In: Intelligence Quotient
Editor: Joseph C. Kush
ISBN: 978-1-62618-728-3

Beaujean, A. A., & Parkin, J. (2013). Using R for the analysis of cognitive abilities and behavior genetic data. In J. Kush (Ed.), Intelligence quotient: Testing, role of genetics and the environment and social outcomes (pp. 129-168). New York, NY: Nova Science.

Chapter 8

USING R FOR THE ANALYSIS OF COGNITIVE ABILITY AND BEHAVIOR GENETIC DATA

***A. Alexander Beaujean*[*,1] *and Jason Parkin*[†,2]**
[1]Baylor Psychometric Laboratory, Baylor University
Waco, Texas, US
[2]Lake Washington School District
Redmond, Washington, US

Keywords: R, Cognitive Ability, Behavior Genetics, lavaan, Latent Variable Models

1. The R Program

1.1. Description

R (R Development Core Team, 2011) is a free, very flexible programming language, available for most operating systems (e.g., Windows, Mac, Linux). Thus, many professional statisticians, university students and faculty, as well as businesses are turning to it for their data analysis needs. The *New York Times* even published an article about R's growing popularity (Vance, 2009).

R is currently maintained by the R core-development team (an international team of volunteer developers), and the R Project's web page is http://www.r-project.org (also known as Comprehensive R Archive Network [CRAN]). This is the main site for R information, directions for obtaining the software, accompanying packages, and some user documentation.

1.2. Installing R

You can download R by going to http://www.r-project.org/. Then, on the left-hand side, click on CRAN and then select a mirror from the USA (unless you want to use R it a different language). This will bring you to a page with links to download R for Linux, Mac,

*E-mail address: Alex_Beaujean@baylor.edu
†E-mail address: jrpzpf@mail.missouri.edu

and Windows. Be sure to download the latest version.

There are 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 the R site (http://www.sciviews.org/ rgui/). There are also many text editors that are either designed to interact with R, or can be modified to do so. Typing in "R text editor" (or something similar) into your favorite search engine will bring up many different text editor options, as well as people's opinons of the editors.

1.3. Starting R

If you use the interactive mode for R (as opposed to batch mode), when you initially start it, something similar to the following syntax will automatically appear.

```
R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-apple-darwin9.8.0/x86_64 (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.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

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.

>
```

The > is called the prompt. It is not typed (if you type it, R will assume you mean "greater than"). Instead, it is used to indicate where you are to type. For interactive mode uses of R , you will type in all your commands at the > prompt. If a command is too long to fit on a single line, a + is used for the continuation prompt. Another symbol that you will use frequently in R is the left arrow, <−, which is R's standard assignment operator (another option is using =, but it is better to reserve using = when your are 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.

The easiest way to begin to use R is as a calculator. For example:

```
> 2+2
[1] 4
```

1.4. 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 use is done through functions that you can apply to data or other objects. R functions are a set of instructions that take your input, compute the desired value(s), and return the result. R comes pre-loaded with a set of commonly used functions, but you can add additional ones by either loading packages with the desired functions, or by writing your own function. To use functions: (*a*) give the function's name followed by parentheses; and (*b*) in the parentheses, give the necessary values for the function's argument(s). Here is an example of a function to calculate the arithmetic mean, named `ArithMean()`.

```
1 > #Gives the arithmetic mean
2 > ArithMean<-function(x) {
3 +  Sx<-sum(x) #the sum() function is native to R
4 +  Mean<- Sx/length(x) #the length() function is native to R
5 +  return(Mean)
6 +  }
7 > ArithMean(c(5,10,15)) #the c() function is is defined later
8 [1] 10
```

First, we told R 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, }, is placed at the end of the function (line 6). After defining the function, line 7 gives the syntax to evaluate one call to it. Since the sum of the numbers in the vector `c(5,10,15)` is 30, the length of the vector (i.e., the number of elements) is 3, and the call to the function returned the value 10.

In the `ArithMean()` function, `x` is the *formal argument*, whereas in the call to function `c(5,10,15)` is the *actual argument*. The formal argument is a placeholder, but `c(5,10,15)` is the value used in the computation. Sometimes R functions have default arguments, which are values that a function's arguments will automatically initialize unless you specify a different value.

1.4.1. Useful R Functions

There are some very helpful R functions that you will use repeatedly.

- *Comment*. This is not really a function, but in R anything after the # sign is assumed to be a comment and R will ignore it. Line 1 of the `ArithMean()` function was a comment, which R ignored when reading that syntax. Comments are extremely helpful "functions", as annotating your R code can save you a lot of time and effort.

