAR and NMAR datasets, I just show the results as the syntax is the same. In Table 7.1-Table 7.6, I compare the results from the analyses using traditional and modern missing data methods for the complete dataset as well as the datasets with values MCAR, MAR, and NMAR. In the tables, I present the estimates for the b , c , and g in Figure 7.1. I used the marker variable method to identify the LV, so a was constrained to 1.0 for all analyses.

The **R** syntax used for the traditional handling of the missing data is on the book's website.

When there are missing data, the first step is to determine the amount of missingness across the observations and the variables (i.e. patterns). The `mice` package contains the `md.pattern()` function that produces an observations-by-variable matrix of missing data patterns (1=observed, 0=missing). The matrix's rows represent observations and the columns are the variables. The rows and columns are sorted by increasing amounts of missing data. The last column and row in the matrix contain row and column counts, respectively.

`md.pattern()`

```
library(mice)
md.pattern(mcar.data)

##      x2 x1 z1 z2 z3
## 101  1  1  1  1  0
##  45  1  1  0  1  1
##  33  1  1  1  0  1
##  54  1  1  1  1  0
##  17  1  1  0  0  1
##  16  1  1  0  1  0
##  28  1  1  1  0  0
##   6  1  1  0  0  0
##    0  0 84 84 104 272
```

For the MCAR data, 101 respondents have no missing data, 45 are only missing values on Z_1 , etc. Likewise, there are no missing values for variables X_2 or X_1 , but Z_1 has 84 missing values, Z_2 has 84, and Z_3 has 104. There are 272 total values missing in the dataset. To translate the frequencies into percentage, just divide the results of `md.pattern()` by the total sample size. For example $\frac{101}{300} = 33.667\%$ of the sample has no missing values.

To test if the missing values are MCAR, I developed the `LittleMCAR()` function for the `BaylorEdPsych` package that calculates Little's (1988) MCAR test for up to 50 variables. The `MissMech` (Jamshidian, Jalal, & Jansen, 2014) package has the `TestMCARNormality()` function, which tests for MCAR values by examining equality of covariances.

`LittleMCAR()`
`TestMCARNormality()`


```
library(BaylorEdPsych)
mcar.little <- LittleMCAR(mcar.data)
mcar.little[c("chi.square", "df", "p.value")]
```

```
## $chi.square
## [1] 13.8
##
## $df
## [1] 23
##
## $p.value
## [1] 0.932
```

```
library(MissMech)
TestMCARNormality(mcar.data)
```

```
## Call:
```

```
..<Output Omitted>..
```

```
##
## Number of Patterns: 7
##
## Total number of cases used in the analysis: 294
##
## Pattern(s) used:
##      z1  z2  z3  x2  x1  Number of cases
## group.1  1  NA  NA   1   1           28
## group.2  1   1  NA   1   1           54
## group.3  1   1   1   1   1          101
## group.4  1  NA   1   1   1           33
## group.5  NA  NA   1   1   1           17
## group.6  NA   1   1   1   1           45
## group.7  NA   1  NA   1   1           16
##
##
##      Test of normality and Homoscedasticity:
##      -----
##
## Hawkins Test:
##
##      P-value for the Hawkins test of normality and homoscedasticity: 0.068
##
##      There is not sufficient evidence to reject normality
##      or MCAR at 0.05 significance level
```

Both tests indicate that the data appear to be MCAR.

Next, I analyze the data using the modern missing data methods I discussed in this chapter, starting with FIML estimation. *lavaan* has a native option to do FIML estimation using the `missing="fiml"` argument.

```
complete.model <- '
read =~ a*z1 + b*z2 + c*z3
read ~ g*x1
# label error variances
z1~~d*z1
```

```

z2~~e*z2
z3~~f*z3
read~~h*read
'
mcarFIML.fit <- sem(complete.model, data=mcar.data, missing="fiml")
summary(mcarFIML.fit, rsquare=TRUE, standardized=TRUE)

```

To add an AV to the model, there are two options. First, specify the model directly in `lavaan`. Doing so, though, requires explicitly modeling some exogenous variables' covariances, which overrides `lavaan`'s default method for estimating the values. I do this by using the `fixed.x=FALSE` argument.

```

# path model with auxiliary variable--saturated correlations
mcarFIMLAux.model<- '
read =~ a*z1 + b*z2 + c*z3
read ~ g*x1
# label error variances
z1~~d*z1
z2~~e*z2
z3~~f*z3
read~~h*read
# saturated correlations
x2~~ z1 + z2 + z3
'
mcarFIMLAux.fit <- sem(mcarFIMLAux.model, data=mcar.data, fixed.x=FALSE, missing="fiml")
summary(mcarFIMLAux.fit, standardized=TRUE)

```

The second option is to use the `auxiliary()` function in the `semTools` package. This function takes as its arguments: (a) a `lavaan` model; (b) the name of the AV(s); (c) the dataset's name; and (d) the function used in `lavaan` (e.g., `cfa()`, `sem()`, `growth()`). It uses FIML estimation and employs either the extra-exogenous-variable or saturated-correlates approach for the AV depending on what variables have missing values (e.g., covariates vs. indicator variables).

`auxiliary()`

For MI, there are two options. The first option is to impute the data, save the imputed datasets (step one), and then have the `semTools` package's `runMI()` function analyze the datasets and pool the results (steps two and three). This option allows *any* imputation program to create the imputed datasets, either inside or outside of **R**. The second option is to have the `runMI()` function complete all three steps. With this option, the two data imputation packages currently available are `Amelia` and `mice`. Below I use the `mice` package with $m=20$ imputed datasets.

`runMI()`

```

library(semTools)
mcarMI.fit <- runMI(complete.model, data = mcar.data, m = 20, miPackage = "mice", fun = "sem")
summary(mcarMI.fit)

```

Table 7.1 Path Coefficient Estimates for Data Missing Completely at Random (MCAR).

Dataset	b	Δb	c	Δc	g	Δg	$1 - R^2$	$\Delta 1 - R^2$	n
Complete	0.931	–	0.565	–	0.762	–	0.871	–	300
Listwise	0.791	0.14	0.475	0.09	0.682	0.08	0.654	0.217	101
Pairwise	0.93	0.00	0.49	0.075	0.718	0.044	0.723	0.148	134
Mean Imputation	0.923	0.008	0.445	0.12	0.52	0.243	0.375	0.50	300
FIML	0.825	0.106	0.444	0.121	0.76	0.00	0.99	-0.121	300
FIML/Aux	0.825	0.106	0.444	0.122	0.765	-0.002	0.995	-0.124	300
MI ($m = 20$)	0.884	0.047	0.443	0.122	0.739	0.024	0.90	-0.03	300

Δ : Difference between parameter estimates from dataset with missing values and complete dataset.
 $1 - R^2$ is the residual/error variance in Y .

Table 7.2 Standard Errors of Path Coefficient Estimates for Data Missing Completely at Random (MCAR).

Dataset	σ_b	$\Delta\sigma_b$	σ_c	$\Delta\sigma_c$	σ_g	$\Delta\sigma_g$	σ_{1-R^2}	$\Delta\sigma_{1-R^2}$	n
Complete	0.052	–	0.054	–	0.114	–	0.09	–	300
Listwise	0.114	-0.062	0.107	-0.054	0.176	-0.062	0.14	-0.049	101
Pairwise	0.117	-0.066	0.092	-0.039	0.167	-0.053	0.141	-0.051	134
Mean Imputation	0.138	-0.086	0.084	-0.031	0.094	0.019	0.074	0.016	300
FIML	0.074	-0.022	0.065	-0.012	0.128	-0.014	0.132	-0.041	300
FIML/Aux	0.073	-0.021	0.063	-0.009	0.127	-0.013	0.131	-0.041	300
MI ($m = 20$)	0.085	-0.033	0.066	-0.013	0.125	-0.011	0.116	-0.026	300

Δ : Difference between parameter estimates from dataset with missing values and complete dataset.
 $1 - R^2$ is the residual/error variance in Y .

Table 7.3 Path Coefficient Estimates for Data Missing at Random (MAR).

Dataset	b	Δb	c	Δc	g	Δg	$1 - R^2$	$\Delta 1 - R^2$	n
Complete	0.931	–	0.565	–	0.762	–	0.871	–	300
Listwise	0.954	-0.023	0.41	0.156	0.776	-0.013	0.691	0.18	145
Pairwise	1.058	-0.127	0.508	0.058	0.754	0.009	0.60	0.271	164
Mean Imputation	0.977	-0.046	0.50	0.066	0.54	0.222	0.384	0.487	300
FIML	0.933	-0.00	0.494	0.071	0.80	-0.039	0.851	0.019	300
FIML/Aux	0.931	0.00	0.543	0.022	0.823	-0.061	0.905	-0.035	300
MI ($m = 20$)	1.01	-0.079	0.569	-0.00	0.75	0.013	0.771	0.10	300

Δ : Difference between parameter estimates from dataset with missing values and complete dataset.
 $1 - R^2$ is the residual/error variance in Y .

Table 7.4 Standard Errors of Path Coefficient Estimates for Data Missing at Random (MAR).

Dataset	σ_b	$\Delta\sigma_b$	σ_c	$\Delta\sigma_c$	σ_g	$\Delta\sigma_g$	σ_{1-R^2}	$\Delta\sigma_{1-R^2}$	n
Complete	0.052	—	0.054	—	0.114	—	0.09	—	300
Listwise	0.086	-0.034	0.077	-0.023	0.147	-0.033	0.107	-0.017	145
Pairwise	0.116	-0.064	0.093	-0.04	0.141	-0.027	0.105	-0.015	164
Mean Imputation	0.116	-0.064	0.083	-0.03	0.089	0.025	0.063	0.027	300
FIML	0.076	-0.024	0.069	-0.016	0.126	-0.012	0.11	-0.019	300
FIML/Aux	0.072	-0.02	0.066	-0.012	0.127	-0.013	0.115	-0.025	300
MI ($m = 20$)	0.08	-0.028	0.081	-0.027	0.117	-0.00	0.094	-0.00	300

Δ : Difference between parameter estimates from dataset with missing values and complete dataset.
 $1 - R^2$ is the residual/error variance in Y .

Table 7.5 Path Coefficient Estimates for Data Not Missing at Random (NMAR).

Dataset	b	Δb	c	Δc	g	Δg	$1 - R^2$	$\Delta 1 - R^2$	n
Complete	0.931	—	0.565	—	0.762	—	0.871	—	300
Listwise	0.888	0.043	0.392	0.173	0.471	0.291	0.442	0.428	143
Pairwise	0.962	-0.031	0.503	0.062	0.462	0.30	0.40	0.472	157
Mean Imputation	1.00	-0.069	0.491	0.074	0.306	0.457	0.235	0.635	300
FIML	0.90	0.032	0.422	0.143	0.52	0.242	0.627	0.243	300
FIML/Aux	0.90	0.031	0.451	0.114	0.566	0.20	0.627	0.244	300
MI ($m = 20$)	0.822	0.109	0.473	0.092	0.60	0.166	0.651	0.22	300

Δ : Difference between parameter estimates from dataset with missing values and complete dataset.
 $1 - R^2$ is the residual/error variance in Y .

Table 7.6 Standard Errors of Path Coefficient Estimates for Data Not Missing at Random (NMAR).

Dataset	σ_b	$\Delta\sigma_b$	σ_c	$\Delta\sigma_c$	σ_g	$\Delta\sigma_g$	σ_{1-R^2}	$\Delta\sigma_{1-R^2}$	n
Complete	0.052	—	0.054	—	0.114	—	0.09	—	300
Listwise	0.139	-0.087	0.10	-0.045	0.126	-0.012	0.091	-0.00	143
Pairwise	0.169	-0.117	0.107	-0.053	0.125	-0.012	0.10	-0.006	157
Mean Imputation	0.165	-0.114	0.094	-0.041	0.073	0.041	0.05	0.041	300
FIML	0.10	-0.046	0.073	-0.019	0.116	-0.00	0.10	-0.005	300
FIML/Aux	0.093	-0.042	0.072	-0.018	0.114	0.00	0.093	-0.00	300
MI ($m = 20$)	0.111	-0.059	0.094	-0.04	0.124	-0.011	0.136	-0.046	300

Δ : Difference between parameter estimates from dataset with missing values and complete dataset.
 $1 - R^2$ is the residual/error variance in Y .

7.4 Summary

In this chapter I discussed three types of missing data: missing completely at random, missing at random, and not missing at random. In addition, I discussed why traditional methods for handling missing data might not work and showed how to implement two more robust methods: full-information maximum likelihood and multiple imputation. Moreover, I introduced auxiliary variables and showed how they can be added to a path model to improve parameter estimation accuracy when there are missing data. I finished the chapter by explicitly working through a example in **R** using a dataset with missing values.

7.5 Writing the Results

A major component of any data analysis should be to check to see if there are any missing data. This is relatively easy to do with the `mice` package's `md.pattern()` function as I demonstrated in Section 7.3. If there are missing values, Schlomer, Bauman, and Card (2010) suggest three imperatives when reporting the analysis results:

1. Report the amount of missing data.
2. Report the types of missing data (or at least the likely missing data types).
3. Report on how the most appropriate method for handling missing data was determined.

When reporting the amount of missing data, give the percentage of missing data, either in the text (e.g., “Missing data ranged from a low of 2% for Reading Comprehension to a high of 10% for Word Reading”) or in a table along with other descriptive statistics. If the study involves multiple time periods report the percentage of attrition and missing data at each time period as well as over the entire study.

To examine the type of missing data, use Little's (1988) test to assess means and the test developed by Jamshidian et al. (2014) to assess covariances. Although neither test has been studied in great depth, hopefully they converge in their results. If they do not converge, or if both tests indicate the missing values are not MCAR, use univariate methods to see if there are particular variables that strongly violate the MCAR assumptions. Be sure to include a conclusion about the types of missing data patterns in the Results section after reporting the amount of missing data.

When determining the most appropriate method to handle missing data, the default choice should be using FIML or MI (with auxiliary variables, if possible), unless you can make an argument that another method is better for a specific situation. If you chose another method, be explicit in your rationale for choosing it and state why it is better than either FIML or MI.

7.6 Exercises

- 7.1 The complete, MCAR, MAR, and NMAR data are on the book's website.
 - 7.1.a Import the three datasets with missing values into **R** and name them `mcar.data`, `mar.data`, and `nmr.data`, respectively.
 - 7.1.b Test to see if the missing data is MCAR.

- 7.1.c Fit the path model in Figure 7.1 to each dataset using listwise deletion.
- 7.1.d Fit the path model in Figure 7.1 to each dataset using FIML.
- 7.1.e Fit the path model in Figure 7.1 to each dataset using FIML and the auxiliary variable.
- 7.1.f Multiply impute the data ($m = 20$), then fit the model using the imputed data and pool the results.
- 7.2 Enders (2011) provides an example of a LVM with missing data. The data consists of 10 items (1, 2, 3, 10, 11, 12, 14, 18, 21, and 24) from the *Eating Attitudes Test* (Garner, Olmsted, Bohr, & Garfinkel, 1982), which is designed to measure two constructs: *Drive for Thinness* and *Food Preoccupation*. I have placed a copy of the dataset on this book's website and named it `eatingattitudes.dat`. In addition to an ID number, the dataset contains three additional variables: a score on an anxiety scale (`anx`), a score on a Western standards of beauty scale (`wsb`), and body mass index values (`bmi`). Although technically the items are categorical variables (responses range from 1-7), treat them as continuous variables for the LVM. The model Enders fit to the data is shown in Figure 7.3.
- 7.2.a Import the dataset into **R** and name it `eating.data`. The ID variable can be removed. The missing values are coded as `-99`. To change all the missing values codes to `NA`, use the `na.strings = "-99"` argument in the `read.table()` function.
- 7.2.b How many missing values are present and where are the data missing?
- 7.2.c Test to see if the missing values are MCAR.
- 7.2.d Fit the model to the data using listwise deletion. What is the sample size?
- 7.2.e Fit the model to the data using FIML.
- 7.2.f Fit the model using FIML and `bmi`, `wsb`, and `anx` as auxiliary variables.
- 7.2.g Multiply impute the data ($m = 20$), treating the eating variables as categorical in the imputation process so the item values are between 1 and 7. Then, fit the model using the imputed data and pool the results.

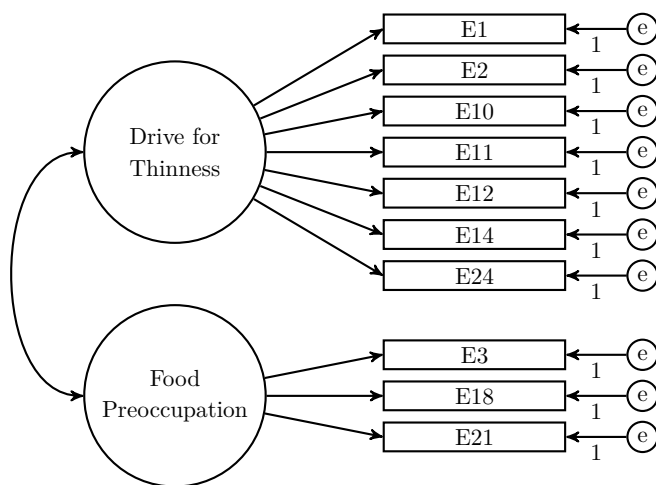


Figure 7.3 Path model of eating *Eating Attitudes Test* items for Exercise 7.2.

7.7 References & Further Readings

- Enders, C. K. (2005). Maximum likelihood estimation. In B. Everitt & D. C. Howell (Eds.), *Encyclopedia of behavioral statistics* (pp. 1164–1170). West Sussex, England: Wiley.
- Enders, C. K. (2011). *Applied missing data analysis*. New York, NY: Guilford.
- Garner, D. M., Olmsted, M. P., Bohr, Y., & Garfinkel, P. E. (1982). The Eating Attitudes Test: Psychometric features and clinical correlates. *Psychological Medicine*, *12*, 871-878.
- Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *10*, 80-100. doi: 10.1207/S15328007SEM1001_4
- Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual Review of Psychology*, *60*, 549-576. doi: 10.1146/annurev.psych.58.110405.085530
- Jamshidian, M., & Jalal, S. (2010). Tests of homoscedasticity, normality, and missing completely at random for incomplete multivariate data. *Psychometrika*, *75*. doi: 10.1007/s11336-010-9175-3
- Jamshidian, M., Jalal, S., & Jansen, C. (2014). MissMech: An **R** package for testing homoscedasticity, multivariate normality, and missing completely at random (MCAR). *Journal of Statistical Software*, *56*(6), 1-30. Retrieved from <http://www.jstatsoft.org/v56/i06>.
- Little, R. J. A. (1988). A test of missing completely at random for multivariate data with missing values. *Journal of the American Statistical Association*, *83*, 1198-1202.
- Little, R. J. A., & Rubin, D. B. (2002). *Statistical analysis with missing data* (2nd ed.). Hoboken, NJ: John Wiley.
- Lottes, I. L., DeMaris, A., & Adler, M. A. (1996). Using and interpreting logistic regression: A guide for teachers and students. *Teaching Sociology*, *24*, 284-298.
- McArdle, J. J. (1994). Structural factor analysis experiments with incomplete data. *Multivariate Behavioral Research*, *29*, 409-454. doi: 10.1207/s15327906mbr2904_5
- Muthén, B. O., Asparouhov, T., Hunter, A. M., & Leuchter, A. F. (2011). Growth modeling with nonignorable dropout: Alternative analyses of the STAR*D antidepressant trial. *Psychological Methods*, *16*, 17-33. doi: 10.1037/a0022634
- Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods*, *7*, 147-177. doi: 10.1037/1082-989X.7.2.147
- Schlomer, G. L., Bauman, S., & Card, N. A. (2010). Best practices for missing data management in counseling psychology. *Journal of Counseling Psychology*, *57*, 1-10. doi: 10.1037/a0018082
- van Buuren, S., & Groothuis-Oudshoorn, K. (2011). Multivariate imputation by chained equations in **R**. *Journal of Statistical Software*, *45*(3). Retrieved from <http://www.jstatsoft.org/v45/i03/>.
- Wilkinson, L., & American Psychological Association Science Directorate Task Force on Statistical Inference. (1999). Statistical methods in psychology journals: Guidelines and explanations. *American Psychologist*, *54*, 594-604. doi: 10.1037/0003-066X.54.8.594

8 | Sample Size Planning

Chapter Contents

8.1	Background	131
8.1.1	Traditional Sample Size Determination	131
8.1.2	Determining Sample Size Through a Monte Carlo Study	133
8.1.3	Example: Sample Size for a Simple Regression Model	134
8.1.4	Example: Sample Size for a Latent Variable Model	137
8.2	Summary	142
8.3	Writing the Results	142
8.4	Exercises	143
8.5	References & Further Readings	144

8.1 Background

When designing a study, whether using a LVM or not, it is important to consider the sample size needed for the analysis. *Required for what?*, you might ask. Usually it is the sample size required to reject a false *null hypothesis* (H_0) for a given effect size.¹ This idea stems from null hypothesis significance testing (NHST). NHST, at least how it is currently implemented, pits two competing hypotheses against each other. H_0 is usually set up to be the antithesis of the hypothesis of interest. Conversely, the *alternative hypothesis* (H_a or H_1) is every hypothesis that is not H_0 . Often, H_0 is specified such that a parameter is equal to a value (e.g., $H_0 : \delta = 0$), which makes H_a be that the parameter is not equal to the value (e.g., $H_a : \delta \neq 0$).

There are four interrelated concepts when determining the needed sample size: (a) type 1 error; (b) type 2 error; (c) sample size; and (d) effect size. Type 1 error (α) is the frequency of rejecting a true H_0 , based on multiple data samples from the same population. Thus, when α is large, I will frequently say my data indicate there is an effect, when one is not present in the population (i.e., large *false +* rate). Conversely, type 2 error (β) is the frequency of accepting a false H_0 , based on multiple data samples from the same population. When β is large, I will often conclude that my data indicate there is *not* an effect when one is present in the population (i.e., large *false -* rate). Statistical **power** is defined in terms of β : it is $1 - \beta$. That is, across the multiple samples, power is the frequency I will say the results indicate there is an effect, when there is one in the population.

β is used to indicate type 2 error and, sometimes, the population version of a path coefficient (see Section 2.1.5). The meanings are independent and should *not* be used interchangeably.

Power

8.1.1 Traditional Sample Size Determination

As an example, say I want to determine if the entrance exam scores for the 2012-2013 entering class at a university are different than the scores from the 2011-2012 entering class. The null and alternative hypotheses are:

¹Kelley and Maxwell (2012) provide an alternative reason for determining sample size: precision of parameter estimates. The methods presented in this chapter can be used for purposes. Since a focus on power is more common when determining sample size, that is the focus I use throughout this chapter.

H_0 : 2012-2013 class' average score = 2011-2012 class' average score

H_a : 2012-2013 class' average score \neq 2011-2012 class' average score

To set α , I have to decide on how many times I want to reject a true H_0 . In this example, that is equivalent to concluding the mean entrance exam score is different between the two groups *when it is the same*. If I am satisfied with that happening 10% of the time; then $\alpha = 0.10$. Likewise, to set β , I have to decide on how many times I want to accept a false H_0 . In this example, that is equivalent to concluding the average entrance exam score in the two groups is the same *when it is different*. Often, this type of error has less cost, so it is set higher than α . If I am satisfied with making a type 2 error 20% of the time; then $\beta = 0.20$ and the power for the test of a between-group mean difference is $1 - \beta = 0.80$.

There are a variety of effect size (ES) measures I could use, but if I assume that the entrance exam scores follow a normal distribution, then a typical ES to use when comparing means is the standardized mean difference, d , defined as

$$d = \frac{\mu_a - \mu_b}{\sigma} \quad (8.1)$$

where μ_a and μ_b are the means for groups A and B , respectively, and σ is the standard deviation (SD) from one of the groups or a pooled SD estimate from both groups. Figuring out the value of the ES is the hardest part in determining sample sizes, as it requires a bit of sooth-saying. For this example, pretend I found that the average entrance exam score SD for the university in the previous twenty classes has been 4.10. Moreover, I found out that at other peer universities, the average entrance exam scores for the 2011-2012 and 2012-2013 classes were 24.00 and 26.00, respectively. My ES estimate is now:

$$d = \frac{26 - 24}{4.1} = 0.488$$

Since I set α , β , and have an ES estimate, the sample size is already determined. To estimate it, though, it is a little cumbersome. Fortunately, the `pwr` package is designed for such situations. When comparing two groups' means, I would typically use a t test for statistical inference (see Section 1.1.12.4). Thus, I use the `pwr` package's `pwr.t.test()` function. The `pwr.t.test()` function has four main arguments, of which I only need to specify three: (a) α (`sig.level`); (b) $1 - \beta$ (`power`); (c) effect size (`d`); and (d) sample size (`n`). Supplying values for any three arguments is sufficient to return the fourth.