- *Concatenate*. The *concatenate* function, `c()`, will concatenate (i.e, join) the arguments you include in the function. If you use the `c()` function in conjunction with <−, you can assign whatever you are concatenating into a new object. For example, to make a data set of 5 observations with the values 4, 5 3, 6, 9, and name it `newData` you would use the following syntax.

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

- *Help*. The `help()` function will obtain information about a function (or certain special words or characters). You can also use a question mark, `?`, as a shortcut for the help function. For example, the following two lines of syntax will return the same results.

```
1 > help(mean)
2
3 > ?mean
```

The `help()` function returns a page that (at least) describes the function (or word/character), its arguments, and gives some examples of how to use the function. Some pages have much more detail than others. If you just want to examine/run the example syntax for a function, you can use the `example()` function.

```
1 > example(mean)
2
3 mean> x <- c(0:10, 50)
4
5 mean> xm <- mean(x)
6
7 mean> c(xm, mean(x, trim = 0.10))
8 [1] 8.75 5.50
```

If you want help on an entire `R` package, then use the package argument in the `help()` function.

```
1 help(package=lavaan) #you need to install the lavaan package first
```

If you do not know exactly you need to do within `R`, then you can search `R`'s documentation via the `help.search()` function. The argument you use in the function needs to be enclosed in quotation marks. For example, if you are interested in testing to see if a variable follows a normal distribution, you could use the following syntax.

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

This produces a response that contains functions from packages that might be of interest. An example of such output is shown in Figure 1, which indicates that in the `stats` package (a package `R` installs by default) there is a function called `shapiro.test()` that will perform the Shapiro-Wilk test for normality.

Topic	Package	Description
`shapiro.test`	`stats`	Shapiro-Wilk Normality Test

Figure 1. Example Results from `help.search()` Function.

Another useful way to get help is to use the Rseek web site (http://www.rseek.org/), which is a site that uses Google to find `R` functions, lists, code, etc. If you are totally lost on where to start asking for help, then typing `help.start()` into `R` will return much of the important documentation needed to navigate `R`, as well as providing yet another search engine for `R` helping materials.

1.5. R Packages

Functions are a very important part of using R. When you first downloaded R, it came with some base packages that supply functions for many statistical analysis [e.g., `mean()`, `var()`]. These functions, however, may not do the specific analysis you specifically need to do. Thus, you can look to see if a contributed R package can do what you need to do. These R packages usually consist of data and functions that were written in the R language (although sometimes they are written in FORTRAN or C and then linked back into R). This user-contribution ability is extremely powerful, as there are many experts in various fields using R, some of whom have contributed packages that can make your data analysis projects much easier.

The list of R packages, with a short description of what they do, can be found in CRAN, but it is very long and hard to navigate unless you know the specific name of the package for which you are looking. An alternative is to examine the *CRAN task views* http://cran.r-project.org/web/views/, which is designed to help users find packages associated with specific types of work. For example, the *Psychometrics* http://cran.r-project.org/web/views/Psychometrics.html view has many packages dealing with item and test analysis.

To install a package from the command line, use the `install.packages()` function, naming the package to install in quotation marks. For example, to install the *BaylorEdPsych* (Beaujean, 2012) package, use the following syntax.

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

The `dep = TRUE` argument tells R that in addition to the package of interest, also download any other package upon which the package of interest is dependent. (This saves you from having to download each required package separately.)

You only need to install a package to the hard disk one time, but you will need to load it into memory every time you start a new R session and need to use one of the package's functions using the `library()` function, e.g.,

```
1 library(BaylorEdPsych)
```

(Notice there are no quotation marks around the package name.)

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

1.6. Inputting Data

1.6.1. Concatenate

The easiest way to enter data into R is to type it in directly using the *concatenate*, `c()` function and assign a name to it. To verify that that your data is in your object (i.e., a vector), just type the object's name.

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

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

Then you can immediately apply functions to the new object, `newData`

```
> mean(newData)
[1] 5.4
> sum(newData)
[1] 27
```

1.6.2. Reading In Data from an External Source

Unless the data set has one variable with a few observations (e.g., data from a textbook example), you will usually want to store your data in an external file and have `R` load the data into its working memory.

`read.table()` One way to read externally-stored data is the `read.table` function. Before doing this, however, it will be beneficial to do three things to your data. First, change all missing values to `NA`, which is the default missing value indicator in `R`. Second, make sure all the variable names are only one word (i.e, there are no spaces), but you can use the "." in lieu of a space (e.g., first.name). Third, either save the file as a tab-delimited text (.txt) file or a comma-delimited .txt or .csv file. Most spreadsheet and database programs can save data either way.

In order to read the file, you will need to point `R` to the directory where it is located. In `R` you have to use a forward slash (how Mac and other UNIX-type systems store files) or double backslash when giving a file location, e.g., `C:\\Regression\\Data.csv`. Let's say you have a `.csv` file called `data.csv` that has your data (with a label on the first row of each variable) located in a folder called *name*, which is in a folder called *file*. You can type either of the following syntaxes into `R` to read the file.

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

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

Notice that the `read.csv()` function is the just like the `read.table()` function, only it assumes you have comma-delimited values, so you do not have to use the `sep=","` argument with it.

If you store your data under many sublayers of folders, or you forget the exact location, you can search for your data file using the `file.choose()` function.

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

1.6.3. Inputting a Correlation/Covariance Matrix

In some situations you will not have access to raw data, but will have access to a covariance matrix. You could type the entire matrix into R using the `matrix()` function, but since such matrices are symmetric, you can make use of the `diag()`, `upper.tri()` and `lower.tri()` functions. By default, R will assume you are entering the data for the matrix by columns. The following code will create a correlation matrix titled `CorM` that consists of the correlations among four variables. In addition, give names to the variables using the `rownames()` and `colnames()` functions.

```
> CorM<-diag(4) #4 x 4 Diagonal matrix
> CorMNames<-c("Var1", "Var2", "Var3", "Var4") #Names of the variables
> rownames(CorM)<-colnames(CorM)<-CorMNames #Gives row and column names
> CorM[lower.tri(CorM, diag=FALSE)]<-c(.85, .84, .68, .61, .59, .41) #
    lower triangle of matrix, order is by columns
> CorM[upper.tri(CorM, diag=FALSE)] <- CorM[lower.tri(CorM)] #make matrix
    full
> CorM

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

1.7. The `lavaan` Package

`lavaan` (Rosseel, 2012) is an R package designed to do structural equation modeling. Information and documentation about it can be found on the web page: http://www.lavaan.org which redirects to http://lavaan.ugent.be.

You can install `lavaan` from CRAN via the following R syntax

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

To compute a model in `lavaan`, first you have to specify the model as text, using a few pre-defined commands that are shown in Table 1.

Table 1. `lavaan` commands

Syntax	Command	Example
∼	Regress onto	Regress B onto A: `B ∼ A`
∼∼	(Co)varaince	Variance of A: `A ∼∼ A` Covariance of A and B: `A ∼∼ B`
∼1	Constant/mean	Regress B onto A, and and include the intercept in the model: `B ∼ 1 + A`
∼=	Define latent variable	Define Factor 1 by A-D: `F1 ∼= A+B+C+D`
:=	Define non-model parameters	Define parameter u2 to be 2 times the square of u: `u2 := 2*(u^2)`
<∼	Define formative variables	Define Variable A by $X_1 - X_4$: `A <∼ X1 + X2 + X3 + X4`

Within the model, you can label parameters by premultiplying the label onto one of the

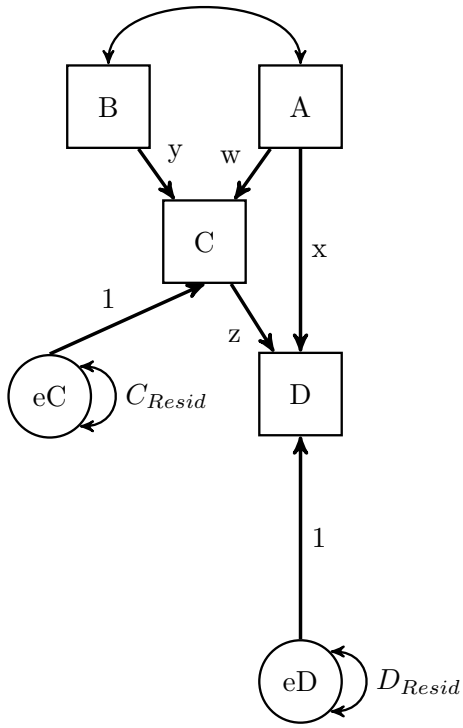

Figure 2. Path Model Example.

variables on the left hand side of an equation. For example, the syntax for specifying the model in Figure 2 in `lavaan` would be as follows.