`pwr.t.test()`

```
library(pwr)
pwr.t.test(d = 0.49, sig.level = 0.1, power = 0.8)

##
##      Two-sample t test power calculation
##
##              n = 52.2
##              d = 0.49
##      sig.level = 0.1
##      power = 0.8
##      alternative = two.sided
```

```
##
## NOTE: n is number in *each* group
```

The results indicate the sample size needs to be at least 52 in each group, for a total sample size of 104.

The steps I just demonstrated is typical method for determining sample size. They are unrealistic, though, for all but the simplest research projects. For example, seldom does the collected data exactly meet all the assumptions for an inferential statistic, there are no missing data, or the variables are measured without error. To determine sample sizes for more realistic situations, I can use a Monte Carlo study.

8.1.2 Determining Sample Size Through a Monte Carlo Study

An alternative to traditional power analysis is a **Monte Carlo** (MC) study. Muthén and Muthén (2002) were the first to demonstrate how to do this in a LVM context using their *MPlus* program. Barrett (2007) had the following to say about their procedure:

The procedure outlined by Muthén and Muthén (2002) is simply beyond the computational stamina of many SEM investigators to implement as a routine method; but at the moment, as a coherent method for power analysis in SEM, it is all there is available to an investigator. It may well be the price that has to be paid for using a modelling [sic] procedure and subsequent goodness-of-fit NHST which is so complex that no routine way of assessing its statistical power exists. (p. 821)

Monte Carlo Study

Because MC studies rely on random data generation, your results and the results I present in this chapter will not be the exact same. They should be fairly close, though.

In MC studies, multiple (p) samples (i.e., datasets) are simulated from a population with known parameter values, variable distributions, and variable relationships. Then, for each of the samples, parameters from the model of interest are estimated along with their standard errors. Last, the parameter estimates and standard errors are averaged over all these samples. When inspecting the combined results, the following statistics should be examined: (a) parameter estimate bias, (b) standard error bias, (c) coverage, and (d) power Muthén and Muthén (2002) suggested criteria for each of these statistics, which are given in Table 8.1.² The chosen sample size is the smallest value that that does not cause any violations of the criteria in Table 8.1.

Parameter estimate bias is the difference between the true (i.e., population value) of the parameter (i.e., the known value) and the average estimated value across the p samples. Because the parameter estimates' scales are dependent on the variables' scales, bias estimates are not comparable. Divide the bias estimate by the known population value, however, remedies the situation and produces **relative bias**, which is a proportion measure. The *relative standard error bias* is the difference between the SD of the p samples' parameter estimates and the average of the parameter's p standard error estimates, divided by the parameter estimates' SD. **Coverage** is the proportion of the p samples that the $(1-\alpha)\%$ confidence interval

Bias

Relative Bias

Coverage

² Kelley and Maxwell (2012) discuss one other criterion: making the confidence interval for the parameter of interest be *sufficiently narrow*. I do not discuss this aspect of sample size determination, but it can be easily incorporated into Monte Carlo simulations.

Table 8.1 Monte Carlo Simulation Criteria.

Measure	Suggested Criteria
Parameter estimate bias (relative)	$\not\geq .10$ for <i>all</i> parameters
Standard error bias (relative)	$\not\geq .10$ for <i>all</i> parameters
	$\not\geq .05$ for parameters on which power is being assessed
Coverage	Between 0.91 and 0.98
Power	Close to 0.80 without going below

Criteria developed by Muthén and Muthén (2002).

contains the true parameter value. In a MC study, *power* is operationalized to be the proportion of the p samples for which the null hypothesis is rejected for a given parameter at the specified α level (i.e., how often p -value $< \alpha$).

8.1.3 Example: Sample Size for a Simple Regression Model

To demonstrate MC methods for determining sample size, I re-visit the entrance exam scores example. To determine the required sample size, I use the `sim()` function in the `simsem` (Pornprasertmanit, Miller, & Schoemann, 2012) package. Before I can simulate the samples, I must specify the entrance exam research question as a structural equation model (SEM), being explicit about all the estimated parameter values. In Figure 3.1, I stated that regression is a type of SEM, and ANOVA is a type of regression. Further, the two-sample t test is a type of ANOVA. Thus, I can specify the entrance exam example as a SEM.

The `simsem` package’s homepage provides many additional examples: <http://simsem.org>

`des()`

A correlation is the same as a standardized regression coefficient when there is one predictor and all the variables are Z scores.

The easiest way to translate the entrance exam example to an SEM is to translate the d ES value into a correlation coefficient, r . The `des()` function in the `compute.es` package converts the d ES to the r ES. The only arguments for the `des()` function are the d ES value and the sample sizes for the two groups. The samples sizes are only used to estimate the ES’ standard error, so any value works for the current purposes as long as it is the same for both groups.

```
library(compute.es)
des(d = 0.49, n.1 = 100, n.2 = 100)

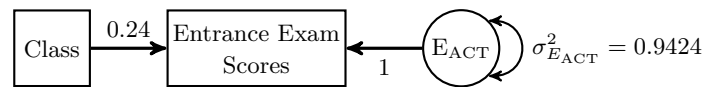
## $MeanDifference
##      d var.d      g var.g
## 0.4900 0.0206 0.4881 0.0204
##
## $Correlation
##      r var.r
## 0.23796 0.00432

..<Output Omitted>..
```

The results indicate that the d ES of 0.49 is equivalent to the r ES of 0.238.

Next, I specify this SEM model using `lavaan` syntax, fixing the values for all free parameters. I do this by specifying the values as labels (see Section 4.5). I already know the regression coefficient value, and, since it is standardized, I do not have to include the intercept in

Figure 8.1 Path model of entrance exam example with fixed parameter values. All variables are standardized.



the model. The only other parameter I need to specify is the error variance. For a simple regression with standardized variables, the error variance is $1 - r^2$, which I obtained through the tracing rules (see Section 2.1.3). Figure 8.1 shows a path model of the entrance exam example with all the specified parameter values.

```
# data generation model
act.model <- '
# regression
act ~ 0.24*class
# error variance
act ~~ 0.9424*act
'
```

I did not provide the variance for the *class* variable since it was not a free parameter.

To make sure my values were specified correctly, I fit the model in `lavaan` using the `do.fit=FALSE` argument, which makes the parameter estimates either the fixed values or the default starting values. Moreover, using the `fitted()` function returns the resulting means and covariances from the model using the fixed or default starting values for parameter estimates.

`fitted()`

```
# show the model's fixed and default values
act.fit <- sem(act.model, do.fit = FALSE)
summary(act.fit, rsquare = TRUE)
fitted(act.fit)
```

The `sim()` function requires two models. The first, which I already specified, generates the data. The second model is the one the `sim()` function uses to estimate the parameters from the simulated samples. Typically, the two models will be the same (except one has explicit parameter values), but this is not required (i.e., a misspecified model).

```
# analysis model
act2.model <- '
act ~ class
'
```

`sim()`

The `sim()` function's main arguments are: (a) the number of samples (i.e., simulation repetitions; `nRep`); (b) the model to generate the data (`generate`); (c) the model to analyze the data (`model`); (d) the sample size (`n`); and (e) the `lavaan` function to use for the analysis (`lavaanfun`). The `multicore` argument is optional, but setting it to `TRUE` tells **R** to use multiple processors within the computer for the data simulation. If a computer has multiple cores, this option can *considerably* lessen the time required to create the data. Another optional argument is `seed`, which determines the starting point for the random sample draws. Setting this value returns the same results for each run of the `sim()` function.

If your computer does not have multiple cores, **R** should ignore the `multicore=TRUE` argument.

Muthén and Muthén (2002) recommend using $p = 10,000$ samples for a sample size study. They also recommend repeating the study with at least two different seed values. In my syntax, I use a seed of 0 for one study, and a seed of 1978 for the second. These values have no real meaning, and any other integer can be selected.

Start with a smaller number of replications (e.g., 20, 500) to make sure the syntax is specified correctly. Then move to 10,000.

summaryParam()

Once all the simulations are complete, the `summaryParam()` function returns the results of interest. If the `detail=TRUE` argument is specified, then it returns all the bias and coverage information.

```
library(simsem)
act.power <- sim(nRep=1000, generate=act.model, model=act2.model, n =104,
lavaanfun = "sem", multicore=TRUE, seed=0)
summaryParam(act.power,alpha = 0.10,detail=TRUE)
```

##	Estimate.Average	Estimate.SD	Average.SE	Power..Not.equal.0.	Std.Est	Std.Est.SD
## act~class	0.241	0.097	0.095	0.806	0.240	0.094
## act~~act	0.925	0.131	0.128	1.000	0.934	0.045
##	Average.Param	Average.Bias	Coverage	Rel.Bias	Std.Bias	Rel.SE.Bias
## act~class	0.240	0.001	0.946	0.004	0.009	-0.018
## act~~act	0.942	-0.017	0.923	-0.018	-0.131	-0.018
##	Average.CI.Width	SD.CI.Width				
## act~class	0.374	0.037				
## act~~act	0.503	0.071				

The *Rel.Bias* column gives the relative bias for the parameter, the *Rel.SE.Bias* column gives the relative bias for the standard error, the *Coverage* column gives the coverage, and the *Power..Not.equal.0* column gives the power. Because I specified the `alpha = 0.10` argument, all the results are based on $\alpha = 0.10$. The parameter of interest is the regression coefficient, `act class`. All the bias and coverage values are within the desired bounds, and power, for $n = 104$, is at the desired level of 0.80. The time it took my computer to complete one run of 10,000 samples was 194 seconds.³

Use the `summaryTime()` function to calculate the completion time.

seq()

If I did not know the sample size to use, then instead of giving a single value to the `n` argument, I can give a range of values using the sequence function, `seq()`. For example, to examine power for values from $n = 50$ to $n = 120$, increasing by increments of 10, I use `seq(50,120,10)` for the `n` argument. This produces one simulation with $n = 50$, one with $n = 60$, etc. To increase the number of simulations at each sample size, wrap the `seq()` function inside the replicate function, `rep()`. For example, to repeat the 50-120 sequence 500 times, I use: `rep(seq(50,120,10), 500)`. I did not use 10,000 repetitions here because, since I simulated data for 8 different sample sizes, that would make $8 \times 10,000 = 80,000$ simulations! Once I find an approximate sample size, I can then do the study with a single sample size and many more simulations.

rep()

```
act.n <- sim(model=act2.model, n =rep(seq(50,120,10), 500), generate=act.model,
lavaanfun = "sem", multicore=TRUE)
```

plotPower()

abline()

After completing the repetitions for the multiple sample sizes, I have the option of plotting a power curve (i.e., a graph of the power estimate as a function of sample size), using the `plotPower()` function, with the parameter of interest as the value for the `powerParam` argument. The curve for the entrance exam example, using sample sizes spanning 50-120, is shown in Figure 8.2. To make reading the graph easier, I added a horizontal line at 0.80 using the `abline()` function, which adds lines to existing **R** plots.

³The computer is a 2010 iMac (OS X 10.8.3) with a 2.93 Ghz i7 processor and 8 GB 1333 MHz DDR3 memory.

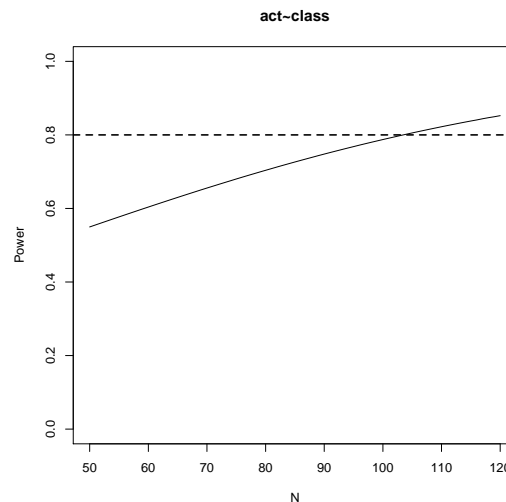


Figure 8.2 Power curve for entrance exam example.

```
# power curve
plotPower(act.n, alpha = 0.1, powerParam = "act-class")
# add line at y= 0.80
abline(h = 0.8, lwd = 2, lty = 2)
```

An alternative to graphically displaying the results is to use the `getPower()` function, which returns the power for each parameter at all specified sample sizes. To find the minimum sample size needed for a specific power level, use the `findPower()` function. It uses an object created using the `getPower()` function and, including the `iv="N"` argument, returns the smallest sample size needed for the specified power level for each parameter. If the `findPower()` function returns `NA`, it means *no* specified sample size produced power at, or above, the desired level. If it returns `Inf`, then *all* specified sample sizes produced power at, or above, the desired level.

`getPower()`

`findPower()`

```
act.pwr.n <- getPower(act.n, alpha = 0.10)
# Find the sample size needed for power = .0.8
findPower(act.pwr.n, iv="N", power=0.8)
## act~class act~~act
##      104      Inf
```

The results from the ACT study indicate that a sample size of 104 might be sufficient for the parameter of interest to have enough power, which I already verified earlier.

8.1.4 Example: Sample Size for a Latent Variable Model

The sample procedures used for the simple two-group study can be extended to more complex designs and analyses. One example comes from Muthén and Muthén (2002).⁴ In this example, they were interested in the sample size needed to find group differences in growth rate. This can be parameterized as a latent curve model (LCM; see Chapter 5), with the major focus being on the regression coefficient from the group variable to the latent slope.

The specifications of the LCM are as follows (Figure 8.3 shows a path model with all pertinent values):

⁴My analysis model differs slightly from that given by Muthén and Muthén (2002). I do not constrain the latent intercept to be 0.0 and I allow the latent slope and latent intercept to covary. Thus, my model has 7 *df* while their model has 9 *df*.

- Four equidistant time periods measuring the same continuous variable ($Y_1 - Y_4$) that is normally distributed.
- There is a covariate representing group membership (X). Group membership is dummy coded (0=no, 1=yes), and there are equal numbers of both groups. Thus, the mean and variance of X are 0.50 and 0.25, respectively.
- The mean of the latent intercept and latent slope are 0.00 and 0.20, respectively.
- The latent mean and latent slope covariance is 0.00.
- The residual variances for the indicator variables are all 0.50.
- The regression coefficient for the latent intercept is 0.50, with a residual variance of 0.25.
- The residual variance for the latent slope is 0.09.
- $\alpha = 0.05$ and the desired power is 0.80
- The regression coefficient for the latent slope is 0.20, which is the same as having a d ES of 0.63.

The ES calculation is a straightforward application of the tracing rules. The between-group difference in latent slopes is 0.20. The latent slope variance is $0.20 \times 0.25 \times 0.20 + 1 \times 0.09 \times 1 = 0.10$. Thus, using Equation (8.1), I get:

$$d = \frac{0.20}{\sqrt{0.10}} = 0.63$$

The procedure to find a sample size for the LCM follows that from the entrance exam example. I have to specify a data generation model and a data analysis model. Since there are many more parameters in this mode, I labeled the parameter of interest as a . For this labeling to work, I have to do it in *both* the generation and analysis models. To check the parameter values, I use the `do.fit=FALSE` argument in `lavaan` with the data generation model. Because I explicitly constrained the means and variances of an exogenous variable (X), I need to include the `fixed.x=FALSE` argument as well.

```
# Data generation model
lcm.pop.model <- '
# latent variable model
i =~ 1*y1 + 1*y2 + 1*y3 + 1*y4
s =~ 0*y1 + 1*y2 + 2*y3 + 3*y4
# latent variable means
i ~ 0.00*1
s ~ 0.20*1
# regressions, with parameter of interest labeled
i ~ 0.50*x
s ~ a*x + 0.20*x
# mean and variance of x
x ~ 0.50*1
x ~~ 0.25*x
# manifest (residual) variances
y1 ~~ 0.5*y1
y2 ~~ 0.5*y2
y3 ~~ 0.5*y3
y4 ~~ 0.5*y4
# latent variable residual variances/covariances
```

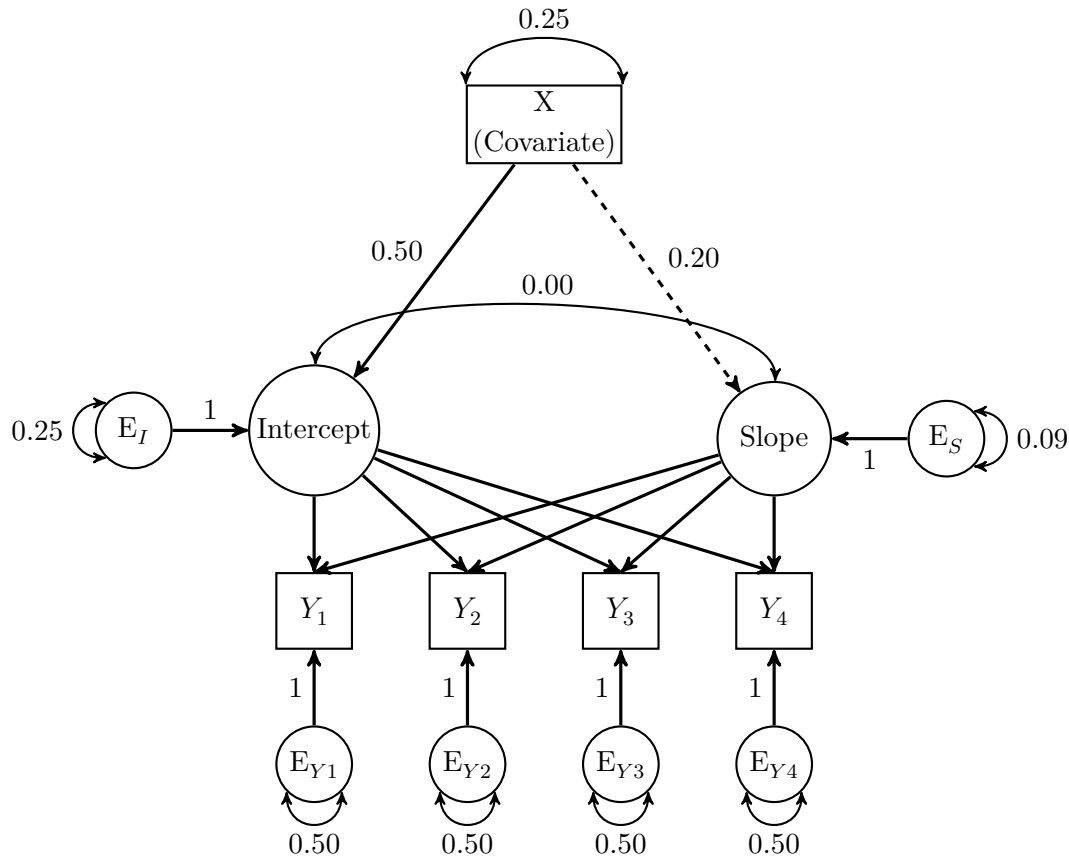


Figure 8.3 Path model for latent curve model sample size example. Values for the latent slope loadings are 0.0-4.0 for $Y_1 - Y_4$, respectively. All latent intercept loadings are 1.0. Dashed path represents the parameter of interest.

```
i~~0.25*i
s~~0.09*s
i~~0.0*s
'

# analysis model
lcm.model <- '
# latent variable model
i =~ 1*y1 + 1*y2 + 1*y3 + 1*y4
s =~ 0*y1 + 1*y2 + 2*y3 + 3*y4
# regressions
i ~ x
s ~ a*x
'
```

Initially, I did not know a specific sample size to examine, so I simulated sample sizes from 100 to 200, by 10. For each sample size, I simulated 500 datasets.

```
# run simulations
lcm.n <- sim(nRep=NULL, model=lcm.model, n=rep(seq(100,200,10), 500), generate=lcm.pop.model,
lavaanfun = "growth", multicore=TRUE)

# obtain power estimates for each sample size
lcm.pwr.n <- getPower(lcm.n)
```



```
# find the sample size needed for power = .0.8
findPower(lcm.pwr.n, "N", 0.8)
##   i~x      a y1~~y1 y2~~y2 y3~~y3 y4~~y4   i~~i   s~~s   i~~s   i~1   s~1
##   100    149    100    Inf    Inf    100    135    100    NA    NA    100
```

It appears that a sample size of 149 should be adequate for this model.

I checked this sample size of 149 by redoing the study using 10,000 repetitions. I did this twice, each with a different seed value. The bias and coverage values were all within the specified limits, and power was 0.80. For space considerations, I only show results from one of the studies.

```
lcm1.power <- sim(nRep=10000, model=lcm.model, n =149, generate=lcm.pop.model,
lavaanfun = "growth", multicore=TRUE, seed=998877)
lcm2.power <- sim(nRep=10000, model=lcm.model, n =149, generate=lcm.pop1.model,
lavaanfun = "growth", multicore=TRUE, seed=99)
summaryParam(lcm1.power,alpha = 0.05,detail=TRUE)
summaryParam(lcm2.power,alpha = 0.05,detail=TRUE)
##   Estimate.Average Estimate.SD Average.SE Power..Not.equal.0. Std.Est Std.Est.SD
## a           0.200         0.072      0.071           0.800      0.323      0.117
##   Average.Param Average.Bias Coverage Rel.Bias Std.Bias Rel.SE.Bias Average.CI.Width
## a           0.200         0.000      0.949     -0.001    -0.004      -0.007      0.279
##   SD.CI.Width
## a           0.023
```

With research designs that have multiple time points, attrition is often an issue. So, an analysis of sample size without accounting for missing data is unrealistic. Mimicking the missing data amounts in Muthén and Muthén's (2002) study, I added missing data to the sample size study to reflect attrition by having an increasing amount of missing data on the outcome variable. Moreover, for the second through the fourth time-periods, the probability of missing data is influenced by the covariate, whereas the first time-point has data missing completely at random.

For a value of zero on the covariate, the outcome variable has 12% of the data missing at the first time period, 18% missing at the second, 27% missing at the third, and the fourth has 50% missing. For a value of one on the covariate, 12% is missing at the first time period, 38% missing at the second, 50% missing at the third, and 73% missing at the fourth.

The `simsem` package has a variety of ways to include missing data in the simulations, most of which use the `miss()` function. I use the logit method, as it has the ability to graph the missing data amounts. The logit method requires a `lavaan`-like script that specifies how much data should be missing for a given variable. Each line of the script begins with a manifest variable, then the regression symbol (`~`), and then values for the amount of missing data. The values after the `~` are input for the inverse logit function, which is defined in Equation (8.2).

`miss()`

$$\text{inverse logit} = \frac{1}{1 + \exp[-1(a + bX)]} \quad (8.2)$$

where a and b are intercept and slope values, respectively, and X is a variable in the model.

For example, if $a = 1.5$ and $b = 0$, then the inverse logit is 0.1824, which would specify that 18.243% of the values should be missing. Likewise, if $a = 1.5$ and $b = 1$, then when

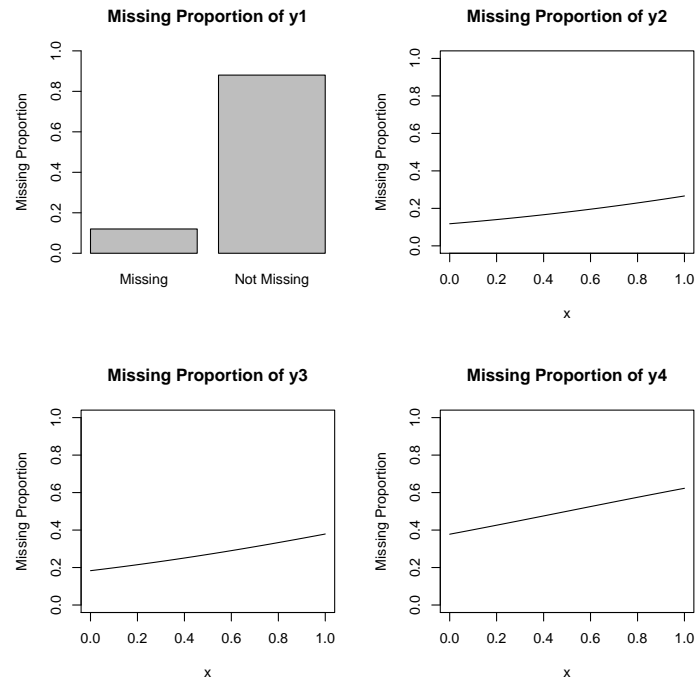


Figure 8.4 Plot of missing data to include in the simulated datasets for each of the four outcome variables, $Y_1 - Y_4$.

$X = 1$ 37.754% of the values should be missing. As an alternative, I can specify the average amount missing via the `p()` specification, placing the proportion missing inside the parentheses. Thus, `y2 ~ p(0.18)` and `y2 ~ -1.5` both indicate that 18.243% of the data should be missing from variable `y2`. The following syntax produces the amount of missing data for the outcome variable in the LCM at the levels I already specified.

```
lcm.pcnt.missing <- '
y1 ~ p(0.12)
y2 ~ p(0.18) + 1*x
y3 ~ p(0.27) + 1*x
y4 ~ p(0.50) + 1*x
'
missing.model <- miss(logit=lcm.pcnt.missing)
```

To plot the amount of missing data that the logit equations specified, use the `plotLogitMiss()` function.

`plotLogitMiss()`

```
plotLogitMiss(lcm.pcnt.missing, xlim = c(0, 1))
```

The resulting plot is shown in Figure 8.4.

Because I am generating data with missing values, I also need to indicate how to estimate the parameters with missing values in the dataset (see Chapter 7). By default, `simsem` uses full information maximum likelihood (FIML) when there are missing values. As the current dataset just has missing values on the outcome variables, not as much information is gained by using FIML as using multiple imputation (MI). To use MI, I have to add argument values indicating the number of imputations for each dataset (`m`) and the **R** package to do the imputation (`package`). I use the `mice` package, in the syntax below, but the *Amelia* package can be used as well.

```
missing.model <- miss(logit = lcm.pcmt.missing, m = 10, package = "mice")
```

I start the sample size study by specifying sample sizes from $n = 200$ to $n = 300$, increasing by 10. As I have to simulate multiple datasets for each sample size *and* I have to impute the data for *each* simulated dataset, I set the number of imputed datasets to $m = 10$ and only did $p = 100$ simulations per sample size. This still took a little over 30 minutes to complete.

```
lcm.missing.n <- sim(nRep=NULL, model=lcm.model, n=rep(seq(200,300,10), 100),
generate=lcm.pop.model, lavaanfun = "growth", multicore=TRUE, miss=missing.model)
```

```
# get power estimates for each sample size
lcm.missing.pwr.n <- getPower(lcm.missing.n)
```

```
# Find the sample size needed for power = .0.8
findPower(lcm.missing.pwr.n, "N", 0.8)
```

```
##   i~x      a y1~~y1 y2~~y2 y3~~y3 y4~~y4 i~~i   s~~s   i~~s   i~1   s~1
##   200     273   Inf    Inf    Inf    200   226    233    NA    NA    200
```

The results indicate that approximately 273 respondents provide sufficient power for a regression coefficient of 0.20. I then did another MC study with $p = 10,000$ repetitions for $n = 273$. I did the study two times, using two different seeds. Each study took over 3.50 hours. I only show the results from one study to save space, but the power, bias, and coverage from both studies were all at acceptable values.

```
lcm1.missing <- sim(nRep=10000, model=lcm.model, n =273, generate=lcm.pop.model,
lavaanfun = "growth", multicore=TRUE, miss=missing.model, seed=4335)
```

```
lcm2.missing <- sim(nRep=1000, model=lcm.model, n =273, generate=lcm.pop.model,
lavaanfun = "growth", multicore=TRUE, miss=missing.model, seed=5443)
```

```
summaryParam(lcm1.missing,alpha = 0.05,detail=TRUE)
```

```
summaryParam(lcm2.missing,alpha = 0.05,detail=TRUE)
```

```
##   Estimate.Average Estimate.SD Average.SE Power..Not.equal.0. Std.Est Std.Est.SD
## a           0.193         0.069         0.068           0.805         0.32         0.121
##   Average.Param Average.Bias Coverage Rel.Bias Std.Bias Rel.SE.Bias Average.CI.Width
## a           0.2         -0.007         0.939        -0.034        -0.098         -0.019         0.265
##   SD.CI.Width Average.FMI1 SD.FMI1 Average.FMI2 SD.FMI2
## a           0.032         0.381         0.119           0.4         0.126
```

8.2 Summary

In this chapter, I discussed sample size planning using a Monte Carlo (MC) simulation study. The MC method is very general and will work for any study that can be specified as a structural equation model. I demonstrated how the sample size estimates from a MC study equal the results from more traditional sample size methods when the model is very simple (e.g., mean differences between two groups). I then showed how MC methods can encompass a wider range of models than traditional methods, as well as incorporate missing data. I concluded the chapter by explicitly working through an example same size study for a longitudinal design that used a latent curve model.

8.3 Writing the Results

Typically, sample size planning is just one component of a larger project, done before any data is collected. Consequently, more is written about them in technical reports and grant

proposals than research articles. Nonetheless, no matter what the document, report the type 1 error (α) and type 2 error (β) levels chosen, the parameters that were of major interest for the sample size study, as well as the effect sizes used. For the effect sizes, be sure to indicate why you used a particular value (e.g., pilot studies, previous publications). Most traditional measures of effect size can be translated to structural equation model parameters (and vice versa), using the appropriate formulae (e.g., Borenstein, 2009) and the tracing rules (see Section 2.1.3). While not required, it is very helpful to create a path model with values for *all* the estimated parameters before starting the MC simulations. Such an exercise not only helps to specify the syntax correctly, but also helps show any parameter for which a value has not been assigned.

8.4 Exercises

- 8.1 Figure 8.5 shows the path model from the motivational example in Chapter 7, with plausible values for the parameter estimates. Assume random assignment to treatment, the mean of the latent variable is 0.0, and the intercepts for all the reading variables are all 0.0.
- 8.1.a Set $g = .40$. Translate this to a d ES.
- 8.1.b Draw a power curve for $n = 100$ to $n = 300$ (increasing by 25) and $\alpha = 0.05$. Generate 500 simulation for each sample size.
- 8.1.c Based on the power curve from Exercise 8.1.b, what sample size is needed for power of 0.80?
- 8.1.d Simulate 10,000 repetitions using two different seeds for the sample size chosen in Exercise 8.1.c. Are the bias and coverage estimates within the specified ranges? If so, how much power is there for the test that the intervention is causing a change in reading ability?
- 8.1.e Repeat Exercise 8.1.d, but have missing data on the reading scores in the following amounts: word reading: 5%; reading comprehension: 15%. Make the missing values for the pseudoword reading variable be a function of the word reading variable. With very low word reading scores (i.e., -3), 25% of the values are missing; with very high word reading scores (i.e., +3), 5% of the values are missing. Use FIML for the parameter estimates. What is the power for the parameter of interest, g , using the sample size selected in Exercise 8.1.c?
- 8.1.f Repeat Exercise 8.1.e, but use MI ($m = 10$) to account for the missing data. What is the power for the parameter of interest?

Warning!

Depending on your computer's processing speed, it could take > 1 hour to complete the 10,000 repetitions in Exercise 8.1.d - Exercise 8.1.f.

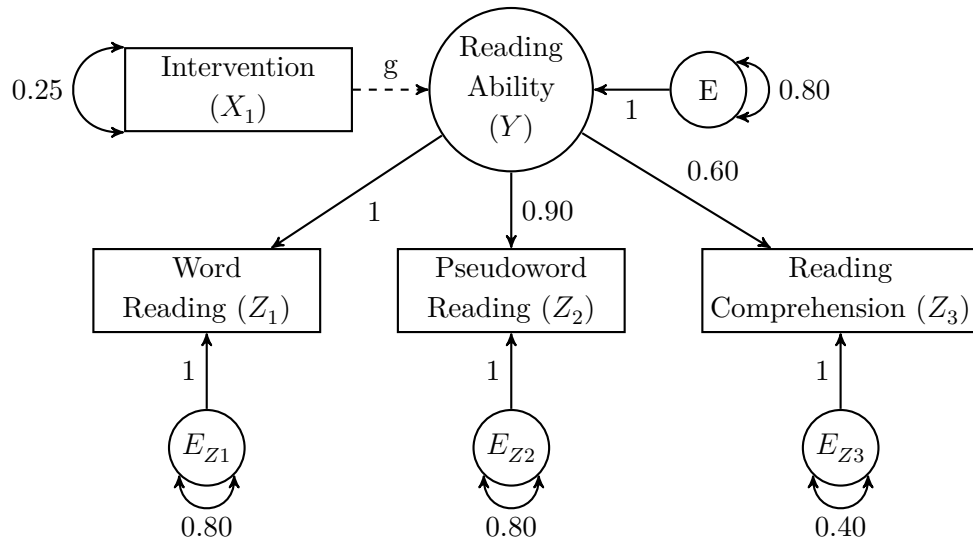


Figure 8.5 Path model of motivational example from Chapter 7. Parameter values are unstandardized. The mean of X_1 is 0.50, and intercepts for all the reading variables are 0.0. The dashed path represents the parameter of interest.

8.5 References & Further Readings

- Barrett, P. (2007). Structural equation modelling: Adjudging model fit. *Personality and Individual Differences*, 42, 815-824. doi: 10.1016/j.paid.2006.09.018
- Borenstein, M. (2009). Effect sizes for continuous data. In H. Cooper, L. V. Hedges, & J. C. Valentine (Eds.), *The handbook of research synthesis and meta-analysis* (2nd ed., pp. 221–235). New York, NY: Russell Sage Foundation.
- Kelley, K., & Maxwell, S. E. (2012). Sample size planning. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.), *APA handbook of research methods in psychology, Vol 1: Foundations, planning, measures, and psychometrics* (pp. 181–202). Washington, DC: American Psychological Association.
- Muthén, L. K., & Muthén, B. O. (2002). How to use a Monte Carlo study to decide on sample size and determine power. *Structural Equation Modeling: A Multidisciplinary Journal*, 9, 599-620. doi: 10.1207/S15328007SEM0904_8
- Pornprasertmanit, S., Miller, P., & Schoemann, A. (2012). *simsem: SIMulated Structural Equation Modeling* [Computer software]. Retrieved from <http://CRAN.R-project.org/package=simsem>.

9 | Hierarchical Latent Variable Models

Chapter Contents

9.1	Background	145
9.1.1	Example: Hierarchical Latent Variable Models	148
9.2	Summary	151
9.3	Writing the Results	151
9.4	Exercises	151
9.5	References & Further Readings	152

9.1 Background

This chapter covers LVMs with hierarchical structures. The terminology used in the literature on hierarchical LVMs can be confusing, not because the concepts are especially difficult, but because of the inconsistent use of terms. I have chosen to use the terms *higher-order* and *bi-factor* purposefully, as there is more consistency with their use than other terms. I used the *hierarchical* term to name this chapter because it has been used to describe both higher-order and bi-factor models.

All the LVMs I have considered in this book have only included LVs that directly influence the indicator variables. An extension of this model is a **higher-order** LVM. A higher-order model specifies that there are LVs that directly influence the indicator variables (called **first-order** LVs [FOLV]) as well as LVs that directly influence the first-order LVs (i.e., **second-order** LVs [SOLV]).

Higher-order models are usually applicable when (a) there are multiple (≥ 3) FOLVs that substantially covary with each other, and (b) there is a SOLV that is hypothesized to account for the relationship among the FOLVs. A typical higher-order model for cognitive abilities data is one that posits that the SOLV *general intelligence* (*g*) is the sole reason why various measures of cognitive ability, each of which are FOLVs, are related to each other (see Figure 9.1a).

With higher-order models, the covariance of the FOLVs is accounted for by a SOLV that represents a higher-order construct (i.e., what is in common among all the FOLV). Consequently, higher-order models decompose a FOLV into two separate parts: the part that is explained by the SOLV and the part that is independent of the SOLV. This second part is sometimes called a **specific factor**, but it is really just error, i.e., the portion of a FOLV's variance that is unexplained by the SOLV. The FOLVs' specific factors are typically uncorrelated with the SOLV and among themselves.

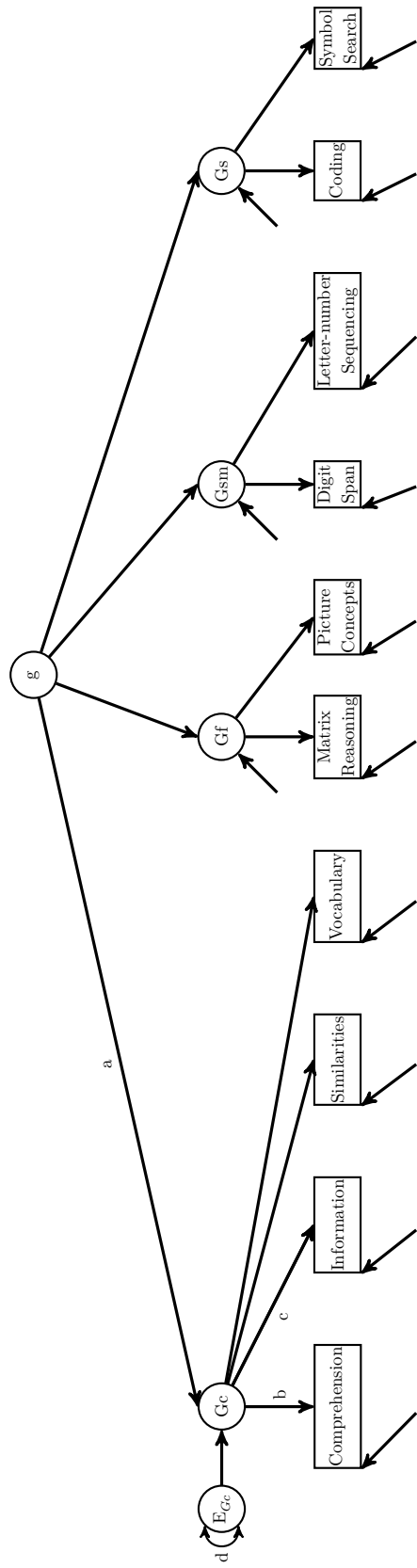
*Higher-Order
Latent Variable
Model*

*First-Order
Latent Variable
(FOLV)*

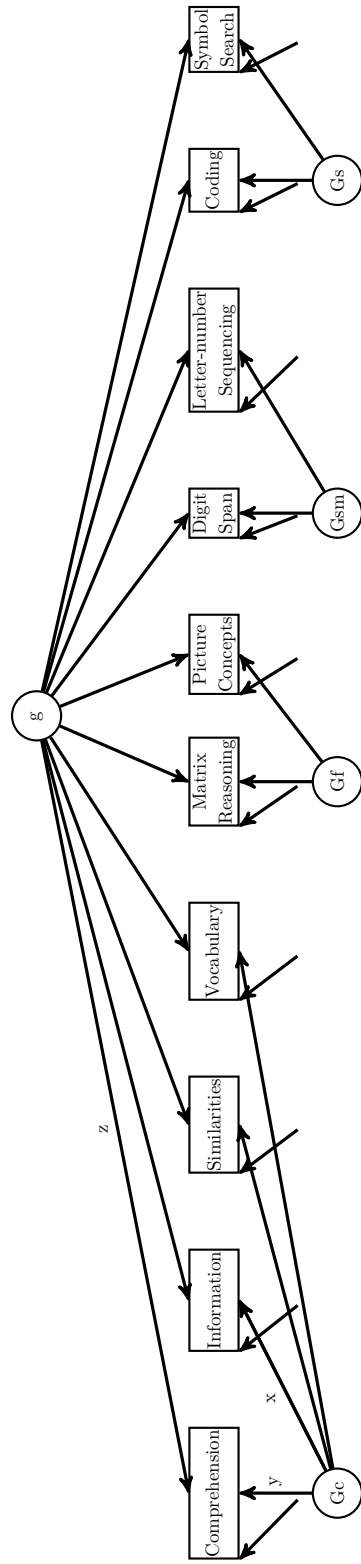
*Second-Order
Latent Variable
(SOLV)*

There can be multiple SOLVs.

Specific Factor



(a) Higher-order model.



(b) Bi-factor model.

Figure 9.1 Hierarchical models of select subtests from the Wechsler Intelligence Scale for Children-Fourth Edition. Gc: Comprehension Knowledge, Gf: Fluid Reasoning, Gsm: Short-term Memory, Gs: Processing Speed. All error variances, except for d , shown with diagonal arrow with no attached node.

With higher-order models, the influence of the SOLV on the manifest variables (MVs) is mediated by the FOLVs. That is, the SOLVs have an *indirect* influence on the MVs (see Section 2.4). The value of a SOLV's indirect influence on a MV can be estimated using the tracing rules (see Section 2.1.3). Specifically, the indirect influence is calculated by multiplying the loading on the FOLV by the FOLV's loading on the SOLV's (e.g., $b \times a$ in Figure 9.1a). Similarly, influence of a specific factor on a MV can be computed by multiplying the loading on the FOLV by the standard deviation of the FOLV's specific factor (e.g., $b \times \sqrt{d}$ in Figure 9.1a). The ratio of the SOLV's indirect influence to the influence of the specific factor is the exact same for all the indicator variables for a given FOLV. For example, in Figure 9.1a:

$$\frac{b \times a}{b \times \sqrt{d}} = \frac{c \times a}{c \times \sqrt{d}}$$

Identification of a SOLV is similar to that with FOLVs (see Section 3.2.1). The difference is that the unique information is not coming from the manifest variables, but the FOLVs. Thus, three FOLVs produce $3 \times 4/2 = 6$ non-redundant pieces of information to estimate three specific factors (error variances) and three loadings (or two loadings and one latent variance). Although the entire LVM may be overidentified, the SOLV part of the model is just-identified with only three FOLVs. With four FOLVs, there are $4 \times 5/2 = 10$ non-redundant pieces of information to estimate four specific factors and four loadings, making the SOLV part of the model overidentified.

An alternative (actually, a generalization) to the higher-order model is a **bi-factor** model. An example is shown in Figure 9.1b. A bi-factor model specifies that *all* the LVs are FOLVs, but some LVs are general (i.e., influence all the manifest variables) and some are **domain specific** (i.e., influence a portion of the manifest variables). Moreover, all the FOLVs are specified to be unrelated with each other, indicating that not only are the domain-specific factors independent of each other, but they are independent of the general LV as well.

Bi-Factor Model

*Domain-Specific
Latent Variable*

Bi-factor models are usually applicable when: (a) there is a general factor that is hypothesized to account for the covariance among the MVs; (b) there are multiple domain-specific LVs, each of which is thought to have an unique influence beyond a general LV; and (c) interest is in the domain-specific LVs as well as the general LV.

There are some parallels between the higher-order and bi-factor models. First, the interpretation of the bi-factor model's general LV and the higher-order models' SOLV are similar. Second, the specific factors in the higher-order model are similar to the domain-specific LVs in the bi-factor model. Third, in both the bi-factor and higher-order models the FOLVs are all uncorrelated with each other.

There are some major distinctions between the two models, as well. The distinctions mainly stem from the fact that in bi-factor models, the domain-specific LVs are totally independent of the general LV, whereas with the higher-order model the SOLV has a direct influence on the FOLVs. This independence yields some results for the bi-factor model that are impossible with the higher-order model. First, bi-factor models allow for a direct examination

Table 9.1 Wechsler Intelligence Scale for Children-Fourth Edition Subtest Covariances ($n = 550$).

		1	2	3	4	5	6	7	8	9	10
Comprehension	1	8.29	5.37	2.83	2.83	5.50	6.18	3.52	3.79	2.30	3.06
Information	2	5.37	9.06	4.44	3.32	6.66	6.73	3.77	4.50	2.67	4.04
Matrix Reasoning	3	2.83	4.44	8.35	3.36	4.20	4.01	3.19	3.72	2.40	3.70
Picture Concepts	4	2.83	3.32	3.36	8.88	3.43	3.33	2.75	3.39	2.38	2.79
Similarities	5	5.50	6.66	4.20	3.43	9.18	6.77	3.88	4.53	2.06	3.59
Vocabulary	6	6.18	6.73	4.01	3.33	6.77	9.12	4.05	4.70	2.59	3.67
Digit Span	7	3.52	3.77	3.19	2.75	3.88	4.05	8.88	4.54	2.65	3.44
Letter Number	8	3.79	4.50	3.72	3.39	4.53	4.70	4.54	8.94	2.83	4.20
Coding	9	2.30	2.67	2.40	2.38	2.06	2.59	2.65	2.83	8.76	4.53
Symbol Search	10	3.06	4.04	3.70	2.79	3.59	3.67	3.44	4.20	4.53	9.73

Data taken from Parkin and Beaujean (2012, pp. 117-118).

of the relationship between the domain-specific LVs and the MVs independent of the general LV. Such relationships cannot be examined directly in higher-order models because the FOLVs are not independent of the SOLV. Second, bi-factor models can examine if a domain-specific LV is directly related to an external variable, over and above the general LV. Again, such relationships cannot be examined directly in higher-order models because the FOLVs are not independent of SOLV. Third, the bi-factor model allows for a determination of the need for the domain-specific LVs without influencing the general factor. For example, in Figure 9.1, the bi-factor model would allow for an examination of whether the Fluid Reasoning (Gf) LV is needed by fitting a model with it and a model without it. With both models, the general LV, g , is unaffected. Such a comparison would be impossible with the higher-order model as removing Gf would change the definition of g .

9.1.1 Example: Hierarchical Latent Variable Models

For this example, I use the Wechsler Intelligence Scale for Children-Fourth Edition (WISC-IV; Wechsler, 2003) data from Chapter 3, but add some additional variables. The revised covariance matrix is shown in Table 9.1.

Before specifying a higher-order model, I fit a model with four FOLVs and examine their correlations to determine if a SOLV is appropriate. This is an important first step because if the FOLVs are not related to each other, then there is no sense in fitting a higher-order model.

```
# model with four FOLVs
wisc4.fourFactor.model <- '
gc =~ Comprehension + Information + Similarities + Vocabulary
gf =~ Matrix.Reasoning + Picture.Concepts
gsm =~ Digit.Span + Letter.Number
gs =~ Coding + Symbol.Search
'

wisc4.fourFactor.fit<-cfa(model=wisc4.fourFactor.model, sample.cov=wisc4.cov,
```

```
sample.nobs=550)
summary(wisc4.fourFactor.fit, fit.measure=TRUE, standardized=TRUE)
```

The correlations among the FOLVs range from 0.56 to 0.824, which indicate a SOLV might be appropriate.

The specification of the higher-order model is similar to that from the four-LV model, only now I specify a SOLV whose indicator variables are the four FOLVs.

```
1 # higher-order model
2 wisc4.higherOrder.model<-'
3 gc =~ Comprehension + Information + Similarities + Vocabulary
4 gf =~ Matrix.Reasoning + Picture.Concepts
5 gsm =~ Digit.Span + Letter.Number
6 gs =~ Coding + Symbol.Search
7
8 g =~ NA*gf + gc + gsm + gs
9 g =~ 1*g
10 '
```

Notice the the `NA*` in front of the `gf` term in line 8. This tells `lavaan` to estimate this loading instead of constraining it to be 1.0 (the default). The trade-off for doing this is that g 's variance has to be constrained to 1.0, which I specify on line 9. I do this because I do not want to constrain all the latent variables' variances to be 1.0 with the `std.lv=TRUE` argument, as I do not want the specific factors' variances to be constrained to be 1.0. I do want to set the SOLV's variance, however, to estimate all the SOLV's loadings.

There are other ways to specify a higher-order model, but I find this way tends to produce fewer problems with parameter estimation.