```
1 example.model<-'
2 C ~ y*B + w*A
3 D ~ z*C + x*A
4 C~~C_Resid*C #Optional, included just to label residual variance
5 D~~D_Resid*D #Optional, included just to label residual variance
6 '
```

Line 1, consists of the name of the model (`example.model`), the assignment operator (`<-`) and single apostrophe, which indicates that the subsequent syntax will be passed as text. Line 2 defines the structural model for C, with labels for each parameter, designed to map onto the labels in Figure 2; likewise, line 3 defines the structural model for D with labels. Lines 4 and 5 are optional. By default, `lavaan` creates an error term for each endogenous variable and estimates the variance while constraining the path to one. So, the only thing lines 4 and 5 contribute is the labeling of these error variances. Likewise, by default `lavaan` creates a covariance term for each exogenous variable (represented by the two-headed arrow from A to B in Figure 2). Line 6 is another single apostrophe, indicating the end of the syntax-as-text option.

You can fit the specified model by using either the `cfa()` (for factor analytic models), or `sem()` (for structural equation models) functions. The `cfa()` and `sem()` functions are really just the calls to a more general `lavaan()` function with some of the default values specified differently. To obtain the default values for the `cfa()` or `sem()` functions, as

well as their various arguments, type `?cfa` or `?sem`, respectively, in R.

`lavaan` accepts either a covariance matrix (with sample size) or raw data as input, using the `sample.cov=` (with `sample.nobs=`) or `data=` arguments, respectively. If using the covariance matrix as input, you can optionally also input a mean vector with the `sample.mean=` argument. By default, `lavaan` uses normal-theory maximum likelihood as the parameter estimation technique for continuous-variable indicators, but you can change this to generalized least squares, weighted least squares (ADF), unweighted least squares, or diagonally weighted least squares (for categorical indictors) by specifying them in the `estimator=` argument.

For example, say data was collected from 500 individuals on variables A-D that were specified in `example.model`, and the data has been inputted into R, naming it `example.data`. To estimate the parameters in `example.model` using `example.data`, use the following `lavaan` syntax.

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

To input the correlation matrix of the data (named, say, `example.cov`) instead of the raw data, use the following syntax

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

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

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

By default, the `summary()` function for `lavaan` objects will produce (*a*) a note indicating if the minimization algorithm converged; (*b*) the sample size; (*c*) estimator; (*d*) χ^2; (*e*) χ^2 degrees of freedom; (*f*) p-value of the χ^2; (*g*) the unstandardized parameter estimates; (*h*) their parameter estimates' standard errors; (*i*) the ratio of the parameter estimates and their standard errors (i.e., *Z*); and (*j*) the ratio's *p*-value. The `summary()` function has these default specifications for its arguments: `standardized=FALSE, fit.measures=FALSE, rsquare=FALSE, modindices=FALSE`. Setting `standardized=TRUE` will include standardized estimates in the results, setting `fit.measures=TRUE` will include various fit indices in the results, setting `rsquare=TRUE` will include the R^2 for each exogenous variable, and setting `modindices=TRUE` will include fit indices.

For the many other functions the package includes, see the documentation for the `lavaan` package here http://cran.r-project.org/web/packages/lavaan/lavaan.pdf

2. Latent Variable Models of Cognitive Ability

2.1. Single Factor Model

Much of the cognitive ability literature indicates that there is a single general (common) intelligence factor (*g*) that influences most measures of cognitive ability (Jensen, 1998; Spearman, 1927). Such a model, using subtests from the Wechsler Intelligence Scale for Children-Fourth Edition (WISC-IV (Wechsler, 2003a)) is shown in Figure 3. This model

specifies that the only reason the subtests are correlated is because of *g*, and that once you account for *g*, the subtests are no longer correlated. *g* does not explain all of the variance in the subtests, however. Those residual/error functions accounting for non-*g* related variance are represented as single-headed arrows not attached to the common factor.

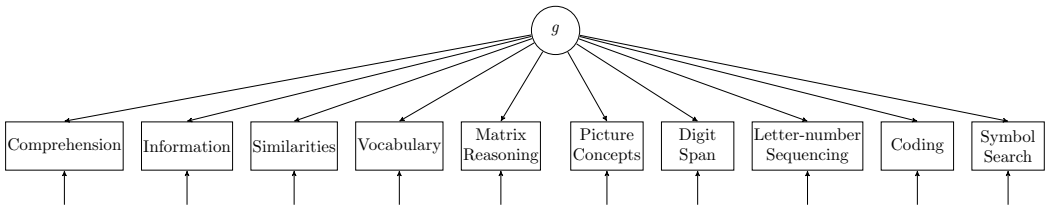

Figure 3. Single factor model of cognitive ability from the WISC=IV.

Some sample correlations and standard deviations (SD) for the following WISC-IV subtests are shown in Table 2 taken from (Parkin & Beaujean, 2012), p. 118: Comprehension, Information, Matrix Reasoning, Picture Concepts, Similarities, Vocabulary, Digit Span, Letter Number Sequencing, Coding and Symbol Search.

Table 2. Correlations and standard deviations for WISC-IV subtests

	Subtest	1	2	3	4	5	6	7	8	9	10
1	Comprehension	1.00									
2	Information	0.62	1.00								
3	Matrix Reasoning	0.34	0.51	1.00							
4	Picture Concepts	0.33	0.37	0.39	1.00						
5	Similarities	0.63	0.73	0.48	0.38	1.00					
6	Vocabulary	0.71	0.74	0.46	0.37	0.74	1.00				
7	Digit Span	0.41	0.42	0.37	0.31	0.43	0.45	1.00			
8	Letter Number	0.44	0.50	0.43	0.38	0.50	0.52	0.51	1.00		
9	Coding	0.27	0.30	0.28	0.27	0.23	0.29	0.30	0.32	1.00	
10	Symbol Search	0.34	0.43	0.41	0.30	0.38	0.39	0.37	0.45	0.49	1.00
	SD	2.88	3.01	2.89	2.98	3.03	3.02	2.98	2.99	2.96	3.12

Note. Correlations and Standard Deviations taken from (Parkin & Beaujean, 2012), p. 118.

To analyze this model in `lavaan`, we need to (*a*) input the correlation matrix and SD vector; (*b*) convert the correlations to covariances (Cudeck, 1989); (*c*) specify the factor model; (*d*) estimate the model parameters; and (*e*) assess the model to see how well it fits the data.

First, enter the data (the correlation matrix and standard deviation vector). One option when entering covariance/correlation matrices is to type the entire matrix into `R` using the `matrix()` function, but a simpler option is to make use of the fact that such matrices are symmetric and use the diagonal, upper and lower triangle, and transpose functions, `diag()`, `upper.tri()`, `lower.tri()`, and `t()`, respectively. By default, `R` assumes the data being entered for a matrix is by columns.