Next, I estimate the parameters and obtain the results.

```
wisc4.higherOrder.fit <- cfa(model=wisc4.higherOrder.model, sample.cov=wisc4.cov,
sample.nobs=550)
summary(wisc4.higherOrder.fit, standardized=TRUE)
```

```
## lavaan (0.5-16) converged normally after 55 iterations
```

```
..<Output Omitted>..
```

```
##
##              Estimate Std.err Z-value P(>|z|) Std.lv Std.all
## Latent variables:
##   gc =~
##     Comprehension    1.000
##     Information      1.160    0.056   20.813    0.000    2.194    0.762
##     Similarities     1.165    0.056   20.753    0.000    2.545    0.846
##     Vocabulary       1.217    0.056   21.857    0.000    2.555    0.844
##   gf =~
##     Matrix.Resnng     1.000
##     Pictur.Cncpts     0.847    0.079   10.754    0.000    2.670    0.885
##   gsm =~
##     Digit.Span        1.000
##     Letter.Number     1.172    0.087   13.505    0.000    1.990    0.689
##   gs =~
##     Coding            1.000
##     Symbol.Search     1.441    0.142   10.141    0.000    1.685    0.566
##   g =~
##     gf                1.851    0.122   15.173    0.000    1.967    0.661
##     gc                1.794    0.112   16.023    0.000    2.306    0.772
```

##	gsm	1.822	0.130	14.070	0.000	0.927	0.927
##	gs	1.293	0.133	9.709	0.000	0.730	0.730
##							
##	Variances:						
##	g	1.000				1.000	1.000
##	Comprehension	3.467	0.240			3.467	0.419
##	Information	2.567	0.203			2.567	0.284
##	Similarities	2.634	0.208			2.634	0.287
##	Vocabulary	1.976	0.181			1.976	0.217
##	Matrix.Resnng	4.377	0.424			4.377	0.525
##	Pictur.Cncpts	6.026	0.435			6.026	0.680
##	Digit.Span	4.996	0.378			4.996	0.564
##	Letter.Number	3.606	0.382			3.606	0.404
##	Coding	5.610	0.430			5.610	0.641
##	Symbol.Search	3.210	0.584			3.210	0.330
##	gc	1.596	0.226			0.332	0.332
##	gf	0.533	0.340			0.135	0.135
##	gsm	0.547	0.241			0.141	0.141
##	gs	1.464	0.263			0.467	0.467

Using the tracing rules, I can estimate the indirect influence of the second-order and first-order LVs on the MVs. For example, the standardized loading of the Comprehension subtest score on g is computed as $0.762 \times 0.818 = 0.623$. Further, to calculate the influence of the Gc specific factor on the Comprehension subtest, I multiply the standardized loading of the Comprehension subtest score on Gc (0.762) by the square root of Gc 's standardized variance ($\sqrt{0.332} = 0.576$), yielding: $0.762 \times 0.576 = 0.439$

The specification of a bi-factor model in `lavaan` is:

```
1 # bi-factor model
2 wisc4.bifactor.model<-'
3 gc =~ Comprehension + Information + Similarities + Vocabulary
4 gf =~ a*Matrix.Reasoning + a*Picture.Concepts
5 gsm =~ b*Digit.Span + b*Letter.Number
6 gs =~ c*Coding + c*Symbol.Search
7 g =~ Information + Comprehension + Matrix.Reasoning + Picture.Concepts + Similarities +
8 Vocabulary + Digit.Span + Letter.Number + Coding + Symbol.Search
9 '
```

The syntax is very similar to that of the four-factor model I fit earlier, but there is one difference. In the four-factor model, I did not have to have any constraints on the parameter estimates, but I need constraints for the bi-factor model (lines 4-6). Because the LVs are uncorrelated with each other, the Gf , Gsm , and Gs LVs are empirically underidentified if I do not constrain the loadings to be the same for these three LVs (see Section 3.2.1.4).

Orthogonal is synonymous with uncorrelated.

To estimate the bi-factor model's parameters, I use the `orthogonal=TRUE` argument. This makes all the exogenous LVs in the model uncorrelated. I standardized all the LVs (i.e., `std.lv=TRUE`) because I had to constrain the loadings for the domain-specific LVs with only two indicator variables. If I do not use this argument, then all the constrained factor loadings are set to 1.0.

```
wisc4.bifactor.fit<-cfa(model=wisc4.bifactor.model, sample.cov=wisc4.cov, sample.nobs=550,
std.lv=TRUE, orthogonal=TRUE)

summary(wisc4.bifactor.fit, fit.measure=TRUE, standardized=TRUE)
```

Table 9.2 Covariances for the Wechsler Preschool and Primary Scale of Intelligence-Fourth Edition (2:6-3:11 Year Old Sample; $n = 600$).

	IN	RV	PN	BD	OA	PM	ZL
Information	9.61	5.48	6.23	3.91	3.75	4.76	2.79
Receptive Vocabulary	5.48	9.61	5.30	3.91	3.75	4.36	2.88
Picture Naming	6.23	5.30	9.00	3.78	3.53	3.65	2.97
Block Design	3.91	3.91	3.78	9.00	3.72	3.74	2.70
Object Assembly	3.75	3.75	3.53	3.72	9.61	3.57	2.42
Picture Memory	4.76	4.36	3.65	3.74	3.57	10.24	3.46
Zoo Locations	2.79	2.88	2.97	2.70	2.42	3.46	9.00

Data taken from Wechsler et al. (2012, p. 70).

9.2 Summary

In this chapter I briefly covered two advanced topics in latent variable modeling. The first was hierarchical LVMs, specifically higher-order and bi-factor models. I discussed the differences between the two types of hierarchical models and worked through an example of both types. The second topic of this chapter was bootstrapping. I explained why bootstrapping can be useful, gave a general idea of the bootstrapping process, and worked through two examples. In the first example I obtained bootstrapped confidence intervals for the mean of a non-normal variable. In the second example, I obtained bootstrapped confidence intervals for LVM model parameter estimates.

9.3 Writing the Results

Writing the results from a hierarchical LVM follows the same procedures as writing about more traditional LVMs (see Section 3.6). Path diagrams of the final model(s) are particularly useful as these models, and their various direct and indirect effects, can often be hard to understand.

9.4 Exercises

- 9.1 Watkins and Beaujean (2014) fit multiple models to the Wechsler Preschool and Primary Scale of Intelligence-Fourth Edition (WPPSI-IV) based on the covariances presented in the technical manual (Wechsler, Coalson, & Raiford, 2012). The test authors suggested there was one general factor along with three other factors: (a) *Verbal Comprehension*, made up of the Information, Receptive Vocabulary, and Picture Naming subtests; (b) *Visual Spatial*, made up of the Block Design and Object Assembly subtests; and (c) *Working Memory*, made up of the Picture Memory and Zoo Locations subtests.
 - 9.1.a Draw path diagrams of the higher-order and bi-factor LVMs. Indicate any parameter constraints by giving them the same label.
 - 9.1.b Fit a higher-order factor model using the covariances in Table 9.2.
 - 9.1.c Fit a bi-factor model using the covariances in Table 9.2.

9.5 References & Further Readings

- Brunner, M., Nagy, G., & Wilhelm, O. (2012). A tutorial on hierarchically structured constructs. *Journal of Personality*, *80*, 796-846. doi: 10.1111/j.1467-6494.2011.00749.x
- Chen, F. F., West, S. G., & Sousa, K. H. (2006). A comparison of bifactor and second-order models of quality of life. *Multivariate Behavioral Research*, *41*, 189-225. doi: 10.1207/s15327906mbr4102_5
- Parkin, J., & Beaujean, A. A. (2012). The effects of Wechsler Intelligence Scale for Children-Fourth Edition cognitive abilities on math achievement. *Journal of School Psychology*, *50*, 113-128. doi: 10.1016/j.jsp.2011.08.003
- Schmiedek, F., & Li, S.-C. (2004). Toward an alternative representation for disentangling age-associated differences in general and specific cognitive abilities. *Psychology and Aging*, *19*, 40-56. doi: 10.1037/0882-7974.19.1.40
- Watkins, M. W., & Beaujean, A. A. (2014). Bifactor structure of the Wechsler Preschool and Primary Scale of Intelligence-Fourth Edition. *School Psychology Quarterly*, *29*, 53-63. doi: 10.1037/spq0000038
- Wechsler, D. (2003). *Wechsler intelligence scale for children* (4th ed.). San Antonio, TX: The Psychological Corporation.
- Wechsler, D., Coalson, D. L., & Raiford, S. E. (2012). *Wechsler preschool and primary scale of intelligence technical and interpretive manual* (4th ed.). San Antonio, TX: The Psychological Corporation.
- Yung, Y.-F., Thissen, D., & McLeod, L. D. (1999). On the relationship between the higher-order factor model and the hierarchical factor model. *Psychometrika*, *64*, 113-128. doi: 10.1007/bf02294531

A | Measures of Model Fit

There is no area as contentious in the field of latent variable modeling as determining how well a model fits the data. New research is published in this area very frequently, so it would be relatively futile for me to indicate *the* best measures to use for assessing model fit. Instead, I present a variety of model fit measures and their general interpretation. To aid in the understanding of the fit measures, I use the model from Section 3.2 throughout this appendix. I reproduce the path model in Figure A.1. Although I leave it up to the reader to determine how to best use the measures, I provide one caution: just because a model fits the data well by some fit index, it does not mean that it is the *best* model or even a valid model. Theory, not fit measures, needs to drive the use and interpretation of LVMs.

When I present the fit measures' formulae, it is the ML version for a single group. With multiple groups or least-squares estimation the measures are similar, but not always the same.

A.1 Background

For a LVM, model fit refers to the ability of a model to reproduce the original covariance matrix. Most fit indexes only work with *overidentified* models, but some (e.g., the residual-based ones) can work with *just-identified* models, as well.

The fit measures that `lavaan` computes by default are given in Table A.1. There are two ways to obtain fit measures in `lavaan`. The first is to use the `fitMeasures()` function. The main arguments are the fitted model and fit measures to produce. For the latter, use the `fit.measures` argument with the names of the fit measures given as a character vector. The second way to obtain fit measures is to use the `summary()` function with the `fit.measures=TRUE` argument.¹

`fitMeasures()`

There are multiple ways to categorize fit indexes. I start with the χ^2 test and then present some alternative fit indexes (AFI) by family (i.e., ones that are based on similar criteria).

A.1.1 χ^2 Test and Its Variants

Latent variable models are fit to data using fitting/discrepancy functions, which examine how close the model-implied covariances (i.e., those calculated based on the model parameter estimates), **C**, are to the actual covariances calculated from the sample, **S**. The goal of the fit function is to have **C** and **S** be as close as possible, so the value returned from the fit function, f , is as small as possible.

There are a variety of fit functions, but the most common are those based on maximum likelihood (ML) for continuous variables, and least squares (usually weighted) for categorical variables. I discuss those based on the ML estimator, but there are analogs to these statistics for (weighted) least squares estimators.

¹ Another method that is “unfinished” is `lavaan::print.fit.measures(fitMeasures())`, which returns the all the fit measures with their full names.

Table A.1 Fit Measures Available in `lavaan`.

Fit Measure	Full Name
<i>Test Statistic and Related</i>	
<code>fmin</code>	Fit function value
<code>chisq</code>	χ^2 value based on the fit function
<code>df</code>	Degrees of freedom (<i>df</i>) for model
<code>pvalue</code>	<i>p</i> -value for obtained χ^2 value and <i>df</i>
<code>baseline.chisq</code>	χ^2 value for baseline model
<code>baseline.df</code>	Degrees of freedom for baseline model
<code>baseline.pvalue</code>	<i>p</i> -value for the baseline model
<code>logl</code>	Logarithm of the likelihood statistic
<code>unrestricted.logl</code>	Logarithm of the likelihood statistic for baseline model
<code>npar</code>	Number of estimated parameters in the model
<code>ntotal</code>	Total sample size
<i>Alternative Fit Indexes</i> (alphabetically organized)	
<code>agfi</code>	Adjusted Goodness-of-Fit Index
<code>aic</code>	Akaike Information Criterion
<code>bic</code>	Bayesian Information Criterion
<code>bic2</code>	Bayesian Information Criterion Adjusted for Sample Size
<code>cfi</code>	Comparative Fit Index
<code>cn_05</code>	Critical <i>n</i> for $\alpha = 0.05$
<code>cn_01</code>	Critical <i>n</i> for $\alpha = 0.01$
<code>ecvi</code>	Expected Cross-Validation Index
<code>gfi</code>	Goodness-of-Fit Index
<code>ifi</code>	Incremental Fit Index
<code>mfi</code>	McDonald Fit Index
<code>nfi</code>	Normed Fit Index
<code>nnfi</code>	Non-Normed Fit Index
<code>pgfi</code>	Parsimony Goodness-of-Fit Index
<code>pnfi</code>	Parsimony Normed Fit Index
<code>rfi</code>	Relative Fit Index
<code>rni</code>	Relative Noncentrality Index
<code>rmsea</code>	Root Mean Square Error of Approximation (RMSEA)
<code>rmsea.ci.lower</code>	Lower bound of 95% confidence interval for RMSEA
<code>rmsea.ci.upper</code>	Upper bound of 95% confidence interval for RMSEA
<code>rmsea.pvalue</code>	<i>p</i> -value associated with H_0 : RMSEA < 0.05
<code>rmr</code>	Root Mean Square Residual (includes means, if used in model)
<code>rmr_nomean</code>	Root Mean Square Residual (no means)
<code>srmr</code>	Standardized Root Mean Square Residual (includes means, if used in model)
<code>srmr_nomean</code>	Standardized Root Mean Square Residual (no means)
<code>tli</code>	Tucker-Lewis Index

n: sample size. α : Type 1 error. H_0 : Null hypothesis.

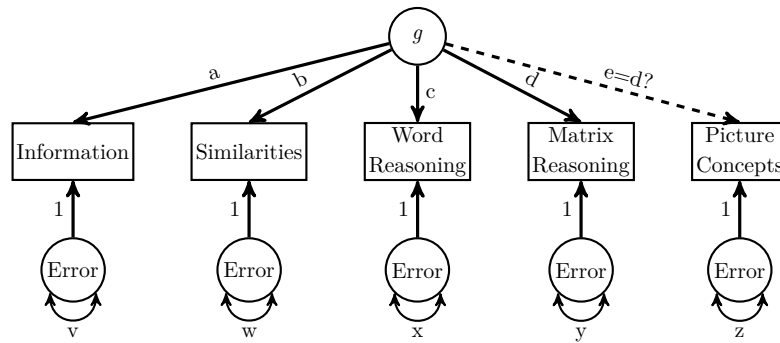


Figure A.1 Single-factor model of five Wechsler Intelligence Scale for Children-Fourth Edition subtests. The dashed line represents the parameter constraint for the nested model.

χ^2 test

Asymptotically (i.e., with really large sample sizes), the sample size multiplied by the value of the fit function produces a statistic, T , that follows a χ^2 distribution with the degrees of freedom (df) equal to the amount of non-redundant information minus the number of estimated parameters.²

```
# fit value, chi-square value of T, df, and p-value
fitMeasures(wisc4.fit, fit.measures = c("fmin", "chisq", "df", "pvalue"))

##    fmin  chisq    df pvalue
##  0.024 26.166  5.000  0.000
```

If a χ^2 value is “non-significant” (i.e., $p\text{-value} > \alpha$), this indicates that the model fits the data relatively well. If the χ^2 value is “significant” (i.e., $p\text{-value} < \alpha$), the model does not fit the data well, but it does not necessarily mean the model is not useful, as there are multiple reasons why a χ^2 value might be larger than expected (e.g., sample size, assumption violations).

Some models are variants of other models. That is, they are the same except that one or more parameters are constrained in one model (alternative/more restrictive), but not the other (baseline/less restrictive). Such models are called **nested models**. For example, if I constrained the loadings for the Matrix Reasoning and Picture Concepts to be the same in Figure A.1, then that model would be nested in the model that does not have the constraint. As another example, the hierarchy of invariance models listed in Table 4.1 are nested in each other (the models with larger numbers are nested within the models with smaller numbers). With nested models, the difference in the T statistics’ values from the more restrictive to the less restrictive model follows a χ^2 distribution with df equal to the difference in the two models’ df . In `lavaan`, two nested models can be compared using the `anova()` function.

Nested Model

The `anova()` function tests a variety of nested models in **R**, not just those from `lavaan`.

`anova()`

²Because of the way the ML fit function is defined in `lavaan`, the T is calculated slightly differently as

$$T = n \times f \times 2 \sim \chi^2$$


```
anova(wisc4.fit, wisc4.2.fit)

## Chi Square Difference Test
##
##           Df    AIC    BIC Chisq Chisq diff Df diff Pr(>Chisq)
## wisc4.fit     5 12689 12732  26.2
## wisc4.2.fit    6 12690 12729  29.4      3.21      1    0.073 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A.1.1.1 Robust χ^2 Estimates

The χ^2 test is valid only if the observations are independent, the non-standardized sample covariance matrix is used, the sample size is sufficiently large, and the manifest endogenous variables follow a multivariate normal distribution. The last assumption is usually the hardest to meet. For situations when the data do not meet the normality requirement, one option is to find a correction for the T statistic. These corrections involve using the mean or the mean and variances to adjust the fit function. Estimators that employ any of these corrections are called **robust estimators**. The test statistics derived from the robust estimators (and their associated robust standard errors) are generally more accurate than the more traditional test statistic when data are non-normal. Table A.2 gives a list of *lavaan*'s robust estimators.

Robust Estimator

Most fit measures in *lavaan* that were derived from a robust estimator have a *.scaled* suffix in the name, e.g., *chisq.scaled*.

Two of the most common ML corrections are the ones developed by Satorra and Bentler (1994) and by Yuan and Bentler (1998). For both estimators, correcting the T value for non-normality requires the estimation of a correction factor, c , which reflects the amount of average multivariate kurtosis. To obtain the corrected T value, T^* , multiply T by c . T^* follows a χ^2 distribution with df equal to the amount of non-redundant information minus the number of estimated parameters.

Testing the difference between nested models using a robust estimator requires taking the correction factor into account. The correction factor for testing the difference between the baseline (less restrictive, T_1) and nested (more restrictive, T_0) model, c_d , is

$$c_d = \frac{df_0 c_0 - df_1 c_1}{df_0 - df_1}$$

where

df_i are the degrees of freedom for model i , and
 c_i is the correction factor for model i .

The corrected difference test statistic, T_d^* is

$$T_d^* = \frac{T_0 - T_1}{c_d}$$

where

T_i is the uncorrected χ^2 value for model i .

The `anova()` function will take the correction into account when comparing models.

Table A.2 Robust Estimators Available in lavaan.

Estimator Variant	Description
	<i>Maximum Likelihood (ML)</i>
MLM	Estimation with robust standard errors and Satorra-Bentler scaled test statistic. For complete data only.
MLMVS	Estimation with robust standard errors and a mean- and variance-adjusted test statistic (Satterthwaite approach). For complete data only.
MLMV	Estimation with robust standard errors and a mean- and variance-adjusted test statistic (scale-shifted approach). For complete data only.
MLF	Estimation with standard errors based on the first-order derivatives and a conventional test statistic. For both complete and incomplete data.
MLR	Estimation with robust (Huber-White) standard errors and a scaled test statistic that is asymptotically equal to the Yuan-Bentler test statistic. For both complete and incomplete data.
	<i>Least-Squares (DWLS/ULS)</i>
WLSM ^{a,b}	Weighted least squares estimation with robust standard errors and a mean-adjusted test statistic. For complete data only.
WLSMVS ^{a,b}	Weighted least squares estimation with robust standard errors and a mean- and variance-adjusted test statistic (Satterthwaite approach). For complete data only.
WLSMV ^{a,b}	Weighted least squares estimation with robust standard errors and a mean- and variance-adjusted test statistic (scale-shifted approach). For complete data only.
ULSM	Unweighted least squares estimation with robust standard errors and a mean-adjusted test statistic. For complete data only.
ULSMVS	Unweighted least squares estimation with robust standard errors and a mean- and variance-adjusted test statistic (Satterthwaite approach). For complete data only.
ULSMV	Unweighted least squares estimation with robust standard errors and a mean- and variance-adjusted test statistic (scale-shifted approach). For complete data only.

^a For the robust weighted least squares variants (WLSM, WLSMVS, WLSMV), lavaan uses the diagonal of the weight matrix for estimation, but uses the full weight matrix to correct the standard errors and to compute the test statistic.

^b As of version 05.15 in lavaan, categorical data can have missing data using `missing="pairwise"` argument.

```
# robust ML estimators (MLM/Satorra-Bentler)
wisc4.fit.mlm <- cfa(wisc4.model, data = wisc4.data, estimator = "MLM")
wisc4.2.fit.mlm <- cfa(wisc4.2.model, data = wisc4.data, estimator = "MLM")
# compare models
anova(wisc4.fit.mlm, wisc4.2.fit.mlm)

## Scaled Chi Square Difference Test (test = satorra.bentler)
##
##              Df    AIC    BIC Chisq Chisq diff Df diff Pr(>Chisq)
## wisc4.fit.mlm    5 12699 12764   26
## wisc4.2.fit.mlm  6 12700 12761   30      3.84      1      0.05 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The T statistic (and its associated χ^2 value) has a long history of use with LVMs, but has some flaws, such as being strongly influenced by sample size—the larger the sample size, *ceteris paribus*, the more likely the χ^2 value will be statistically “significant”. Consequently a variety of **alternative fit indexes** (AFIs) have been developed to assess model fit. These AFIs are not test statistics, so they do not follow a particular probability distribution. Instead, simulation studies and rules-of-thumb tend to dictate what values to use to indicate whether a model fits the data.

Alternative Fit Index

A.2 Alternative Fit Indexes

To make their presentation easier to comprehend, I classify the alternative fit indexes (AFIs) into three different families.

1. *Incremental indexes* (sometimes called relative or comparative fit indexes).

These indexes determine model fit through the comparison of the model of interest with a baseline model. The baseline model (sometimes called the null or independence model) is a model where the *only* estimated parameters are the variances of the manifest variables (and the means or thresholds if they are part of the model). The df for the null model are $m(m-1)/2$, where m is the number of variables in the model.

2. *Parsimony indexes*

These indexes are constructed so that the model complexity is taken into account when assessing model fit. In general, models with more parameters (fewer degrees of freedom) are penalized. Sometimes these indexes use the parsimony index, which is $\frac{df}{df_0}$ where df is the model of interest’s degrees of freedom and df_0 is the degrees of freedom for the baseline model.

3. *Absolute (Standalone) indexes*

These indexes do not compare models or account for complexity. Hence, they measure the absolute fit of the model.

A.2.1 Incremental Indexes

Comparative Fit Index (CFI)

$$CFI = 1 - \frac{\max[\chi_I^2 - df_I, 0]}{\max[\chi_I^2 - df_I, \chi_B^2 - df_B, 0]}$$

where B subscripts the baseline model, I subscripts the model of interest, and $\max[]$ represents the maximum of the values within the brackets.

CFI values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "cfi")
```

```
## cfi
## 0.98
```

Normed Fit Index (NFI)/ Δ_1

$$NFI = 1 - \frac{f_B}{f_I} = 1 - \frac{\chi_B^2}{\chi_I^2}$$

where f is the value of the fit function (or χ^2 is the χ^2 based on the fit function), B subscripts the baseline model, and I subscripts the model of interest.

NFI values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "nfi")
```

```
## nfi
## 0.975
```

Parsimonious Normed Fit Index (PNFI)

$$PNFI = 1 - \frac{df_I(f_B - f_I)}{df_B(f_B)}$$

where f is the value of the fit function, df are the degrees of freedom, B subscripts the baseline model, and I subscripts the model of interest.

The PNFI is a modification of the NFI that takes parsimony of the model into account.

PNFI values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "pnfi")
```

```
## pnfi
## 0.488
```

Incremental Fit Index (IFI)/ Δ_2 /Bollen's Nonnormed Fit Index (NNFI)

$$IFI/NNFI = \frac{\chi_B^2 - \chi_I^2}{\chi_B^2 - df_I}$$

B subscripts the baseline model, I subscripts the model of interest, and df are the degrees of freedom.

A modification of the NFI to lessen the dependence on sample size. Values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "ifi")

## ifi
## 0.98
```

Relative Fit Index (RFI)/ ρ_1 /Bollen's Normed Fit Index

$$RFI = 1 - \left(\frac{\chi_I^2}{df_I} / \frac{\chi_B^2}{df_B} \right)$$

where B subscripts the baseline model, I subscripts the model on interest, and df are the degrees of freedom.

RFI values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "rfi")

## rfi
## 0.951
```

Tucker Lewis Index (TLI)/ ρ_2 /Bentler-Bonett Non-Normed Fit Index (NNFI)

$$TLI = \frac{\chi_B^2/df_B - \chi_I^2/df_I}{\chi_B^2/df_B - 1}$$

where B subscripts the baseline model, I subscripts the model on interest, and df are the degrees of freedom.

TLI/NNFI values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "tli")

## tli
## 0.96
```

A.2.2 Parsimony Indexes

Expected Cross Validation Index (ECVI)

$$ECVI = f_I + \frac{2t}{n - p - 2}$$

where I subscripts the model of interest, f is the value of the fit function, n is the sample size, p is the number of manifest variables, and t is the number of parameters estimated in the model.