```
1 WiscIV.cor<-matrix(NA, nrow=10, ncol=10) #Create empty 10 x 10 matrix
2 diag(WiscIV.cor)<-1 #Place 1s on the diagonal of each matrix
3 #Create the lower triangle of the matrix
```

```
WiscIV.cor[lower.tri(WiscIV.cor)]<-c(0.62, 0.34, 0.33, 0.63, 0.71, 0.41,
    0.44, 0.27, 0.34, 0.51, 0.37, 0.73, 0.74, 0.42, 0.50, 0.30, 0.43,
    0.39, 0.48, 0.46, 0.37, 0.43, 0.28, 0.41, 0.38, 0.37, 0.31, 0.38,
    0.27, 0.30, 0.74, 0.43, 0.50, 0.23, 0.38, 0.45, 0.52, 0.29, 0.39,
    0.51, 0.30, 0.37, 0.32, 0.45, 0.49)
#Transpose the lower triangle to input the upper triangle of the matrix
WiscIV.cor[upper.tri(WiscIV.cor)]<-t(WiscIV.cor)[upper.tri(WiscIV.cor)]
#input the SDs
WiscIV.sd<-c(2.88,3.01,2.89,2.98,3.03,3.02,2.98,2.99,2.96,3.12)
```

Next, convert the correlations to covariances.

```
#make covaraice out of correlations
library(lavaan)
WiscIV.cov<-cor2cov(WiscIV.cor, WiscIV.sd)
#name the rows and columns of the covariance matrix
rownames(WiscIV.cov)<-colnames(WiscIV.cov)<-c("Comprehension", "
    Information", "Matrix.Reasoning", "Picture.Concepts", "Similarities",
     "Vocabulary",  "Digit.Span", "Letter.Number",  "Coding", "Symbol.
    Search")
#To see the actual WiscIV.cov object
WiscIV.cov
```

The `cor2cov()` function is part of the `lavaan` package, so you need to load that, via the `library()` function, before using it. The covariance matrices used for analysis in `lavaan` need to have both row and column names or `lavaan` will not be able to match the variable in the data with the variable in the model.

Next, specify the single factor model in `lavaan`.

```
WISC.oneFactor.model<-'
g=~ Comprehension + Information + Matrix.Reasoning + Picture.Concepts +
    Similarities + Vocabulary +  Digit.Span + Letter.Number + Coding +
    Symbol.Search
'
```

The model itself is enclosed in apostrophes (i.e., it is a text object). *g* is defined (=~) using all 10 WISC-IV subtests, which is specified by using the + sign before each additional variable after the =~.

To obtain the parameter estimates, first, estimate the parameters using the `cfa()` function.