The ECVI was designed to be a single-sample approximation to the cross-validation coefficient obtained from a validation sample. It is used when comparing two or more (not necessarily nested) models. Smaller values indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "ecvi")

##  ecvi
##  0.085
```

Information-Theoretic Criterion

Information-theoretic methods emphasize minimizing the amount of parameters needed in a model to explain the data. Thus, these indexes attempt to select models that are the most parsimonious/efficient representations of the observed data. They are used when comparing two or more (not necessarily nested) models, with smaller values indicating better fit. Three common information-theoretic criterion indexes are:

1. Akaike's information criterion (AIC)
2. Schwarz's Bayesian information criterion (BIC)
3. Sample-Size Adjusted Bayesian information criterion (SABIC)

$$AIC = \chi^2 + 2q$$

$$BIC = \chi^2 + q \times \ln(n)$$

$$SABIC = \chi^2 + q \times \ln((n + 2)/24)$$

where $\ln()$ is the log function, q is the number of model parameters, and n is the sample size.

lavaan's formulae are a little different than the ones above and produce larger values, but the interpretation is the same. The formulae it uses are:

$$AIC^* = -2\ln(L) + 2q$$

$$BIC^* = -2\ln(L) + q \ln(n)$$

$$SABIC^* = -2\ln(L) + q \times \ln((n + 2)/24)$$

where L is the likelihood function's value.

```
fitMeasures(wisc4.fit, fit.measures = c("aic", "bic", "bic2"))

##  aic  bic  bic2
## 12689 12732 12701
```

Noncentrality Parameter-Based Indexes

Noncentrality parameter-based indexes use the χ^2 distribution's noncentrality parameter (NCP). The NCP is defined as:

$$NCP = \chi_I^2 - df$$

where I subscripts the model of interest, and df are the degrees of freedom.

Relative Noncentrality Index (RNI)

$$RNI = 1 - \frac{\chi_I^2 - df_I}{\chi_B^2 - df_B}$$

where B subscripts the baseline model, I subscripts the model on interest, and df are the degrees of freedom.

RNI values closer to 1.0 indicating better fit.

McDonald's Noncentrality Index/Fit Index (Mc/MFI)

$$Mc/MFI = \exp(-0.5d_I)$$

where $\exp()$ is the exponentiation function, and d_I is the scaled non-centrality parameter (non-centrality index) for the model of interest, defined as:

$$d = \frac{\chi_I^2 - df}{n}$$

where I subscripts the model in interest, df are the degrees of freedom, and n is the sample size.

Mc/MFI values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "mfi")
```

```
##      mfi
## 0.981
```

Root Mean Square Error of Approximation (RMSEA)

$$RMSEA = \sqrt{\frac{\chi_I^2 - df_I}{(df_I)(n)}}$$

where I subscripts the model of interest, df are degrees of freedom, and n is the sample size.

The RMSEA assess if the specified model reasonably approximate the data (as opposed to assessing if it is an exact fit). Typically it is bounded between 0.0 and 1.0 (although it can go higher than 1.0), with values closer to 0.0 indicating better fit. Others have developed ways to calculate confidence intervals and p -values for it. The p -value (sometimes called the p of Close Fit; PCLOSE) is an one-sided test of the null hypothesis that the RMSEA equals 0.05. Thus, by rejecting the null hypothesis (i.e, p -value $> \alpha$), one can conclude that the model is “close-fitting.”

```
fitMeasures(wisc4.fit, fit.measures=c("rmsea","rmsea.ci.lower", "rmsea.ci.upper",
"rmsea.pvalue"))
```

```
##      rmsea rmsea.ci.lower rmsea.ci.upper  rmsea.pvalue
##      0.088         0.057         0.123         0.025
```

A.2.3 Absolute Fit Indexes

χ^2/df Ratio

$$\frac{\chi^2}{df}$$

This is an estimate of how many times larger the obtained χ^2 value is than the expected value (under H_0) of a χ^2 with a given number of df . Smaller values (e.g., < 2 or 3) indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "chisq")/fitMeasures(wisc4.fit, fit.measures = "df")

## chisq
## 5.23
```

Goodness of Fit Index (GFI)

$$GFI = 1 - \frac{f_I}{f_S}$$

where I subscripts the model of interest, f is the values of a fit function, and f_S uses one of the fit functions, substituting **S** for **S-C**.

The GFI compares the fit of the model of interest to a model that allows all the variables to covary. It is somewhat akin to R^2 in regression in that it attempts to indicate the proportion of the variances and covariances in **S** accounted for by the model. It should fall between 0.0 and 1.0, but negative values are possible. Values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "gfi")

## gfi
## 0.983
```

Adjusted Goodness of Fit Index (AGFI)

$$AGFI = 1 - \frac{c}{df}(1 - GFI)$$

where c is the total amount of non-redundant information in the covariance matrices (and mean vector, if used).

The AGFI is a parsimony adjustment of the GFI, similar to an adjusted R^2 used with regression models. Values should be between 0.0 and 1.0, but negative values and values larger than 1.0 are possible. Larger values indicate better fit, but values much larger than 1.0 likely indicate the model does not fit the data well.

```
fitMeasures(wisc4.fit, fit.measures = "agfi")

## agfi
## 0.948
```


Parsimony Goodness of Fit Index (PGFI)

$$PGFI = \frac{df}{df_0} \times GFI$$

where df is the model of interest's degrees of freedom and df_0 is the degrees of freedom for the baseline model.

The PGFI adjusts the GFI by multiplying by the parsimony index. Values closer to 1.0 indicate better fit.

```
fitMeasures(wisc4.fit, fit.measures = "pgfi")
```

```
## pgfi
## 0.328
```

Hoelter's Critical n (CN)

$$CN = \frac{\chi_{\alpha_0}^2}{f_I} + 1$$

where f is the value of the fit function, I subscripts the model of of interest, and $\chi_{\alpha_0}^2$ is the value of the χ^2 distribution for the df in the model at a specified α level, α_0 .

CN is an estimate of the sample size needed to reject the null hypothesis (i.e., the sample size needed for the p -value to be $< \alpha$). Values should be larger than 200.

```
fitMeasures(wisc4.fit, fit.measures = c("cn_05", "cn_01"))
```

```
## cn_05 cn_01
## 232 316
```

(Standardized) Root Mean Square Residual (SRMR/RMR)

The formula for the SRMR is cumbersome, but, syntactically, it is calculated using the following steps: (a) subtract the model-implied correlation matrix, \mathbf{C} , from the sample correlation matrix, \mathbf{S} , to produce the residual correlation matrix; (b) remove the redundant values; (c) square the remaining values; (d) sum all the squared values; (e) divide the sum by the number of non-redundant elements in the matrix; and (f) take the square root of the resulting number. SRMR values closer to 0.0 indicate better fit.

A fit measure related to the SRMR is the root mean square residual (RMR), which is calculated exactly like the SRMR, but uses covariances instead of correlations. Unlike the SRMR, it is not bounded between 0 and 1, so is mostly useful when comparing different models of the same data.

If the means are used in the model, then there are versions of the SRMR and RMR that include and exclude them. If the means are not used in the model, then both values are the same.

```
fitMeasures(wisc4.fit, fit.measures = c("srmr", "rmr", "srmr_nomean", "rmr_nomean"))
```

```
##          srmr          rmr rmr_nomean
##          0.034          0.297          0.297
```

Residuals

A special type of absolute fit index are the model's residuals (i.e, the difference between the sample covariances and model-implied covariances).

Even when one or more global fit indexes meet a recommended arbitrary criterion for acceptance, there may be clusters of residuals, which suggest and could support additional factors defined by three or more measures. Importantly, the cluster must correspond to a nameable attribute of the examinees. . . . We may declare a measurement model to be an acceptable approximation if the residuals do not support a more complex, substantively convincing model. . . . A careful examination of the residuals may reveal an unsystematic scatter across the matrix with no troublingly large values. (My students often asked me for a criterion for "troublingly large." I would offer ">.1," pointing out that a residual less than .1 would not allow a product of two salient loadings (>.3). However, this cannot be a rule for all contexts. The question is an embarrassing one.) (McDonald, 2010, p. 679)

In *lavaan*, the `residuals()` function returns the residuals. By default, it returns the unstandardized residuals. Adding the `type="cor"` argument to the function returns the residuals in a correlation metric. Alternatively, using the `type="normalized"` argument normalizes the residuals, while the `type="standardized"` argument standardizes the residuals. Both normalized and standardized residuals have a metric similar to *Z*-scores. The difference is in the term used for the denominator (for more detail, see *Standardized residuals in Mplus*, 2007).

`residuals()`

```
residuals(wisc4.fit)$cov
```

```
##          Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information      0.000
## Similarities    -0.001  0.000
## Word.Reasoning   0.040  0.066  0.000
## Matrix.Reasoning 0.124 -0.038 -0.507  0.000
## Picture.Concepts -0.293 -0.122  0.246  0.940  0.000
```

```
residuals(wisc4.fit, type = "cor")$cor
```

```
##          Infrmt Smlrts Wrd.Rs Mtrx.R Pctr.C
## Information      0.000
## Similarities      0.000  0.000
## Word.Reasoning    0.004  0.007  0.000
## Matrix.Reasoning  0.014 -0.004 -0.059  0.000
## Picture.Concepts -0.033 -0.013  0.028  0.109  0.000
```

```
residuals(wisc4.fit, type = "normalized")$cov

##              Infrmt Smlrts Wrds.Rs Mtrx.R Pctr.C
## Information      0.000
## Similarities    -0.003  0.000
## Word.Reasoning   0.086  0.145  0.000
## Matrix.Reasoning 0.298 -0.091 -1.281  0.000
## Picture.Concepts -0.716 -0.295  0.604  2.384  0.000

residuals(wisc4.fit, type = "standardized")$cov

##              Infrmt Smlrts Wrds.Rs Mtrx.R Pctr.C
## Information      0.000
## Similarities    -0.037  0.000
## Word.Reasoning   0.567  0.820  0.000
## Matrix.Reasoning 1.082 -0.329 -3.404  0.000
## Picture.Concepts -2.599 -0.901  1.210  3.510  0.000
```

lavTables()

For categorical indicator variables, the `lavTables()` function returns the observed and expected frequencies for pairwise tables of categorical variables.

Below I use the LSAT *Figure Classification* items from Chapter 6.

```
lavTables(twoP.fit)

##      id lhs rhs nobs row col obs.freq obs.prop est.prop  X2
## 1   1  Q1  Q2 1000   1   1    31    0.031    0.030 0.025
## 2   1  Q1  Q2 1000   2   1   260    0.260    0.261 0.003
## 3   1  Q1  Q2 1000   1   2    45    0.045    0.046 0.016

..<Output Omitted>..

## 38 10  Q4  Q5 1000   2   1    85    0.085    0.090 0.314
## 39 10  Q4  Q5 1000   1   2   192    0.192    0.197 0.144
## 40 10  Q4  Q5 1000   2   2   678    0.678    0.673 0.042
```

A.3 References & Further Readings

- McDonald, R. P. (2010). Structural models and the art of approximation. *Perspectives on Psychological Science*, 5, 675-686. doi: 10.1177/1745691610388766
- Satorra, A., & Bentler, P. M. (1994). Corrections to test statistics and standard errors in covariance structure analysis. In A. von Eye & C. C. Clogg (Eds.), *Latent variables analysis: Applications for developmental research* (pp. 399-419). Thousands Oaks, CA: Sage.
- Standardized residuals in Mplus*. (2007). Los Angeles, CA: Muthén and Muthén. Retrieved from <http://www.statmodel.com/download/StandardizedResiduals.pdf>.
- Yuan, K.-H., & Bentler, P. M. (1998). Robust mean and covariance structure analysis. *British Journal of Mathematical and Statistical Psychology*, 51, 63-88. doi: 10.1111/j.2044-8317.1998.tb00667.x

B | Additional R Latent Variable Model Packages

This book has used `lavaan` to fit the majority of the LVMs. `lavaan` is not the only package in **R** that can fit LVMs, however. I demonstrate how to use two other packages to fit the WISC-IV example from Chapter 3 (see Section 3.3 for the `lavaan` syntax).

B.1 sem

The `sem` (Fox, 2006) package was one of the first comprehensive latent variable modeling programs for **R**. It has a variety of ways to input models, but I only demonstrate the method that is closest to using `lavaan` path notation.

```
library(sem)
wisc4.model <- specifyModel ()
g -> Information, NA, 1
g -> Similarities, lam2
g -> Word.Reasoning, lam3
g -> Matrix.Reasoning, lam4
g -> Picture.Concepts, lam5
g <-> g, phi
```

```
wisc4.fit <- sem(wisc4.model, S = wisc4.cov, N = 550)
summary(wisc4.fit)
```

The results are identical.

B.2 MplusAutomation

Unlike the `sem` package, `MplusAutomation` (Hallquist & Wiley, 2013) requires having the *Mplus* (Muthén & Muthén, 1998-2012) program already installed. Moreover, it requires facility with *Mplus* syntax, which is not only different from `lavaan`, but is different from **R**. The *Mplus* technical manual is freely available at <http://www.statmodel.com/ugexcerpts.shtml>

Using `MplusAutomation` requires creating (or designating) a folder to store all the output. I created a folder named *MplusAutoWISC*.

Before doing any analyses in *Mplus*, I need to prepare the data in **R** and then export it as a tab-delimited plain text file (the `sep="\t"` argument tells **R** to use tab delimitation). *Mplus* does not expect any variable names in the data file. I export the correlation matrix, but I could export the raw data, instead, if I had it available.

```
library(lavaan)
WiscIV.cor <- lower2full(c(.72,.64,.51,.37,.63,.48,.38,.37,.38,.38),
diagonal=FALSE)
diag(WiscIV.cor) <- 1
write(WiscIV.cor, "MplusAutoWISC/WISCIV.DAT", sep="\t")
```

I need to write the *Mplus* syntax in a file with an *inp* extension, e.g., `file.inp`. I could write the *inp* file's syntax in an external program, but I choose to write it in **R** using the `cat()` function, and then export it as an *inp* file. The `sep="\n"` argument tells **R** to separate each group of text with the single quotations (i.e., between commas) by a new line.

`cat()`

```
library(MplusAutomation)

# write .inp file
cat(
  'TITLE: CFA of WISC-IV Data',
  'DATA: FILE IS WISCIV.dat;',
  'TYPE IS FULLCORR;',
  'NOBSERVATIONS IS 550;',
  'VARIABLE: NAMES ARE ',
  'INSS SISS WRSS MRSS PSSS;',
  'MODEL: ',
  'G BY INSS SISS WRSS MRSS PSSS;',
  'ANALYSIS:',
  'ESTIMATOR IS ML;',
  'OUTPUT:',
  'STDYX RESIDUAL;',
  file="MplusAutoWISC/WISCIV_CFA.inp", sep="\n")
```

`runModels()`

To run the *inp* file, use the `runModels()` function. The argument in this function is the address of the folder where the *inp* file resides. If there is more than one *inp* file in the same folder, `runModels()` evaluates all of them. The function saves the results from each *inp* file into a different file (within the same directly) with the same name, but uses the *out* file extension.

```
runModels("MplusAutoWISC")
```

`extractModelSummaries()`

The `extractModelSummaries()` function extracts the results (in an *.out* file) into **R**. The argument for this function is the address of the folder where the *out* file is located. If there is more than one *out* file, `runModels()` extracts the results from all of them.

```
extractModelSummaries("MplusAutoWISC")
```

`extractModelParameters()`

To import the parameter estimates into an **R** list, use the `extractModelParameters()` function, which imports the unstandardized and standardized (if they were estimated) parameter estimates. The `readModels()` function will also import the results into an **R** list, but imports more of the information in the *out* file. Type `?readModels` in **R** for a list of all the *Mplus* output it imports.

`readModels()`

```
extractModelParameters("MplusAutoWISC")

## $unstandardized
##           paramHeader param    est    se est_se pval
## 1           G.BY  INSS 1.000 0.000 999.00 999
## 2           G.BY  SISS 1.058 0.052  20.51    0

..<Output Omitted>..

## 10 Residual.Variances MRSS 0.591 0.037  16.02    0
## 11 Residual.Variances PSSS 0.774 0.035  22.42    0
```

```
readModels("MplusAutoWISC")

## Reading model: MplusAutoWISC/WISCIV_CFA.out
## $input
## $title
## [1] "CFA of WISC-IV Data"
##
## $data
## $data$file
## [1] "WISCIV.dat"
##
## $data$type
## [1] "FULLCORR"
##

..<Output Omitted>..
```

As alternative to using the `cat()` function to specify the model in *Mplus*, *lavaan* can directly output *Mplus* syntax using the `lavExport()` function.

`lavExport()`

```
WiscIV.mplus <- lavExport(WiscIV.fit, target = "Mplus", export = FALSE)
cat(WiscIV.mplus)

## TITLE:
## [This syntax is autogenerated by lavExport]
## DATA:
## file is sem.mplus.raw;

..<Output Omitted>..

## g@1;
## OUTPUT:
## sampstat standardized tech1;
```

In addition, using the `mplus2lavaan()` function, *lavaan* can read some *Mplus* files, convert the *Mplus* syntax to *lavaan* syntax, and then fit the model. For this function to work, the data and the *inp* files should be in the same directory.

`mplus2lavaan()`

```
Mplus.out <- mplus2lavaan("WISCIV_CFA.inp")
```

B.3 OpenMx

OpenMx (Boker et al., 2011) is another LVM package for **R**, built to extend the original *Mx* program (<http://www.vcu.edu/mx/>). *Mx* is a general latent variable program known for its facility in estimating models with genetically informative data (see Section 4.6). At the time of writing this book, **OpenMx** is only available for **R** version 2.15 (<http://openmx.psyc.virginia.edu>). If you have the current version of **R** loaded on your computer (which is recommended), then you need to either install a second version of **R** (2.15), or build **OpenMx** directly from source code. For instructions on how to do this, see <http://openmx.psyc.virginia.edu/wiki/howto-build-openmx-source-repository>. Fox, Byrnes, Boker, and Neale (2012) provide some examples using **OpenMx**'s syntax.

B.4 References

- Boker, S. M., Neale, M. C., Maes, H. H. M., Wilde, M., Spiegel, M., Brick, T., . . . Fox, J. (2011). OpenMx: An open source extended structural equation modeling framework. *Psychometrika*, 76, 306-317. doi: 10.1007/s11336-010-9200-6
- Fox, J. (2006). Teacher's corner: Structural equation modeling with the sem package in R. *Structural Equation Modeling: A Multidisciplinary Journal*, 13, 465-486. doi: 10.1207/s15328007sem1303_7
- Fox, J., Byrnes, J. E., Boker, S. M., & Neale, M. C. (2012). Structural equation modeling in **R** with the **sem** and **OpenMx** packages. In R. H. Hoyle (Ed.), *Handbook of structural equation modeling* (pp. 325-340). New York, NY: Guilford.
- Hallquist, M., & Wiley, J. (2013). *MplusAutomation: Automating Mplus model estimation and interpretation* [Computer program]. Retrieved from <http://CRAN.R-project.org/package=MplusAutomation>.
- Muthén, L. K., & Muthén, B. O. (1998-2012). *Mplus user's guide* (7th ed.). Los Angeles, CA: Author

C | Exercise Answers

Note. To import an external dataset, include the file location along with the file name (see Section 1.1.5.2).

Chapter 1: Introduction to R

Exercise 1.1

```
T <- rnorm(n = 100, mean = 50, sd = 10)
Z <- rnorm(n = 100, mean = 0, sd = 1)
prac1.data <- data.frame(cbind(T, Z))
```

Exercise 1.2

```
cor(prac1.data$T, prac1.data$Z)
# or
with(prac1.data, cor(T, Z))
```

Exercise 1.3

```
# Import tab-delimited file
HW.data <- read.table(file = "HeightWeight.dat", header = TRUE)
# Import comma-delimited file
HW.data <- read.csv(file = "HeightWeight.dat", header = TRUE)
head(HW.data)
```

Exercise 1.4

```
data(galton)
lm(child ~ parent, data = galton)
```

Chapter 2: Path Models and Analysis

Exercise 2.1.a

```
MathHmwk.data <- read.table("MathHmwk.txt", header = TRUE)
```

Exercise 2.1.b

```
# unstandardized regression
math.lm <- lm(MathAchievement ~ MathHomework, data = MathHmwk.data)
summary(math.lm)

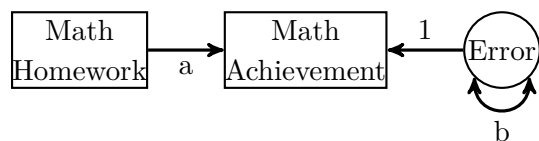
# standardized regression
math2.lm <- lm(scale(MathAchievement) ~ scale(MathHomework), data = MathHmwk.data)
summary(math2.lm)
```

Exercise 2.1.c

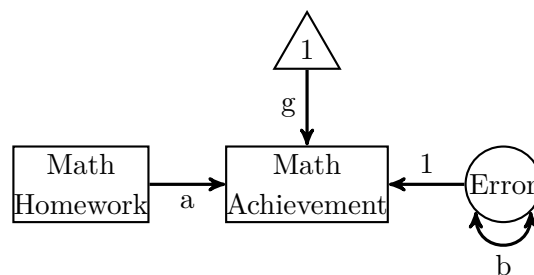
The path models are shown in Figure C.1.

Exercise 2.1.d

```
# path model
library(lavaan)
math.model <- '
MathAchievement ~ MathHomework
'
```

(a) Path model of a standardized regression.



(b) Path model of an unstandardized regression.

Figure C.1 Path models for Exercise 2.1

```
# unstandardized results with an intercept
math.fit <- sem(math.model, data = MathHmwk.data, meanstructure = TRUE)
summary(math.fit)
```

```
# standardized results without an intercept (standardized results are in the Std.all
# column)
math2.fit <- sem(math.model, data = MathHmwk.data)
summary(math2.fit, standardized = TRUE)
```

Exercise 2.2.a

```
privSchool.cor <- c(1, 0.178, 0.23, 0.106, 0.195, 1, 0.327, 0.245, 0.356, 1, 0.183, 0.721,
1, 0.178, 1)
privSchool.cor <- upper2full(privSchool.cor)
dimnames(privSchool.cor) <- list(c("Race", "SES", "CogAbil", "SchTyp", "AcadAch"), c("Race",
"SES", "CogAbil", "SchTyp", "AcadAch"))
```

Exercise 2.2.b

```
full.model <- '
AcadAch ~ j*SchTyp + g*SES + d*Race + i*CogAbil
SchTyp ~ f*SES + b*Race + h*CogAbil
CogAbil ~ e*SES + c*Race
SES ~ a*Race
'
full.fit <- sem(full.model, sample.cov=privSchool.cor, sample.nobs=18058)
```

Exercise 2.2.c

```
parameterEstimates(full.fit)
```

Path j has a value of 0.022. Since the input was a correlation matrix, this indicates that, on average, private schools' academic achievement scores are approximately 0.022 standard deviations higher than public schools' academic achievement.

Exercise 2.3.a

```
mackinnon.cov <- lower2full(c(84.85, 71.28, 140.34, 18.83, -6.25, 72.92, 60.05, 84.54, 37.18,
139.48))
rownames(mackinnon.cov) <- colnames(mackinnon.cov) <-
c("TeachExp", "SocClim", "MatCov", "StAch")
```

Exercise 2.3.b

```

mackinnon.model <- '
StAch ~ b1*SocClim + b2*MatCov + c*TeachExp
MatCov ~ a2*TeachExp
SocClim ~ a1*TeachExp
ind1 := a1*b1
ind2 := a2*b2
'

```

Exercise 2.3.c

```

mackinnon.fit <- sem(mackinnon.model, sample.cov = mackinnon.cov, sample.nobs = 40)
summary(mackinnon.fit, standardized = TRUE)

```

The indirect effect through social climate is 0.478. The indirect effect through material covered is 0.118.

Chapter 3: Basic Latent Variable Models

Exercise 3.1.a

```

psychSoc.cov <- c(0.77, 0.38, 0.65, 0.39, 0.39, 0.62, -0.25, -0.32, -0.27, 6.09)
psychSoc.cov <- lower2full(psychSoc.cov)
rownames(psychSoc.cov) <- colnames(psychSoc.cov) <- c("Dep.1", "Dep.2", "Dep.3", "SocActivity")

```

Exercise 3.1.b

```

# marker variable
marker.model <- '
PsychSocLV =~ Dep.1 + Dep.2 + Dep.3 + SocActivity
'

marker.fit <- cfa(marker.model, sample.cov=psychSoc.cov, sample.nobs=6053)

# standardized latent variable
## method 1
stdLV.fit1 <- cfa(marker.model, sample.cov=psychSoc.cov, std.lv=TRUE, sample.nobs=6053)
## method 2
stdLV.model <- '
PsychSocLV =~ NA*Dep.1 + Dep.1 + Dep.2 + Dep.3 + SocActivity
PsychSocLV~~1*PsychSocLV
'

stdLV.fit2 <- cfa(stdLV.model, sample.cov=psychSoc.cov, sample.nobs=6053)

# effects-coding
ec.model <- '
PsychSocLV =~ NA*Dep.1 + a*Dep.1 + b*Dep.2 + c*Dep.3 + d*SocActivity
a+b+c+d==4
'

ec.fit <- cfa(ec.model, sample.cov=psychSoc.cov, sample.nobs=6053)

```

Exercise 3.2.a

```

mobility.cov <- c(0.77, 0.38, 0.65, 0.39, 0.39, 0.62, -0.25, -0.32, -0.27, 6.09, 0.31,
  0.29, 0.26, -0.36, 7.67, 0.24, 0.25, 0.19, -0.18, 0.51, 1.69, -3.16, -3.56, -2.63, 6.09,
  -3.12, -4.58, 204.79, -0.92, -0.88, -0.72, 0.88, -1.49, -1.41, 16.53, 7.24)
mobility.cov <- lower2full(mobility.cov)
rownames(mobility.cov) <- colnames(mobility.cov) <- c("Dep.1", "Dep.2", "Dep.3", "SocActivity",
  "Falls", "Chronic", "TotActivity", "PersMobility")

```

Exercise 3.2.b

```
mobility.model <- '
PsychSocLV =~ Dep.1 + Dep.2 + Dep.3 + SocActivity
PsyHealthLV =~ Falls + Chronic + TotActivity
PersMobility ~ PsychSocLV + PsyHealthLV
'
mobility.fit <- sem(mobility.model, sample.cov=mobility.cov, sample.nobs=6053)
summary(mobility.fit, fit.measures=TRUE, standardized=TRUE)
```

Physical health (-0.914) appears to be a stronger predictor of personal mobility than psychosocial health (0.101).

Exercise 3.3.a

```
dda.cov <- lower2full(c(59.66, 11.18, 22.3, 2.63, 0.41, 1), diagonal = TRUE)
rownames(dda.cov) <- colnames(dda.cov) <- c("DV1", "DV2", "Cat1")
```

Exercise 3.3.b

```
dda.model<- '
F1 <- 1*DV1 + a*DV1 + b*DV2
Cat1 ~ c*F1
Cat1~~d*Cat1
'
```

Exercise 3.3.c

```
dda.fit <- sem(dda.model, sample.cov = dda.cov, sample.nobs = 288)
parameterEstimates(dda.fit, standardized = TRUE)
```

The function coefficients are 1.016 (*a*) and -0.057 (*b*), the canonical correlation is 0.341 (*c*), and R^2 is 0.116 (i.e., $1 - 0.884$)

Chapter 4: Latent Variable Models with Multiple Groups

Exercise 4.1.a

```
# WISC
WISC1.names <- c("Comp", "Arith", "Sim", "Voc", "DigSp", "PicComp", "BlkDsgn", "Cod")

WISC1_7yo.cor<-c(1, 0.31, 1, 0.36, 0.4, 1, 0.51, 0.46, 0.45, 1, 0.29, 0.4, 0.33, 0.43, 1, 0.39,
  0.29, 0.27, 0.36, 0.33, 1, 0.32, 0.27, 0.29, 0.33, 0.24, 0.28, 1, 0.22, 0.32, 0.15, 0.22,
  0.27, 0.12, 0.26, 1)
WISC1_7yo.cor<-lower2full(WISC1_7yo.cor)
rownames(WISC1_7yo.cor)<-colnames(WISC1_7yo.cor)<-WISC1.names
WISC1_7yo.mean<-c(7.83, 5.50, 5.67, 21.50, 7.67, 8.00, 6.50, 34.83)
WISC1_7yo.SD<-c(2.69, 1.50, 2.36, 6.06, 1.85, 2.18, 5.97, 9.94)
WISC1_7yo.cov<-cor2cov(WISC1_7yo.cor, WISC1_7yo.SD)
names(WISC1_7yo.mean)<-WISC1.names
names(WISC1_7yo.SD)<-WISC1.names

# WISC-R
WISC2.names<-c("Sim", "Arith", "Voc", "Comp", "DigSp", "PicComp", "BlkDsgn", "Cod")

WISC2_7yo.cor<-c(1, 0.43, 1, 0.59, 0.5, 1, 0.45, 0.43, 0.61, 1, 0.34, 0.47, 0.35, 0.34, 1, 0.36,
  0.25, 0.37, 0.33, 0.18, 1, 0.5, 0.42, 0.43, 0.43, 0.32, 0.42, 1, 0.18, 0.22, 0.27, 0.18,
  0.18, 0.13, 0.26, 1)
WISC2_7yo.cor<-lower2full(WISC2_7yo.cor)
rownames(WISC2_7yo.cor)<-colnames(WISC2_7yo.cor)<-WISC2.names
WISC2_7yo.mean<-c(8.33, 7.83, 19.67, 11.00, 8.67, 13.67, 12.00, 39.00)
```

```

WISC2_7yo.SD<-c(3.26, 1.77, 5.07, 3.63, 2.86, 4.25, 8.82, 8.05)
WISC2_7yo.cov<-cor2cov(WISC2_7yo.cor, WISC2_7yo.SD)
names(WISC2_7yo.mean)<-WISC2.names
names(WISC2_7yo.SD)<-WISC2.names

# WISC-III
WISC3.names<-c("Sim", "Arith", "Voc", "Comp", "DigSp", "PicComp", "Cod", "BlkDsgn")

WISC3_7yo.cor<-c(1.00, 0.46, 1.00, 0.64, 0.41, 1.00, 0.46, 0.33, 0.61, 1.00, 0.36, 0.39, 0.30,
  0.22, 1.00, 0.33, 0.30, 0.31, 0.23, 0.25, 1.00, 0.18, 0.22, 0.19, 0.13, 0.09, 0.24, 1.00,
  0.37, 0.36, 0.35, 0.22, 0.30, 0.47, 0.20, 1.00)
WISC3_7yo.cor<-lower2full(WISC3_7yo.cor)
rownames(WISC3_7yo.cor)<-colnames(WISC3_7yo.cor)<-WISC3.names
WISC3_7yo.mean <- c(10.00, 12.83, 17.00, 12.50, 10.50, 13.67, 43.67, 18.67)
WISC3_7yo.SD <- c(3.19, 2.19, 4.76, 3.52, 2.53, 3.85, 10.51, 10.25)
WISC3_7yo.cov<-cor2cov(WISC3_7yo.cor, WISC3_7yo.SD)
names(WISC3_7yo.mean)<-WISC3.names
names(WISC3_7yo.SD)<-WISC3.names

# WISC-IV
WISC4.names<-c("BlkDsgn", "Sim", "DigSp", "Cod", "Voc", "Comp", "PicComp", "Arith")

WISC4_7yo.cor <- c(1.00, 0.49, 1.00, 0.29, 0.37, 1.00, 0.23, 0.16, 0.12, 1.00, 0.41, 0.73, 0.33,
  0.09, 1.00, 0.33, 0.58, 0.27, 0.15, 0.63, 1.00, 0.43, 0.37, 0.13, 0.25, 0.43, 0.45, 1.00,
  0.52, 0.55, 0.51, 0.27, 0.43, 0.46, 0.38, 1.00)
WISC4_7yo.cor <- lower2full(WISC4_7yo.cor)
rownames(WISC4_7yo.cor)<-colnames(WISC4_7yo.cor)<-WISC4.names
WISC4_7yo.mean <- c(18.67, 11.83, 12.17, 45.83, 21.67, 15.17, 17.83, 15.00)
WISC4_7yo.SD <- c(9.36, 5.20, 2.72, 10.44, 6.54, 4.93, 5.35, 4.10)
WISC4_7yo.cov<-cor2cov(WISC4_7yo.cor, WISC4_7yo.SD)
names(WISC4_7yo.mean)<-WISC4.names
names(WISC4_7yo.SD)<-WISC4.names

# Combine the covarainces, means, and sample sizes
WISC_7yo.cov <- list(WISC1=WISC1_7yo.cov, WISC2=WISC2_7yo.cov, WISC3=WISC3_7yo.cov,
  WISC4=WISC4_7yo.cov)
WISC_7yo.mean <- list(WISC1=WISC1_7yo.mean, WISC2=WISC2_7yo.mean, WISC3=WISC3_7yo.mean,
  WISC4=WISC4_7yo.mean)
WISC_7yo.n <- list(WISC1=200, WISC2=200, WISC3=200, WISC4=200)

```

Exercise 4.1.b

```

# Beaujean & Sheng's final model with partial invariance
WISC_7yo.model<- '
WISC1:
g =~ col*Comp + ar1*Arith + si1*Sim + vo1*Voc + DigSp + pc1*PicComp + bd1*BlkDsgn + Cod
Comp ~~ Voc + PicComp
BlkDsgn~bd1.i*1
Comp~col.i*1
PicComp~pc1.i*1
Arith~ar1.i*1
Voc~vo1.i*1

WISC2:
g =~ co2*Comp + ar2*Arith + si2*Sim + vo2*Voc + DigSp + pc2*PicComp + BlkDsgn + Cod
DigSp ~~ Arith
BlkDsgn ~~ Voc

g~g2*1
Arith~ar2.i*1

```

```

Voc~vo2.i*1
PicComp~pc2.i*1
Comp~co2.i*1
Sim~si2.i*1

WISC3:
g =~ Comp + Arith + Sim + Voc + DigSp + PicComp + BlkDsgn + Cod
Voc ~~ Comp + Sim
PicComp ~~ BlkDsgn
Comp ~~ Sim
g~g3*1

WISC4:
g =~ Comp + ar4*Arith + si4*Sim + Voc + DigSp + pc4*PicComp + BlkDsgn + Cod
Voc ~~ Sim + Comp + PicComp
Arith ~~ DigSp
Comp ~~ Sim + PicComp + Arith
PicComp ~~ DigSp
g~g4*1
Arith~ar4.i*1
PicComp~pc4.i*1
Sim~si4.i*1
BlkDsgn~bd4.i*1
'

WISC_7yo.fit<-cfa(WISC_7yo.model, sample.cov=WISC_7yo.cov, sample.mean=WISC_7yo.mean,
meanstructure=TRUE, sample.nobs=WISC_7yo.n, std.lv=TRUE,group.equal=c("loadings","intercepts"))

```

Exercise 4.2

```

# ACE model
bmi.ace.model<- '
# genetic model
A1 =~ NA*P1 + c(a,a)*P1 + c(.5,.5)*P1
A2 =~ NA*P2 + c(a,a)*P2 + c(.5,.5)*P2
C1 =~ NA*P1 + c(c,c)*P1
C2 =~ NA*P2 + c(c,c)*P2
# variances
A1 ~~ 1*A1
A2 ~~ 1*A2
C1 ~~ 1*C1
C2 ~~ 1*C2
P1~~c(e2,e2)*P1
P2~~c(e2,e2)*P2
# covariances
A1 ~~ c(1,.5)*A2
A1 ~~ 0*C1 + 0*C2
A2 ~~ 0*C1 + 0*C2
C1 ~~ c(1,1)*C2
# constrain c to be > 0
c > 0.00001
'

bmi.ace.fit<-cfa(bmi.ace.model, sample.cov=bmi.cov, sample.nobs=bmi.n)

```

```

# CE model
bmi.ce.model<- '
# genetic Model
C1 =~ NA*P1 + c(c,c)*P1
C2 =~ NA*P2 + c(c,c)*P2
# variances

```

```

C1 ~~ 1*C1
C2 ~~ 1*C2
P1~~c(e2,e2)*P1
P2~~c(e2,e2)*P2
# covariances
C1 ~~ c(1,1)*C2
'
bmi.ce.fit<-cfa(bmi.ce.model, sample.cov=bmi.cov, sample.nobs=bmi.n)

```

Exercise 4.3.a

```

# group 1
group1.cor <- c(1.00000, 0.759, 1.000, 0.762, 0.787, 1.000, 0.0278, 0.010, -0.058,
1.000, -0.061, -0.061, -0.141, 0.785, 1.000, -0.022, -0.052, -0.102, 0.816,
0.816, 1.000)
group1.cor <- lower2full(group1.cor)
rownames(group1.cor) <- colnames(group1.cor) <- paste("I",seq(1,6),sep="")
group1.sd <- c(0.6678, 0.685, 0.707, 0.714, 0.663, 0.653)
group1.cov <- cor2cov(group1.cor, group1.sd)
group1.mean <- c(3.135, 2.991, 3.069, 1.701, 1.527, 1.545)
# group 2
group2.cor <- c(1.000, 0.813, 1.000, 0.850, 0.835, 1.000, -0.188, -0.155,
-0.215, 1.000, -0.289, -0.250, -0.338, 0.784, 1.000, -0.293, -0.210,
-0.306, 0.800, 0.832, 1.000)
group2.cor <- lower2full(group2.cor)
rownames(group2.cor) <- colnames(group2.cor) <- paste("I",seq(1,6),sep="")
group2.sd <- c(0.703, 0.718, 0.762, 0.650, 0.602, 0.614)
group2.cov <- cor2cov(group2.cor, group2.sd)
group2.mean <- c(3.073, 2.847, 2.979, 1.717, 1.580, 1.550)
# combine the data
little.cov <- list(group1=group1.cov, group2=group2.cov)
little.mean <- list(group1=group1.mean, group2=group2.mean)
little.n <- list(group1=380, group2=379)

```

Exercise 4.3.b

```

# marker variable
little.model2 <- '
F1 =~ a*I1 + b*I2 + c*I3
F2 =~ e*I5 + d*I4 + f*I6
'

## configural
little.configural2.fit <- cfa(little.model2, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n)
## weak
little.weak2.fit <- cfa(little.model2, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n,group.equal=c("loadings"))
## strong
little.strong2.fit <- cfa(little.model2, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n, group.equal=c("loadings", "intercepts"))

# standardized LV
little.configural1.model <- '
F1 =~ NA*I1 + a*I1 + b*I2 + c*I3
F2 =~ NA*I4 + d*I4 + e*I5 + f*I6
F1~~1*F1
F2~~1*F2
'

## configural
little.configural1.fit <- cfa(little.configural1.model, sample.cov=little.cov,
sample.mean=little.mean, sample.nobs=little.n)

```

```

## weak
little.weak1.model <- '
F1 =~ NA*I1 + a*I1 + b*I2 + c*I3
F2 =~ NA*I4 + d*I4 + e*I5 + f*I6
F1 ~~ c(n1,n2)*F1
F2 ~~ c(o1,o2)*F2
n1==1
o1==1
'

little.weak1.fit <- cfa(little.weak1.model, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n, group.equal=c("loadings"))

## strong
little.strong1.fit <- cfa(little.weak1.model, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n, group.equal=c("loadings", "intercepts"))

# effects coding
## configural
little.configural3.model <- '
F1 =~ NA*I1 + c(a1,a2)*I1 + c(b1,b2)*I2 + c(c1,c2)*I3
F2 =~ NA*I4 + c(d1,d2)*I4 + c(e1,e2)*I5 + c(f1,f2)*I6
# invariance constraints
F1~c(a11,a12)*1
F2~c(a121,a122)*1
I1~c(m11,m12)*1
I2~c(m21,m22)*1
I3~c(m31,m32)*1
I4~c(m41,m42)*1
I5~c(m51,m52)*1
I6~c(m61,m62)*1
# effects-coding constraints
a1+b1+c1==3
d1+e1+f1==3
a2+b2+c2==3
d2+b2+f2==3
m11+m21+m31==0
m41+m51+m61==0
m12+m22+m32==0
m42+m52+m62==0
'

little.configural3.fit <- cfa(little.configural3.model, sample.cov=little.cov,
sample.mean=little.mean, sample.nobs=little.n)

## weak
little.weak3.model <- '
F1 =~ NA*I1 + c(a,a)*I1 + c(b,b)*I2 + c(c,c)*I3
F2 =~ NA*I4 + c(d,d)*I4 + c(e,e)*I5 + c(f,f)*I6
# invariance constraints
I1~c(m11,m12)*1
I2~c(m21,m22)*1
I3~c(m31,m32)*1
I4~c(m41,m42)*1
I5~c(m51,m52)*1
I6~c(m61,m62)*1
F1~c(a11,a12)*1
F2~c(a121,a122)*1
# effects-coding constraints
a+b+c==3
d+e+f==3
m11+m21+m31==0

```

```

m41+m51+m61==0
m12+m22+m32==0
m42+m52+m62==0
,

little.weak3.fit <- cfa(little.weak3.model, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n)
little.weak3.fit

## strong
little.strong3.model <- '
F1 =~ NA*I1 + c(a,a)*I1 + c(b,b)*I2 + c(c,c)*I3
F2 =~ NA*I4 + c(d,d)*I4 + c(e,e)*I5 + c(f,f)*I6
# invariance constraints
F1~c(al11,al12)*1
F2~c(al21,al22)*1
I1~c(m1,m1)*1
I2~c(m2,m2)*1
I3~c(m3,m3)*1
I4~c(m4,m4)*1
I5~c(m5,m5)*1
I6~c(m6,m6)*1
# effects-coding constraints
a+b+c==3
d+e+f==3
m1+m2+m3==0
m4+m5+m6==0
,

little.strong3.fit <- cfa(little.strong3.model, sample.cov=little.cov, sample.mean=little.mean,
sample.nobs=little.n, group.equal=c("loadings", "intercepts"))

```

Chapter 5: Models with Multiple Time Periods

Exercise 5.1.a

```

# import data
ccawLongG0.c <- c(3.59, 3.11, 3.10, 2.91, 2.80, 2.82, 3.22, 3.05, 2.86, 3.30, 2.88, 2.63, 2.62,
2.82, 2.71)
ccawLongG0.mean <- c(11.97, 11.72, 12.03, 11.96, 12.10)
ccawLongG1.c <- c(3.42, 3.03, 3.18, 2.62, 2.73, 2.69, 2.95, 2.97, 2.59, 3.02, 2.89, 2.91, 2.67,
2.83, 3.25)
ccawLongG1.mean <- c(9.80, 12.00, 13.94, 15.96, 18.10)

# place data in matrices
library(lavaan)
ccawLongG0.cov <- lower2full(ccawLongG0.c)
ccawLongG1.cov <- lower2full(ccawLongG1.c)

# name the variables
colnames(ccawLongG0.cov) <- rownames(ccawLongG0.cov) <- paste("T",seq(1,5,1), sep="")
colnames(ccawLongG1.cov) <- rownames(ccawLongG1.cov) <- paste("T",seq(1,5,1), sep="")
names(ccawLongG0.mean) <- paste("T",seq(1,5,1), sep="")
names(ccawLongG1.mean) <- paste("T",seq(1,5,1), sep="")

# combine the data into lists
ccawLong.cov <- list(G0=ccawLongG0.cov, G1=ccawLongG1.cov)
ccawLong.mean <- list(G0=ccawLongG0.mean, G1=ccawLongG1.mean)
ccawLong.n <- list(G0=30,G1=30)

```


Exercise 5.1.b

```

# 1. intercept and residual
latGrowth0.model <- '
i =~ 1*T1 + 1*T2 + 1*T3 + 1*T4 + 1*T5
# constrain the mean of the latent intercept to be 0.0
i~~0*i
# constrain residuals to be the same across time periods
T1~~c(r,r)*T1
T2~~c(r,r)*T2
T3~~c(r,r)*T3
T4~~c(r,r)*T4
T5~~c(r,r)*T5
'

# 2. intercept and residual + random intercept
latGrowth1.model <- '
i =~ 1*T1 + 1*T2 + 1*T3 + 1*T4 + 1*T5
T1~~c(r,r)*T1
T2~~c(r,r)*T2
T3~~c(r,r)*T3
T4~~c(r,r)*T4
T5~~c(r,r)*T5
'

# 3. intercept and residual + random intercept + random slope and slope-intercept covariance
latGrowth2.model <- '
i =~ 1*T1 + 1*T2 + 1*T3 + 1*T4 + 1*T5
s =~ 0*T1 + 1*T2 + 2*T3 + 3*T4 + 4*T5
# constrain the mean of the latent slope to be 0.0
s ~ 0*1
T1~~c(r,r)*T1
T2~~c(r,r)*T2
T3~~c(r,r)*T3
T4~~c(r,r)*T4
T5~~c(r,r)*T5
'

# 4. intercept and residual + random intercept + random slope and slope-intercept covariance +
# fixed slope
latGrowth3.model <- '
i =~ 1*T1 + 1*T2 + 1*T3 + 1*T4 + 1*T5
s =~ 0*T1 + 1*T2 + 2*T3 + 3*T4 + 4*T5
T1~~c(r,r)*T1
T2~~c(r,r)*T2
T3~~c(r,r)*T3
T4~~c(r,r)*T4
T5~~c(r,r)*T5
'

# 5. intercept and residual + random intercept + random slope and slope-intercept covariance +
# fixed slope + group differences in intercept (intercept x group interaction)
latGrowth4.model <- '
i =~ 1*T1 + 1*T2 + 1*T3 + 1*T4 + 1*T5
s =~ 0*T1 + 1*T2 + 2*T3 + 3*T4 + 4*T5
# constrain the mean of the latent slope to be the same between groups
s~c(sl,sl)*1
T1~~c(r,r)*T1
T2~~c(r,r)*T2
T3~~c(r,r)*T3
T4~~c(r,r)*T4
T5~~c(r,r)*T5
'

# 6. intercept and residual + random intercept + random slope and slope-intercept covariance +

```

```
# fixed slope + group differences in intercept and slope (slope x group interaction)
latGrowth5.model <- '
i =~ 1*T1 + 1*T2 + 1*T3 + 1*T4 + 1*T5
s =~ 0*T1 + 1*T2 + 2*T3 + 3*T4 + 4*T5
T1~~c(r,r)*T1
T2~~c(r,r)*T2
T3~~c(r,r)*T3
T4~~c(r,r)*T4
T5~~c(r,r)*T5
'
```

Exercise 5.1.c

```
latGrowth0.fit <- growth(latGrowth0.model, sample.cov=ccawLong.cov, sample.mean=ccawLong.mean,
sample.nobs=ccawLong.n, group.equal=c("means"))

latGrowth1.fit <- growth(latGrowth1.model, sample.cov=ccawLong.cov, sample.mean=ccawLong.mean,
sample.nobs=ccawLong.n, group.equal=c("means", "lv.variances"))

latGrowth2.fit <- growth(latGrowth2.model, sample.cov=ccawLong.cov, sample.mean=ccawLong.mean,
sample.nobs=ccawLong.n, group.equal=c("means", "lv.variances", "lv.covariances"))

latGrowth3.fit <- growth(latGrowth3.model, sample.cov=ccawLong.cov, sample.mean=ccawLong.mean,
sample.nobs=ccawLong.n, group.equal=c("means", "lv.variances", "lv.covariances"))

latGrowth4.fit <- growth(latGrowth4.model, sample.cov=ccawLong.cov, sample.mean=ccawLong.mean,
sample.nobs=ccawLong.n, group.equal=c("lv.variances", "lv.covariances"))

latGrowth5.fit <- growth(latGrowth5.model, sample.cov=ccawLong.cov, sample.mean=ccawLong.mean,
sample.nobs=ccawLong.n, group.equal=c("lv.variances", "lv.covariances"))
```

Exercise 5.2.a

```
# reshape data
voelkleLong.data <- reshape(voelkle.data, direction = 'long',
varying = list(c('x1', 'x2', 'x3', 'x4')), v.name="X")
# make the time variable a categorical variable (factor)
voelkleLong.data$time <- factor(voelkleLong.data$time)
# make the id variable a categorical variable (factor)
voelkleLong.data$id <- factor(voelkleLong.data$id)

# repeated measures ANOVA
rmANOVA.fit <- aov(X ~ time + Error(id/time), data=voelkleLong.data)
summary(rmANOVA.fit)
```

Exercise 5.2.b

```
# latent growth model version of repeated measures ANOVA
lgcANOVA.model <- '
# loadings
i =~ 1*x1 + 1*x2 + 1*x3 + 1*x4
s1 =~ -.671*x1 + -.224*x2 + .224*x3 + .671*x4
s2 =~ .5*x1 + -.5*x2 + -.5*x3 + .5*x4
s3 =~ -.224*x1 + .671*x2 + -.671*x3 + .224*x4
# variances
i ~~ a*i
s1 ~~ b*s1
s2 ~~ c*s2
s3 ~~ d*s3
# covariances
s3~~ e*i + f*s1 + g*s2
```

```

s2 ~~ h*i + i*s1
s1 ~~ j*i
# intercepts
i~k*1
s1~l*1
s2~m*1
s3~n*1
# residual variances
x1~~0*x1
x2~~0*x2
x3~~0*x3
x4~~0*x4
'

lgcANOVA.fit <- growth(lgcANOVA.model, data=voelkle.data)
summary(lgcANOVA.fit)

```

Table C.1 Repeated Measures ANOVA Source Table.