```
WISC.oneFactor.fit<-cfa(model=WISC.oneFactor.model, sample.cov=WiscIV.cov
    , sample.nobs=550, std.lv=TRUE)
```

The first two arguments to the `cfa()` function are the model and the data (i.e., covariance matrix), which we have already defined. The third argument, `sample.nobs()`, tells `lavaan` the sample size from which the covariance matrix was calculated and needs to be used anytime a covariance matrix is used for input. The last argument, `std.lv=TRUE`, tells `lavaan` to estimate all the factor loadings (pattern coefficients) and constrain the latent variable's variance to unity. If the `std.lv=TRUE` option is not used, then the factor variance would be estimated and the first indicator variable's loading would be set to unity.

To print the parameter estimates, use the `summary()` function. Specifying the `standardized=TRUE` argument in the `summary()` function produces the standardized factor loadings (standardized regression weights) as well as the unstandardized loadings.

```
lavaan (0.5-9) converged normally after  28 iterations

  Number of observations                           550

  Estimator                                         ML
  Minimum Function Chi-square                  276.092
  Degrees of freedom                                35
  P-value                                        0.000

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  g =~
    Comprehension     2.159    0.107   20.118    0.000    2.159    0.750
    Information       2.534    0.106   23.911    0.000    2.534    0.843
    Matrix.Resnng     1.680    0.116   14.439    0.000    1.680    0.582
    Pictur.Cncpts     1.408    0.124   11.328    0.000    1.408    0.473
    Similarities      2.525    0.107   23.537    0.000    2.525    0.834
    Vocabulary        2.615    0.105   25.003    0.000    2.615    0.867
    Digit.Span        1.649    0.121   13.613    0.000    1.649    0.554
    Letter.Number     1.896    0.118   16.093    0.000    1.896    0.635
    Coding            1.127    0.126    8.917    0.000    1.127    0.381
    Symbol.Search     1.616    0.128   12.590    0.000    1.616    0.518

Variances:
    Comprehension     3.619    0.246                      3.619    0.437
    Information       2.624    0.202                      2.624    0.290
    Matrix.Resnng     5.515    0.348                      5.515    0.662
    Pictur.Cncpts     6.882    0.426                      6.882    0.776
    Similarities      2.789    0.211                      2.789    0.304
    Vocabulary        2.268    0.187                      2.268    0.249
    Digit.Span        6.144    0.385                      6.144    0.693
    Letter.Number     5.327    0.341                      5.327    0.597
    Coding            7.477    0.458                      7.477    0.855
    Symbol.Search     7.105    0.443                      7.105    0.731
    g                 1.000                               1.000    1.000
```

The top part of the output can be used as a double-check to make sure the model was specified correctly. For this model, $n = 550$ (correct!) and the $df = 10 \times 11/2 - (10+10) = 55 - 20 = 35$ (correct!). The factor loadings in the `Estimate` column give the unstandardized loadings, while the loadings in the `Std.all` column give the standardized loadings. Both indicate that each indicator is a relatively strong measure of *g*, with Vocabulary being the most saturated (i.e., having the strongest loading) and Coding being the least.

To estimate the communality of each indicator (i.e., the amount of variance *g* explains), either calculate the values "manually" by squaring each loading, or let `R` do the work by extracting the loadings by using the `inspect()` function with the `what="rsquare"` argument.

```
> inspect(object=WISC.oneFactor.fit, what="rsquare")
```

```
Comprehension  Information Matrix.Reasoning Picture.Concepts
0.5628758         0.7098617         0.3384281         0.2236483

Similarities    Vocabulary        Digit.Span
0.6956403         0.7508676         0.3069146

Letter.Number        Coding    Symbol.Search
0.4030142         0.1451074         0.2688097
```

From the output, you can see that *g* explains from 15-75% of the variance in the subtests.

Last, to obtain some measures of model fit, use the `fitMeasures()` function.