Source	SS	df	MS
Within (Time)	236.514	3	78.838
Within (Error)	52.345	102	0.513
Within Subjects	288.858	105	2.751
Between Subjects	85.356	34	2.51
Total	374.214	139	2.692

SS: Sum of squares; *df*: Degrees of freedom; MS: Mean squares.

To show the repeated measures ANOVA (RM-ANOVA) and the LCM model specified in Figure 5.6 return the same results, I first made a source table for the RM-ANOVA in Table C.1. Now, I just need to show the results from the LCM match those in Table C.1. The sum of squares within (time) ($SS_{\text{within (time)}}$) is $n \times (t - 1) \times \sigma_x^2$, where σ_x^2 is the variance of the t means. Thus, $SS_{\text{within (time)}} = 35 \times 3 \times 2.253 = 236.514$. The $SS_{\text{within (error)}}$ is $n \times \left(\sum_{j=1}^k \sigma_{sj}^2 \right)$, where σ_{sj}^2 is the variance of the j th slope. Thus, $SS_{\text{within (error)}} = 35(1.063 + 0.255 + 0.177) = 52.308$. The $SS_{\text{between subjects}}$ is $n \times t \times \sigma_i^2$, where σ_i^2 is the variance of the latent intercept. Thus, $SS_{\text{between}} = 35 \times 4 \times 0.61 = 85.356$. The other SS values in Table C.1 are derive from the ones I have already calculated. The $SS_{\text{within subjects}} = SS_{\text{within (time)}} + SS_{\text{within (error)}}$, and the $SS_{\text{total}} = SS_{\text{within subjects}} + SS_{\text{between subjects}}$.

Exercise 5.2.c

```

lcm.model <- '
i =~ 1*x1 + 1*x2 + 1*x3 + 1*x4
s =~ 0*x1 + 1*x2 + 2*x3 + 3*x4
'

lcm.fit <- growth(lcm.model, data=voelkle.data)
summary(lcm.fit)

```

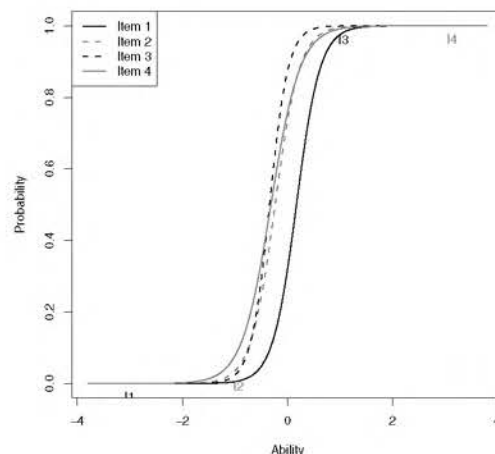


Figure C.2 Abortion data item characteristic curves for Exercise 6.1.d.

Chapter 6: Models with Dichotomous Indicator Variables

Exercise 6.1.a

```
library(ltm)
data(Abortion)
# rename items
names(Abortion) <- paste("I", 1:4, sep = "")
```

Exercise 6.1.b

```
colMeans(Abortion)
```

Exercise 6.1.c

```
# logistic
library(ltm)
abortionL.irt <- ltm(Abortion ~ z1, IRT.param = TRUE)
coef(abortionL.irt)
# normal
library(psych)
abortionN.irt <- irt.fa(Abortion, plot = FALSE)
abortionN.irt$irt
```

Exercise 6.1.d

```
plot(abortionL.irt)
```

The ICCs are shown in Figure C.2.

Exercise 6.1.e

```
# logistic
library(ltm)
abortionL.si <- ltm(Abortion ~ z1, IRT.param = FALSE)
coef(abortionL.si)
# normal
library(psych)
abortionN.si <- irt.fa(Abortion, plot = FALSE)
abortionN.si$fa
```

Exercise 6.1.f

```
library(lavaan)
abortion.model <- '
LV =~ I1 + I2 + I3 + I4
'
```

```
abortion.fit <- cfa(abortion.model, data=Abortion, std.lv=TRUE,
ordered=paste("I",seq(1:4), sep=""))
summary(abortion.fit)
```

Exercise 6.2.a

```
gss.data <- read.table("gss2000.dat", header = TRUE)
```

Exercise 6.2.b & Exercise 6.2.c

```
gssConversion.model<- '
# loadings
Theta =~ l1*word.a + l2*word.b + l3*word.c + l4*word.d + l5*word.e + l6*word.f + l7*word.g +
      l8*word.h + l9*word.i + l10*word.j
# thresholds
word.a | th1*t1
word.b | th2*t1
word.c | th3*t1
word.d | th4*t1
word.e | th5*t1
word.f | th6*t1
word.g | th7*t1
word.h | th8*t1
word.i | th9*t1
word.j | th10*t1
# convert loadings to slopes (logistic scale)
alphaa.L := (l1)/sqrt(1-l1^2)*1.7
alphab.L := (l2)/sqrt(1-l2^2)*1.7
alphac.L := (l3)/sqrt(1-l3^2)*1.7
alphad.L := (l4)/sqrt(1-l4^2)*1.7
alphae.L := (l5)/sqrt(1-l5^2)*1.7
alphaf.L := (l6)/sqrt(1-l6^2)*1.7
alphag.L := (l7)/sqrt(1-l7^2)*1.7
alphah.L := (l8)/sqrt(1-l8^2)*1.7
alphai.L := (l9)/sqrt(1-l9^2)*1.7
alphaj.L := (l10)/sqrt(1-l10^2)*1.7
# convert thresholds to locations (logistic scale)
loca.L := th1/l1
locb.L := th2/l2
locc.L := th3/l3
locd.L := th4/l4
loce.L := th5/l5
locf.L := th6/l6
locg.L := th7/l7
loch.L := th8/l8
loci.L := th9/l9
locj.L := th10/l10
'

gssConversion.fit <- cfa(gssConversion.model, data = gss.data, std.lv = TRUE,
ordered = paste("word.",letters[1:10], sep=""))
summary(gssConversion.fit)
```

The first item, *word.a*, is the easiest and the last item, *word.j*, is the most difficult.

Exercise 6.2.d

```
library(ltm)
gss.IRT <- ltm(gss.data ~ z1, IRT.param = TRUE)
coef(gss.IRT)
```

Chapter 7: Models with Missing Data

Exercise 7.1.a

```
mcar.data <- read.table("mcar.dat", sep = ",", header = TRUE)
mar.data <- read.table("mar.dat", sep = ",", header = TRUE)
nmar.data <- read.table("nmar.dat", sep = ",", header = TRUE)
```

Exercise 7.1.b

```
library(BaylorEdPsych)
library(MissMech)

LittleMCAR(mcar.data)
TestMCARNormality(mcar.data)

LittleMCAR(mar.data)
TestMCARNormality(mar.data)

LittleMCAR(nmar.data)
TestMCARNormality(nmar.data)
```

Exercise 7.1.c

```
complete.model <- '
read =~ a*z1 + b*z2 + c*z3
read ~ g*x1

z1~~d*z1
z2~~e*z2
z3~~f*z3
read~~h*read
'

mcarListwise.fit <- sem(complete.model, data=mcar.data)
marListwise.fit <- sem(complete.model, data=mar.data)
nmarListwise.fit <- sem(complete.model, data=nmar.data)
```

Exercise 7.1.d

```
mcarFIML.fit <- sem(complete.model, data=mcar.data, missing='fiml')
marFIML.fit <- sem(complete.model, data=mar.data, missing='fiml')
nmarFIML.fit <- sem(complete.model, data=nmar.data, missing='fiml')
```

Exercise 7.1.e

```
library(semTools)
mcarFIMLAux.fit <- auxiliary(complete.model, aux="x2", data=mcar.data, fun="sem")
marFIMLAux.fit <- auxiliary(complete.model, aux="x2", data=mar.data, fun="sem")
nmarFIMLAux.fit <- auxiliary(complete.model, aux="x2", data=nmar.data, fun="sem")
```

Exercise 7.1.f

```
library(semTools)
mcarMI.fit <- runMI(complete.model, data=mcar.data, m=20, miPackage="mice", fun="sem")
marMI.fit <- runMI(complete.model, data=mar.data, m=20, miPackage="mice", fun="sem")
nmarMI.fit <- runMI(complete.model, data=nmar.data, m=20, miPackage="mice", fun="sem")
```

Exercise 7.2.a

```
eating.data <- read.table("eatingattitudes.dat")
# remove first variable
eating.data$id <- NULL
```

Exercise 7.2.b

```
library(mice)
md.pattern(eating.data)
```

Exercise 7.2.c

```
library(BaylorEdPsych)
library(MissMech)

LittleMCAR(eating.data)
TestMCARNormality(eating.data)
```

The missing values do not appear to be MCAR.

Exercise 7.2.d

```
library(lavaan)
eating.model <- '
drive =~ eat1 + eat2 + eat10 + eat11 + eat12 + eat14 + eat24
foodpre =~ eat3 + eat18 + eat21
'

eatingListwise.fit <- cfa(eating.model, data=eating.data)
summary(eatingListwise.fit, standardized=TRUE)
```

There are 267 observations using listwise deletion.

Exercise 7.2.e

```
eatingFIML.fit <- cfa(eating.model, data = eating.data, missing = "fiml")
summary(eatingFIML.fit, standardized = TRUE)
```

Exercise 7.2.f

```
library(semTools)
eatingFIMLAux.fit <- auxiliary(eating.model, aux = c("bmi", "wsb", "anx"), data = eating.data,
  fun = "sem")
summary(eatingFIMLAux.fit, standardized = TRUE)
```

Exercise 7.2.g

```
library(semTools)
# make the eating items categorical for the imputation process
items <- paste("eat", c(1, 2, 10, 11, 12, 14, 24, 3, 18, 21), sep = "")
eating.data[, items] <- lapply(eating.data[, items], as.factor)

# create imputed dataset via Amelia
library(Amelia)
eating.amelia <- amelia(eating.data, m = 20, ords = paste("eat", c(1, 2, 10, 11, 12, 14, 24,
  3, 18, 21), sep = ""))

eatAmelia.fit <- runMI(eating.model, data = eating.amelia$imputations, fun = "cfa")
summary(eatAmelia.fit)

# create imputed dataset via mice
library(mice)
eating.mice <- mice(eating.data, m = 20, meth = c(rep("polyreg", 10), rep("pmm", 3)))

# place all the imputed mice datasets into a single list
eating.mice.data <- NULL
for (i in 1:20) {
  eating.mice.data[[i]] <- complete(x = eating.mice, action = i, include = FALSE)
```

```

}

eatMice.fit <- runMI(eating.model, data = eating.mice.data, fun = "cfa")
summary(eatMice.fit, standardized = TRUE)

```

Chapter 8: Sample Size Planning

Exercise 8.1.a

If $g = 0.40$, then Y 's variance is $0.40 \times 0.25 \times 0.40 + 1 \times 0.80 \times 1 = 0.84$. That makes

$$d = \frac{0.15}{\sqrt{0.84}} = 0.436$$

Exercise 8.1.b

```

# data generating model
read.pop.model <- '
read =~ 1*z1 + 0.90*z2 + 0.60*z3
# regression
read ~ g*x1 + 0.40*x1
# means and variances
x1~0.5*1
z1~0*1
z2~0*1
z3~0*1
# variances
x1~~0.25*x1
z1~~0.80*z1
z2~~0.80*z2
z3~~0.40*z3
read~~0.80*read
'

# analysis model
read.model <- '
read =~ z1 + z2 + z3
read ~ g*x1
'

# population moments
read.fit<-sem(read.pop.model, do.fit=FALSE, fixed.x=FALSE)
summary(read.fit,rsquare=TRUE)
fitted(read.fit)

# monte carlo simulation
read.n <- sim(model=read.model, n =rep(seq(100,300,25), 500), generate=read.pop.model,
lavaanfun = "sem", multicore=TRUE)

# power curve
plotPower(read.n, alpha=0.05, powerParam=c("g"))
abline(h=.8, lwd=2, lty=2)

```

The power curve is shown in Figure C.3.

Exercise 8.1.c

According to the power curve, there should be enough power when $n \approx 220$.

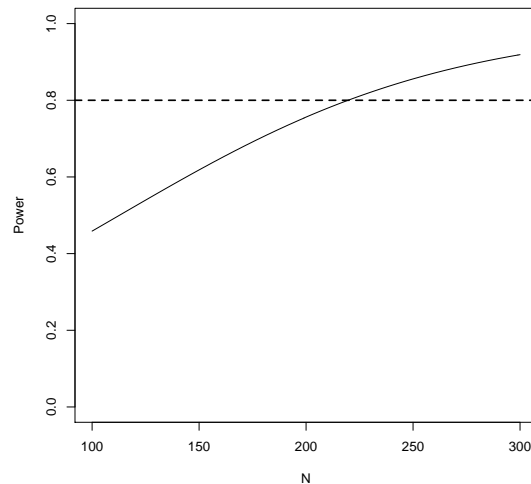


Figure C.3 Power curve for Exercise 8.1.

Exercise 8.1.d

```
read1.power <- sim(nRep=10000, model=read.model, n =220, generate=read.pop.model,
lavaanfun = "sem", multicore=TRUE, seed=0)

read2.power <- sim(nRep=10000, model=read.model, n =220, generate=read.pop.model,
lavaanfun = "sem", multicore=TRUE, seed=10458)

summaryParam(read1.power,alpha = 0.05,detail=TRUE)
summaryParam(read2.power,alpha = 0.05,detail=TRUE)
```

The bias and coverage estimates are within the specified ranges for both studies. Power ≈ 0.80 in both studies.

Exercise 8.1.e

Power ≈ 0.65 for $n = 220$ with the specified missing data percentages using FIML estimation.

Exercise 8.1.f

Power ≈ 0.77 for $n = 220$ with the missing data using MI ($m = 10$).

Chapter 9: Hierarchical Latent Variable Models

Exercise 9.1.a

See Figure C.4.

Exercise 9.1.b

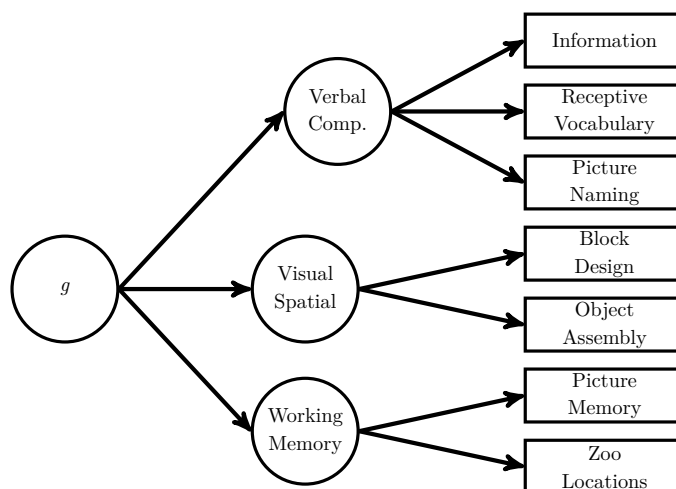
```
wppsiH0.model<-'
VC =~ IN + RV + PN
VS =~ BD + OA
WM =~ PM + ZL
g =~ NA*VC + VC + VS + WM
g~~1*g
'

wppsiH0.fit <- cfa(wppsiH0.model, sample.cov=WPPSI2_3.cov,
sample.nobs=600)
summary(wppsiH0.fit)
```

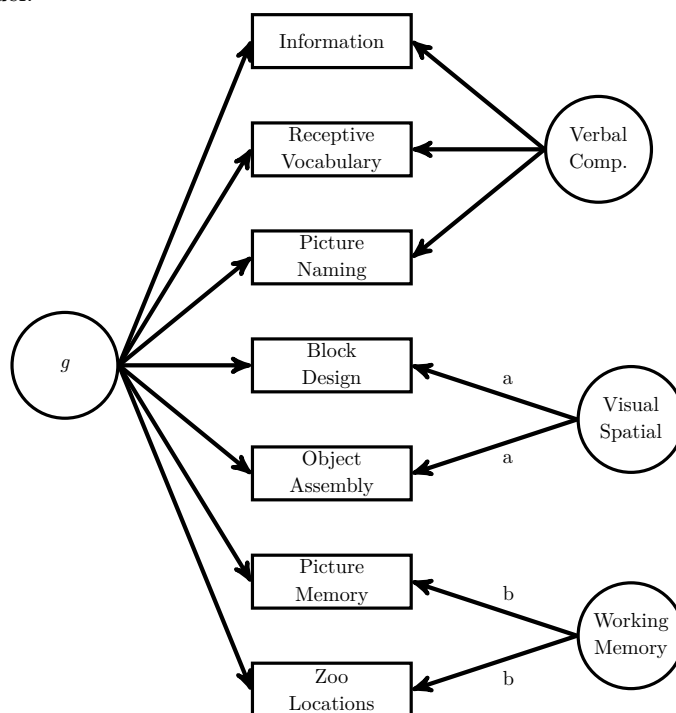
Exercise 9.1.c

```
wppsiBF.model<- '
g =~ IN + RV + PN + BD + OA + PM + ZL
VC =~ RV+ IN + PN
VS =~ a*BD + a*OA
WM =~ b*PM + b*ZL
'

wppsiBF.fit<-cfa(wppsiBF.model, sample.cov=WPPSI2_3.cov,
sample.nobs=600, orthogonal=TRUE, std.lv=TRUE)
summary(wppsiBF.fit)
```



(a) Higher-order model.



(b) Bi-factor model. Paths with the same label are constrained to be equal.

Figure C.4 Hierarchical latent variable models for 2:6-3:11 year-old group of WPPSI-IV norming data. Errors terms not shown.

Glossary

Italicized words indicate other terms in glossary.

Alternative fit index (AFI) Measure of how a latent variable model fits a given dataset that is an alternative to the traditional χ^2 -distributed statistic. Sometimes called practical fit index.

Auxiliary variable (AV) A variable that is not of interest in answering a research question, but is included in the model because it is either a potential cause or correlate of missingness in the variables of interest.

Behavior genetics (BG) Academic field that examines the genetic and/or environmental influences on physical or behavioral traits (i.e., a phenotype) by using genetically informative research designs.

Bi-factor latent variable model A type of hierarchical *latent variable model* comprised of *first-order latent variables*. The general *latent variable* influences all the *manifest variables*, while the *domain-specific latent variables* only influence a subset of the manifest variables. Sometimes called a nested-factors or direct hierarchical model.

Bias In a *Monte Carlo Study*, the difference between the actual parameter value and the average of all the estimated values. See also *relative bias*.

Communality The proportion of an *indicator variable's* variance that is due to the *latent variables* in the model. It is calculated as the sum of the squared standardized *loadings* for all latent variables that influence a given variable.

Conditional parameterization *Item factor analysis* parameterization that constrains the *underlying variables' error* variances to be 1.0, and typically constrains the thresholds to their values when estimated using the *marginal parameterization*. The variance of the *underlying variables* are calculated from the *loadings* and the *latent variable's* variance. Sometimes called the theta or unstandardized parameterization.

Configural invariance First step in assessing *measurement invariance*. It requires the *indicator variables* form the same *latent variable* structure (i.e., number of LVs, number of indicator variables, pattern of constrained and estimated parameters) in all groups.

Confirmatory factor analysis (CFA) A statistical technique for data reduction that specifies the number, meaning, and associations of the *latent variables*, as well as the pattern of the estimated *factor loadings*, before analyzing the data. See *latent variable model*.

Coverage In a *Monte Carlo Study*, the proportion of confidence intervals, across all the simulated datasets, that contain the actual parameter value. It is a measure of standard error accuracy.

Degrees of freedom (*df*) In a *latent variable model*, it is the difference between the number of non-redundant pieces of information in the data and the number of parameters to estimate. Will only be > 0 for *overidentified models*.

Domain-specific latent variable In a *bi-factor latent variable model*, they are the *latent variables* that only influence a portion of the *manifest variables*.

Empirical underidentification Situation when a *latent variable model* is only identified if one of its parameters (usually a latent correlation or *loading*) is not equal to 0.0 or 1.0. See *identification of latent variable*.

Endogenous variable A variable that has at least one direct cause within a specific model. In regression, it is sometimes called an outcome, criterion, or downstream variable, while with an experimental design it is called dependent variable.

Error The discrepancy between an observations's actual value on a variable and the value predicted by the other variables in the model. The discrepancies' variability represents the amount of variance in an *endogenous variable* not explained by the other variables in the model. Sometimes called a residual or disturbance..

Exogenous variable A variable that has no direct cause within a specific model. In regression, it is sometimes called a predictor, source, or upstream variable, while in an experimental design it is called an independent variable.

Exploratory factor analysis (EFA) A statistical method for reducing data whereby *factors* (i.e. *latent variables*) are extracted from a dataset without specifying the number and pattern of *factor loadings* between the *manifest variables* and the factors.

Factor A *latent variable*.

Factor loading The relationship between an *indicator variable* and a *latent variable*. With *reflective latent variables*, they are interpreted as regression coefficients (i.e, how much does the indicator variable increase as the latent variable increases a single unit). The larger the absolute value of the loading, the more relevant the indicator variable is in defining the latent variable.

First-order latent variable (FOLV) *Latent variables with manifest indicator variables*. See also *higher-order latent variable model* and *bi-factor latent variable model*.

Formative latent variable A *latent variable* that is the weighted sum of the indicator variables. Unlike a *reflective latent variable*, a formative latent variable model posits that the latent variable is the result of the indicator variables. Sometimes called an emergent factor.

Full information maximum likelihood (FIML) Type of maximum likelihood parameter estimation technique that finds the most likely parameter estimates for each observation, with the final model estimates being the ones that are the most likely across all observations. One modern method for handling missing data.

Higher-order latent variable model A type of hierarchical *latent variable model* where *first-order latent variables* serve as *indicator variables* to form *second-order latent variables*. Sometimes called an indirect hierarchical or second-order latent variable model.

Identification of latent variable Determination of whether a model's parameters can be uniquely determined from the amount of non-redundant information in the data (i.e., covariances, variances, and, when applicable, means).

Indicator variable In reflective *latent variable models*, indicator variables are *manifest variables* that are directly influenced by one or more *latent variables* and an *error* term.

Indirect effect The influence of one variable on a second variable through a third (or fourth, or fifth, ...) variable. The indirect variables are mediating variables.

- Item characteristic curve** Graphical representation of an *item response theory* model that shows the probability of endorsing a given response to an item (e.g., answering correctly) as a function of the *latent variable*.
- Item discrimination** In *item response theory* models, it is the measure of both how well an item measures the *latent variable* and how well an item differentiates between examinees having abilities below and above the *item location*.
- Item factor analysis (IFA)** *Latent variable model* with categorical *indicator variables*. They can be parameterized to give results equivalent to models based on *item response theory*.
- Item location** In an *item response theory* model, it is a description of where the item best functions along the *latent variable*'s scale. Sometimes called item difficulty.
- Item response theory (IRT)** Broad class of *latent variable models* designed to describe the probability of a response to a categorical *indicator variable*, conditional on the level of the *latent variable*.
- Just-identified model** A model with the same number of parameters to estimate as there is non-redundant information in the data. Thus, the degrees of freedom are always zero. Sometimes called a saturated model. See *identification of latent variable*.
- Latent curve model (LCM)** *Latent variable model* used to model change over time for one or more variables measured on the same individuals. Sometimes called a latent growth curve or latent trajectory model.
- Latent variable (LV)** A variable for which values cannot be directly observed. Its measurement has to be inferred from values on one or more *manifest variables*. Sometimes called a factor or a construct.
- Latent variable model (LVM)** A statistical model that relates a set of *manifest variables* to one or more *latent variables*. Sometimes called a measurement model, factor analysis, or confirmatory/exploratory factor analysis.
- Manifest variable (MV)** A variable that can be directly observed and measured.
- Marginal parameterization** *Item factor analysis* parameterization that standardizes the marginal (unconditional) distribution of the *underlying variables*, which then makes the scaling factor equal 1.0. The *underlying variables*' error variances are calculated from the *loadings* and the *latent variable*'s variance. Sometimes called the delta or standardized parameterization.
- Mean and covariance structure (MACS)** Extension of typical *latent variable model* whereby mean-level information is analyzed (structured) in addition to the covariances.
- Measurement invariance** When the relationship between the *indicator variables* and the *latent variables* does not differ among two or more groups.
- Missing at random (MAR)** When missing values on a given variable are unrelated to the variable itself, but are related to other variables. For the parameter estimates to be unbiased, the variables related to the missingness must be included in the model.
- Missing completely at random (MCAR)** When missing values on a given variable are unrelated to both that variable and any other variable.
- Missing data pattern** Configuration of observed and missing values in a dataset.
- Monte Carlo Study (MC)** An experiment in which variable or model properties are investigated by using purposefully simulated data.

Multiple imputation (MI) Method for handling missing data that creates multiple datasets, each of which contain different plausible estimates of the missing values.

Multiple indicators multiple independent causes model (MIMIC) *Latent variable model* with at least one *exogenous manifest variable* (the independent causal indicator[s]) predicting an *endogenous latent variable*.

Nested model A model that estimates a subset of another model's parameters, and thus has fewer *degrees of freedom*.

Not missing at random (NMAR) When missing values on a given variable are related to the variable itself, even after including all possible covariates in the model. Sometimes called non-ignorable missingness or missing not at random.

Overidentified model A model with fewer parameters to estimate than non-redundant information in the data. See *identification of latent variable*.

Partial invariance Model where a *measurement invariance* constraint is not warranted for only a few of the parameter estimates within an invariance level, but is warranted for all the others.

Path coefficient A measure of the direct effect of one variable on another variable. Usually assumed to be standardized (i.e., variables transformed to *Z* scores) unless otherwise stated. Sometimes called regression coefficients, especially if *latent variables* are not included in the *path model*.

Path model An diagram of variable relationships that uses geometric figures to represent variable types, and arrows to represent variable relationships. Single-headed arrows show direct relationships, whereas double-headed arrows indicate a non-directional (i.e., covariance) relationships. Sometimes called path diagram or path figure.

Pattern coefficient See *factor loading*.

Power The frequency of concluding the data, taken from multiple samples of the same population, indicate a variable has an effect when there is one in the population. In a *Monte Carlo Study*, it is the proportion of the simulated datasets for which the null hypothesis is rejected for a parameter estimate at a given type 1 error rate (i.e., how often $p\text{-value} < \alpha$).

Random effects A model parameter that can vary across observations, which allows it to have unexplained (i.e., *error*) variance.

Reflective latent variable *Latent variable model* where the *indicator variables* are dependent on, or the results of, the *latent variable*. Sometimes called an effect indicator model.

Relative bias A method to make *bias* estimates from a *Monte Carlo Study* comparable by dividing the bias estimate by the known population value. The resulting values are in proportions.

Robust estimator Parameter estimators whose values are less influenced by assumption violations than the traditional maximum likelihood estimator.

Second-order latent variable (SOLV) In a *higher-order latent variable model*, they are *latent variables* formed by using other latent variables as *indicator variables*.

Specific factor In a *higher-order latent variable model*, the part of a *first-order latent variable* not explained by the *second-order latent variable*.

- Strict invariance** Fourth step in assessing *measurement invariance*, although some consider it an optional step. It requires that *indicator variables'* error variances are the same across groups.
- Strong invariance** Third step in assessing *measurement invariance*. It requires that *indicator variables'* intercepts are the same across groups. Sometimes called scalar invariance.
- Structural equation model (SEM)** A model that consists of *manifest variables* and *latent variables*, as well as a structural model (*path model*) specifying their relationships. Sometimes called a simultaneous equation model, linear structural relationship model, or covariance structure analysis.
- Structural invariance** Assess if latent variances, covariance, or means are the same across groups. Can only be assessed with models that at least have *strong invariance*.
- Structure coefficient** Correlation of an *indicator variable* with a *latent variable*. Only when the *latent variables* are uncorrelated, or there is one *latent variable*, do the structure coefficients and *pattern coefficients* (*factor loadings*) have the same values.
- Tetrachoric correlation** Correlation between two dichotomous variables, each of which are coarse measures of a normally-distributed *underlying variable* and, together, have a bivariate-normal distribution. The resulting value approximates what the Pearson correlation would have been between the variables if they were measures on a continuous scale.
- Time-invariant covariate** Covariate for a *latent curve model* whose values are stable over all the data collection time periods.
- Time-varying covariate** Covariate for a *latent curve model* whose values can vary across the data collection time periods.
- Tracing rules** Standard rules for reading paths in a *path model* that yield the expected variances and covariances among the model's variables. Sometimes called Wright's rules, after Sewall Wright.
- Underidentified model** A model with more parameters to estimate than non-redundant information in the data. See *identification of latent variable*.
- Underlying variable** Continuous *latent variable*, often assumed to follow normal distribution, that underlies a dichotomous variable in *item factor analysis* models. Categorical *indicator variables* (i.e., items) are coarse measurements of the underlying variable.
- Uniqueness** The proportion of an *indicator variable's* variance that is not explained by the *latent variables* in the model. It is calculated as one minus the *communality*.
- Weak invariance** Second step in assessing *measurement invariance*. It requires that *indicator variables'* loadings are the same across groups. Sometimes called pattern or metric invariance.

Author Index

- Adler, M. A., 130
Aiken, L. S., 89, 92
American Psychological Association Science
 Directorate Task Force on Statistical
 Inference, 118, 130
Asendorpf, J. B., 33, 36
Asparouhov, T., 130

Bagozzi, R. P., 55
Baker, F. B., 113
Barrett, P., 133, 144
Bartholomew, D., 111, 113
Bates, T., 169, 170
Bauer, D. J., 113
Bauman, S., 128, 130
Baumgartner, H., 78
Beauducel, A., 113
Beaujean, A. A., 20, 31, 37, 43, 52, 53, 55,
 62, 65, 75, 78, 111, 113, 148, 151,
 152
Bentler, P. M., 156, 166
Bock, R. D., 94, 113
Bohr, Y., 129, 130
Boker, S. M., 36, 169, 170
Bolger, N., 55
Bollen, K. A., 38, 55, 80, 81, 84, 88, 92
Boomsma, A., 51, 55, 75, 78
Borenstein, M., 143, 144
Braddy, P. W., 61, 78
Brick, T., 169, 170
Briggs, N. E., 92
Brosseau-Liard, P. É., 113
Brown, T. A., 55
Brunner, M., 152
Byrnes, J. E., 169, 170

Camilli, G., 102, 113
Card, N. A., 55, 77, 78, 128, 130
Carlson, G., 62, 65, 78
Chen, F. F., 152

Chou, C.-P., 55
Coalson, D. L., 151, 152
Cohen, J., 89, 92
Cohen, P., 89, 92
Conner, M., 33, 36
Cotter, R. A., 94, 113
Crawley, M., 20
Curran, P. J., 80, 81, 84, 88, 92, 113

De Fruyt, F., 33, 36
De Houwer, J., 33, 36
DeFries, J. C., 71, 78
DeMaris, A., 130
Denissen, J. J. A., 33, 36
DiStefano, C., 113

Edwards, J. R., 55
Edwards, M. C., 113
Embretson, S. E., 113
Enders, C. K., 129, 130
Estabrook, R., 169, 170

Fiedler, K., 33, 36
Fiedler, S., 33, 36
Field, A., 20
Field, Z., 20
Finch, J. F., 113
Finney, S. J., 113
Flora, D. B., 113
Forero, C. G., 110, 113
Fox, J., 167, 169, 170
Freeman, M. J., 62, 65, 78
Funder, D. C., 33, 36

Galbraith, J., 111, 113
Garfinkel, P. E., 129, 130
Garner, D. M., 129, 130
Graham, J. M., 54, 55
Graham, J. W., 114, 130
Groothuis-Oudshoorn, K., 120, 130
Guttman, L., 101, 113

- Hallquist, M., 167, 170
Herzberg, P. Y., 113
Ho, M.-H. R., 51, 55
Horn, J. L., 56, 78
Hoyle, R. H., 51, 55, 75, 78
Huh, J., 55
Hunter, A. M., 130

Jalal, S., 123, 128, 130
Jamshidian, M., 123, 128, 130
Janke, M. C., 52, 53, 55
Jansen, C., 123, 128, 130
Johnson, E. C., 61, 78

Kamata, A., 113
Kashy, D., 55
Keith, T. Z., 34
Kelley, K., 131, 133, 144
Kenny, D. A., 55
Kenny, S., 169, 170
Kim, S.-H., 113
Kliegl, R., 33, 36

Lance, C. E., 60, 78
Ledford, E. C., 78
Leuchter, A. F., 130
Li, S.-C., 152
Lieberman, M., 94, 113
Little, R. J. A., 114, 116, 123, 128, 130
Little, T. D., 55, 77, 78
Loehlin, J. C., 55
Lottes, I. L., 130

MacCallum, R. C., 92
MacKinnon, D. P., 34
Maes, H. H. M., 72, 78, 169, 170
Matloff, N., 20
Maxwell, S. E., 131, 133, 144
Maydeu-Olivares, A., 110, 113
McArdle, J. J., 36, 56, 78, 130
McClearn, G. E., 71, 78
McDonald, R. P., 51, 55, 165, 166
McGuffin, P., 71, 78
McLeod, L. D., 152
Meade, A. W., 61, 78

Mehta, P., 169, 170
Miles, J., 20
Miller, P., 134, 144
Millsap, R. E., 78
Moustaki, I., 111, 113
Muthén, B. O., 92, 130, 133–135, 137, 140, 144, 167, 170
Muthén, L. K., 133–135, 137, 140, 144, 167, 170

Nagy, G., 152
Neale, M. C., 72, 78, 169, 170
Nicol, A. A. M., 34, 52, 55
Nosek, B. A., 33, 36

Olmsted, M. P., 129, 130

Page, E. B., 34
Panter, A. T., 51, 55, 75, 78
Parkin, J., 43, 55, 148, 152
Perugini, M., 33, 36
Pexman, P. M., 34, 52, 55
Plomin, R., 71, 78
Pornprasertmanit, S., 134, 144
Preacher, K. J., 92

R Development Core Team, 20
Raiford, S. E., 151, 152
Reese, L. M., 94, 113
Reise, S. P., 113
Revelle, W., 20
Rhemtulla, M., 113
Rizopoulos, D., 113
Roberts, B. W., 33, 36
Rosseel, Y., 36
Rubin, D. B., 114, 130
Rutter, M., 78

Satorra, A., 156, 166
Savalei, V., 113
Schafer, J. L., 130
Schlomer, G. L., 128, 130
Schmiedek, F., 152
Schmitt, M., 33, 36
Schoemann, A., 134, 144

- Sheng, Y., 75, 78, 111, 113
Slegers, D. W., 55, 77, 78
Sousa, K. H., 152
Spiegel, M., 169, 170
Spies, J., 169, 170
Steele, F., 111, 113
Steenkamp, J.-B. E. M., 78

Thissen, D., 152

Umstattd-Meyer, M. R., 52, 53, 55

van Aken, M. A. G., 33, 36
van Buuren, S., 120, 130
Vandenberg, R. J., 60, 78
Voelkle, M. C., 91, 92

Watkins, M. W., 151, 152

Weber, H., 33, 36
Wechsler, D., 38, 55, 62, 65, 78, 148, 151, 152
West, S. G., 89, 92, 113, 152
Wicherts, J. M., 33, 36
Wichman, A. L., 92
Wilde, M., 169, 170
Wiley, J., 167, 170
Wilhelm, O., 152
Wilkinson, L., 118, 130
Wirth, R. J., 113
Wolfe, L. M., 36
Wright, S., 23, 36, 194

Youngstrom, E., 62, 65, 78
Yuan, K.-H., 156, 166
Yung, Y.-F., 152

Subject Index

Bold page numbers indicate page number for glossary entry.

- Auxiliary Variable (AV), 121, 125, **190**
- Behavior Genetics, 71–74, **190**
- Bi-Factor Model, *see* Latent Variable Model
- Bias, *see* Monte Carlo Study
- Categorical Variable, *see* Indicator Variable
- Conditional Parameterization, *see* Item Factor Analysis
- Confirmatory Factor Analysis, *see* Latent Variable Model
- Construct, *see* Latent Variable
- Correlation, 14
 - Matrix, *see* Covariance
 - Model-Implied, 46
 - Pearson, 14
 - Residual, 46
 - Spearman, 14
 - Tetrachoric, 96, 105, **194**
- Covariance, 14
 - Matrix
 - Data Entry, 7, 29
 - Model-Implied, 46
 - Residual, 46
- Coverage, *see* Monte Carlo Study, 133
- Degrees of Freedom, 40, 45, 61, 154, **190**
- Dichotomous Variable, *see* Indicator Variable
- Difficulty, *see* Item Location
- Discrimination, *see* Item Response Theory
- Disturbance, *see* Error
- Domain-Specific Latent Variable, *see* Latent Variable
- Empirical Underidentification, *see* Identification of Latent Variable
- Endogenous Variable, *see* Variable, 57
- Error, 22, 24, 26, 38, 40, 42, 50, 59, 73, 83, 86, 99, 100, 145, **191**
- Exogenous Variable, *see* Variable, 57
- Exploratory Factor Analysis, 37, 121, **191**
- Factor, *see* Latent Variable
- Factor Analysis, *see* Latent Variable Model, **190**
- Factor Loading, 38, 39, 48, **191**
- FIML, *see* Full Information Maximum Likelihood
- First-Order Latent Variable, 145, *see* Latent Variable
- Fit Indexes, 153–166
 - Alternative Fit Indexes, 154, 158–165, **190**
 - χ^2 , 153–158
 - Robust, 156–158
 - Residuals, 165–166
- Full Information Maximum Likelihood (FIML), 119, 124, **191**
- Heritability, 71
- Higher-Order Model, *see* Latent Variable Model
- Homogeneity of Latent Variable Variances, *see* Invariance
- Identification of Latent Variable Model, 39–42, **191**
 - Empirical Underidentification, 42, 150, **191**
 - Hierarchical Model, 147
 - Just-Identified, 39, **192**
 - Latent Variable Scale, 41, 47–48
 - Means and Intercepts, 58
 - Number of Indicator Variables, 40–41
 - Overidentified, 39, **193**
 - Underidentified, 39, **194**
- Imputation, *see* Missing Data

- Indicator Variable, 38, 40–41, **191**
 - Categorical/Dichotomous, 93
- Indirect Effect, 30–32, **191**
- Invariance, 56, 58, 62–70, **192**
 - Configural, 58, **190**
 - Evaluation, 61
 - Latent Variable Covariances, 59
 - Latent Variable Means, 59
 - Latent Variable Variances, 59
 - Partial, 60, 68, **193**
 - Strict, 59, **194**
 - Strong, 59, **194**
 - Structural, 60, **194**
 - Weak, 59, **194**
- Inverse Logit, 140
- Item Characteristic Curve, 100, 106, **192**
- Item Factor Analysis, 96–100, 104, **192**
 - Conditional Parameterization, 100, **190**
 - Conversion Formulae, 104
 - Delta Parameterization, *see* Marginal Parameterization
 - Marginal Parameterization, 99, **192**
 - Theta Parameterization, *see* Conditional Parameterization
 - Threshold, 96, 97
 - Underlying Variable, 96–103, **194**
- Item Response Theory, 100–104, **192**
 - Conversion Formulae, 104
 - Cumulative Logistic Distribution Constant (D), 102, 105
 - Item Discrimination, 102, **192**
 - Item Location, 101, 102, **192**
 - Logistic, 102
 - Normal Ogive, 101
- Latent Curve Model, 79–88, 137–142, **192**
 - Fixing Loadings, 79
 - Interaction, 87
 - Non-Linear Change, 88
 - Piecewise Linear Change, 88
 - Time-Invariant Covariate, 84, 85, **194**
 - Time-Variant Covariate, 84, 85
 - Time-Varying Covariate, **194**
- Latent Variable, 23, **192**
 - Domain-Specific, 147, **190**
 - First-Order, **191**
 - Formative, 38, 54, **191**
 - Identification, *see* Identification of Latent Variable Model
 - Reflective, 38, **193**
 - Scale, 41
 - Effects-Coding, 41, 48, 58
 - Marker Variable, 41, 44, 58
 - Standardized, 41, 47, 58
 - Second-Order, **193**
- Latent Variable Model, 37–39, **192**
 - Bi-Factor, 146–148, **190**
 - Communality, 39, **190**
 - Higher-Order, 145–147, **191**
 - Identification, *see* Identification of Latent Variable Model
 - Means and Intercepts, 56–58
 - Single-Indicator, 40, 42
 - Uniqueness, 39, **194**
- L^AT_EX, 33
- Listwise Deletion, *see* Missing Data
- MACS, *see* Mean and Covariance Structure
- Manifest Variable, *see* Variable
- Marginal Parameterization, *see* Item Factor Analysis
- Mean and Covariance Structure (MACS), 56, **192**
- Measurement Invariance, *see* Invariance, *see* Invariance
- MI, *see* Multiple Imputation
- MIMIC, *see* Multiple Indicators Multiple Independent Causes
- Missing Data
 - Pattern, 114, **192**
 - Auxiliary Variable, *see* Auxiliary Variable
 - Full Information Maximum Likelihood, *see* Full Information Maximum Likelihood
 - Imputation, 118
 - in **R**, 11
 - Listwise Deletion, 12, 118

- Missing at Random (MAR), 116, **192**
 - Missing Completely at Random (MCAR), 114, **192**
 - Multiple Imputation, *see* Multiple Imputation
 - Not Missing at Random (NMAR), 117, **193**
 - Pairwise Deletion, 118
 - Planned Missing Design, 115
 - Model Modification, 47
 - Expected Parameter Change, 47
 - Modification Index, 47
 - Monte Carlo Study, 133–137, **192**
 - Bias, **190**
 - Parameter Estimate, 133
 - Relative Parameter Estimate, 133
 - Coverage, **190**
 - Relative Bias, **193**
 - Standard Error Bias, 133
 - Multiple Imputation (MI), 119–121, 125, **193**
 - Chained Equations, 120
 - Multiple Indicators Multiple Independent Causes (MIMIC), 85, 86, **193**
 - Nested Model, 155, **193**
 - Noncentrality Parameter, 161
 - Ogive, 101
 - Pairwise Deletion, *see* Missing Data
 - Parameter Constraint, 48, 61–62, 70–71
 - Group Equality Constraint, 61–62
 - Labels, 70–71
 - Specific Value, 71
 - Parameter Estimation, 98, 103
 - Full Information, 103
 - Least Squares, 98, 157
 - Limited Information, 98
 - Robust Estimator, 156, 157, **193**
 - Weighted Least Squares, 98
 - Robust (WLSMV), 98, 108, 110, 157
 - Partial Invariance, *see* Invariance
 - Path Analysis, 23–30
 - Path Coefficient, 26–27, **193**
 - Standardized, 26
 - Unstandardized, 26
 - Path Model, 21–23, **193**
 - Symbols, 21
 - Pattern Coefficient, *see* Factor Loading
 - Power, 131, **193**
- R**
- Comment, 3
 - Concatenate, 3
 - Correlation, 14
 - Covariance, 14
 - Data
 - Categorical, 12
 - Factor, 12
 - Import from External Source, 6
 - Import from Other Programs, 6
 - Input, 5
 - Manipulation, 10
 - Packages, 7
 - Simulation, *see* Simulation
 - Sort, 11
 - Subset, 10, 11
 - Summarizing, 12
 - Data Frame, 6
 - Descriptive Statistics, 12
 - Factors, 12
 - Function, 2–4
 - Actual Argument, 4
 - Formal Argument, 4
 - Writing, 4
 - Help, 3
 - Installing, 2
 - Lists, 64, 73
 - Logical Operators, 48
 - Missing Data, 11
 - NA, 6, 11, 48, 129
 - Packages, 4
 - Simulation, 8
 - Table, 13
 - Frequency, 13
 - Relative Frequency, 13
 - R^2 , 24, 26, 29, 39, 86
 - Random Effects, 80, **193**

Relative Bias, *see* Monte Carlo Study

Residual, *see* Error

Sample Size Planning, 131–142

Missing Data, 140–142

Second-Order Latent Variable, 145, *see* Latent Variable

Specific Factor, 145, **193**

Standardized Coefficient, *see* Path Coefficient

Structural Equation Model, 37, 50, **194**

Structure Coefficient, 39, 48, **194**

t Test, 16

Tetrachoric Correlation, *see* Correlation

Threshold, *see* Item Factor Analysis

Tracing Rules, 23–26, 45, 46, 48, **194**

Means and Intercepts, 57–58

Unstandardized Coefficient, *see* Path Coefficient

Variable

Endogenous, 21, **191**

Exogenous, 21, **191**

Latent, *see* Latent Variable

Manifest, 23, 38, **192**

Standardized, 14, 44

Wald Statistic, 16

Wright's Rules, *see* Tracing Rules

Z Score, 14

Z Test, 16

R Function Index

?, *see* help()
[[], 66
\$, 8

abline(), 136
anova(), 155
attach(), 9
auxiliary(), 125

bartlett.test(), 17

c(), 3, 5, 71
cat(), 168
cfa(), 28, 32, 44
coef(), 106
colnames(), 7, 29, 31
sem(), 31
Concatenate, *see* c()
cor(), 14
cor2cov(), 43
cov(), 14
cov2cor(), 46
cut(), 13

data(), 7
cbind(), 19
data.frame(), 19
des(), 134
describe(), 12
detach(), 9

example(), 3
extractModelParameters(), 168
extractModelSummaries(), 168

factor(), 12
file.choose(), 6
findPower(), 137, 140
fitMeasures(), 64
fitMeasures(), 46, 153
fitted(), 46, 135
function(), 4

getPower(), 137
growth(), 80

head(), 9
help(), 3
help.start(), 4

install.packages(), 5
irt.fa(), 106

lavaan(), 28
lavaan Model Syntax
 Constant (~1), 28
 Constrain Parameter (>, <, ==), 48, 71
 Covary (~~), 28
 Define Formative Latent Variable (<~), 28
 Define Non-Model Parameter (:=), 28, 31, 108
 Define Reflective Latent Variable (=~), 28
 Define Reflective Latent Variable (~=), 44
 fixed.x, 125, 138
 Group Equality Constraint (group.equal), 61
 Group Equality Constraint Release (group.partial), 62, 67
 Include Means (meanstructure or sample.mean), 64
 Label Parameter (*), 28, 71, 134
 Free Loading, 48
 missing, 124
 ordered, 107
 orthogonal, 150
 Regress Onto (~), 28
 Standardize All Latent Variables (std.lv), 48, 108, 150
 Threshold (|), 28, 107
 Use Only Starting/Fixed Values (do.fit=FALSE), 135, 138

lavExport(), 169
lavTables(), 166
leveneTest(), 17
library(), 5
list(), 64, 73
LittleMCAR(), 123
lm(), 9
lm(), 19
Logical Operators, 48
lower2full(), 7, 29, 31, 42
ltm(), 105

md.pattern(), 123
mean(), 4
miss(), 140
modificationIndices(), 47
modindices(), 67
mplus2lavaan(), 169

na.omit(), 12, 16

order(), 11, 67

parameterEstimates(), 33, 44, 52
plot(), 106
plotLogitMiss(), 141
plotPower(), 136
pnorm(), 96
pwr.t.test(), 132

read.csv(), 6
read.table(), 6, 9, 129

readModels(), 168
rep(), 136
reshape(), 91
residuals(), 46, 165
rev(), 11
rnorm(), 8
rownames(), 7, 29, 31
runMI(), 125
runModels(), 168

scale(), 15
sem(), 28, 30, 32
length(), 14
seq(), 8, 14, 136
sim(), 135
source(), 18
summary(), 12, 29, 44, 50
summaryParam(), 136
summaryTime(), 136

t.test(), 16
table(), 13
TestMCARNormality(), 123
tetrachoric(), 105

var(), 4
var.test(), 17

with(), 9

xtable(), 33, 52

z.test(), 16

R Package Index

Amelia, 125

BaylorEdPsych, 5, 123

car, 17

compute.es, 134

lavaan, 7, 27, 32, 80, 107, 124

ltm, 105, 111

mice, 120, 123, 125, 141

MissMech, 123

MplusAutomation, 167

OpenMx, 169

psych, 12, 19, 104, 105

pwr, 132

sem, 167

semTools, 125

simsem, 134, 135

TeachingDemos, 16

xtable, 33

R Dataset Index

Abortion, 111

galton, 19

lsat6, 94, 104

MLBPitching2011, 8