```
> fitMeasures(WISC.oneFactor.fit)
chisq                df            pvalue    baseline.chisq
276.092            35.000             0.000          2552.014

baseline.df   baseline.pvalue
45.000             0.000

cfi               tli              logl unrestricted.logl
0.904             0.876        -12676.518        -12538.472
npar            aic
20.000         25393.037

bic            ntotal          bic2             rmsea
25479.235      550.000         25415.747        0.112

rmsea.ci.lower   rmsea.ci.upper
0.100              0.124

rmsea.pvalue      srmr       srmr_nomean
0.000             0.067             0.067
```

While `lavaan` does not print out every fit statistic values, it does print enough information to estimate any other fit statistics. For example, McDonald's (McDonald, 1989) noncentrality index (Mc), is calculated as:

$$Mc = \exp(-0.5 d_T) \tag{1}$$

where
exp() is the exponentiation function, and
d_T is the scaled NCP (noncentrality index) for the target model, i.e, $d_T = \frac{\chi^2 - df}{n-1}$.
This is calculated in R using the following syntax:

```
> NCP<-as.numeric(fitMeasures(WISC.oneFactor.fit,  fit.measures="chisq"))
    - as.numeric(fitMeasures(WISC.oneFactor.fit,  fit.measures="df")) #
   Non-centrality Parameter
> d<- NCP/550 #scaled NCP
> Mc<-exp(-0.5*d)
> Mc
[1] 0.8031815
```

where the `as.numeric()` function strips away everything in the object except the number.

2.2. Multifactor Model

Many current scholars in cognitive ability research argue that there is more than a single factor (general) factor influencing performance on cognitive ability tests (e.g. (Newton & McGrew, 2010)). Instead, they posit that there are multiple factors, although these factors are related to each other. Such a multifactor model of the WISC-IV data is shown in Figure 4, which posits that there are four factors: (*a*) Comprehension-Knowledge (*Gc*), (*b*) Fluid Intelligence (*Gf*), (*c*) Short-term Retrieval (*Gsm*), and (*d*) Processing Speed (*Gs*) (cf. (Wechsler, 2003b)).

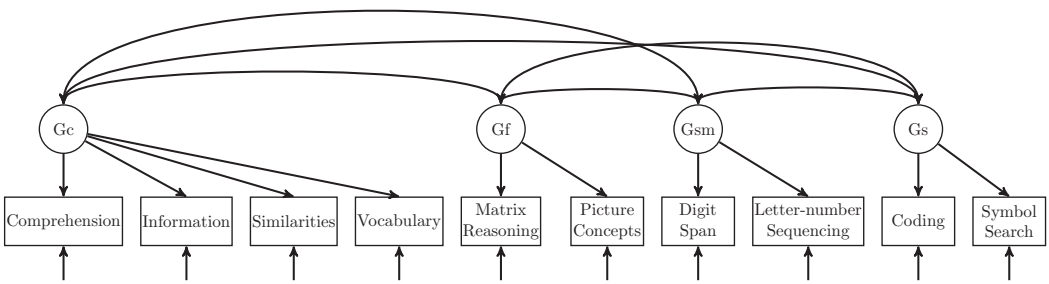

Figure 4. Four Factor Model of Cognitive Ability from the WISC-IV.

Since this model uses the same data as the single factor model, the data entry component is the same. The model specification in `lavaan` is as follow.

```
> WISC.fourFactor.model<-'
Gc =~ Comprehension + Information +  Similarities + Vocabulary
Gf =~ Matrix.Reasoning + Picture.Concepts
Gsm =~  Digit.Span + Letter.Number
Gs =~ Coding + Symbol.Search
'
```

To estimate the parameters and obtain the fit statistics, use the following syntax.

```
> WISC.fourFactor.fit<-cfa(model=WISC.fourFactor.model, sample.cov=WiscIV
    .cov, sample.nobs=550, std.lv=TRUE)
> summary(WISC.fourFactor.fit, fit.measure=TRUE, standardized=TRUE)

lavaan (0.5-9) converged normally after  40 iterations

  Number of observations                           550

  Estimator                                         ML
  Minimum Function Chi-square                   51.634
  Degrees of freedom                                29
  P-value                                        0.006

Chi-square test baseline model:

  Minimum Function Chi-square                 2552.014
  Degrees of freedom                                45
  P-value                                        0.000

```

```
Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.991
  Tucker-Lewis Index (TLI)                       0.986

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -12564.289
  Loglikelihood unrestricted model (H1)     -12538.472

  Number of free parameters                         26
  Akaike (AIC)                               25180.578
  Bayesian (BIC)                             25292.636
  Sample-size adjusted Bayesian (BIC)        25210.101

Root Mean Square Error of Approximation:

  RMSEA                                          0.038
  90 Percent Confidence Interval          0.020  0.054
  P-value RMSEA <= 0.05                          0.885

Standardized Root Mean Square Residual:

  SRMR                                           0.020

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  Gc =~
    Comprehension     2.192    0.107   20.526    0.000    2.192    0.762
    Information       2.544    0.106   24.002    0.000    2.544    0.846
    Similarities      2.558    0.107   23.963    0.000    2.558    0.845
    Vocabulary        2.670    0.104   25.780    0.000    2.670    0.885
  Gf =~
    Matrix.Resnng     1.999    0.136   14.698    0.000    1.999    0.692
    Pictur.Cncpts     1.677    0.136   12.340    0.000    1.677    0.563
  Gsm =~
    Digit.Span        1.968    0.126   15.589    0.000    1.968    0.661
    Letter.Number     2.305    0.126   18.234    0.000    2.305    0.772
  Gs =~
    Coding            1.778    0.136   13.060    0.000    1.778    0.601
    Symbol.Search     2.541    0.152   16.740    0.000    2.541    0.815

Covariances:
  Gc ~~
    Gf                0.779    0.041   18.896    0.000    0.779    0.779
    Gsm               0.765    0.033   23.395    0.000    0.765    0.765
    Gs                0.560    0.042   13.171    0.000    0.560    0.560
  Gf ~~
    Gsm               0.824    0.050   16.405    0.000    0.824    0.824
    Gs                0.705    0.054   12.940    0.000    0.705    0.705
```

```
  Gsm ~~
    Gs                0.707    0.047   15.199    0.000    0.707    0.707

Variances:
    Comprehension     3.474    0.240                      3.474    0.420
    Information       2.573    0.204                      2.573    0.284
    Similarities      2.621    0.207                      2.621    0.286
    Vocabulary        1.977    0.181                      1.977    0.217
    Matrix.Resnng     4.340    0.424                      4.340    0.521
    Pictur.Cncpts     6.052    0.434                      6.052    0.683
    Digit.Span        4.991    0.377                      4.991    0.563
    Letter.Number     3.611    0.381                      3.611    0.405
    Coding            5.585    0.428                      5.585    0.639
    Symbol.Search     3.261    0.574                      3.261    0.336
    Gc                1.000                               1.000    1.000
    Gf                1.000                               1.000    1.000
    Gsm               1.000                               1.000    1.000
    Gs                1.000                               1.000    1.000
```

Notice the `fit.measure=TRUE` argument in the `summary()` function. Specifying this gives the fit indices along with the model parameter estimates, which is an alternative to using the `fitMeasures()` function.

As the single factor model is nested in the four-factor model (i.e., the one-factor model is a more restrictive version of the four factor model (Brunner, Nagy, & Wilhelm, 2012)), the models can be compared directly using the `anova()` function.

```
Chi Square Difference Test

                     Df   AIC   BIC   Chisq Chisq diff Df diff Pr(>Chisq)
WISC.fourFactor.fit 29 25181 25293  51.634
WISC.oneFactor.fit  35 25393 25479 276.092     224.46       6  < 2.2e-16
    ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1
```

It appears that from both the change in χ^2 values and the other fit measures that the four-factor model fits the data somewhat better than the single factor model.

2.3. Higher-Order Model

The models considered thus far have only estimated factors that directly influence the subtests. An alternative to this kind of model is a higher-order model, which specifies that there are factors that directly influence the subtests (i.e, first-order factors) as well as factors that directly influence the factors (i.e., second-order factors) (Rindskopf & Rose, 1988). A typical higher-order model for cognitive abilities data is one that posits that *g* is the sole reason why the the first-order factors are correlated with each other (Carroll, 1993). Such a model is shown in Figure 5.

The specification of this model is similar to that from the single- and multi-factor models, except this model also requires specifying a factor made up of the four other factors.

```
WISC.higherOrder.model<-'
gc =~ Comprehension + Information +  Similarities + Vocabulary
```

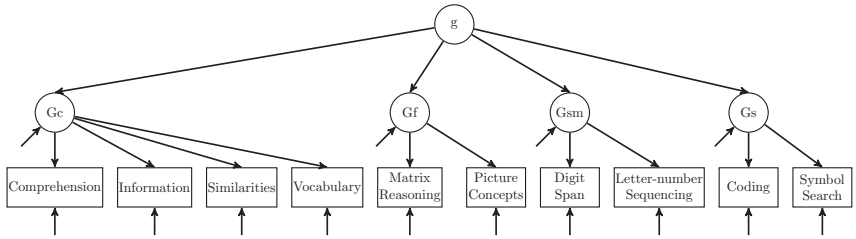

Figure 5. Higher Order Factor Model of Cognitive Ability from the WISC-IV.

```
3 gf =~ Matrix.Reasoning + Picture.Concepts
4 gsm =~  Digit.Span + Letter.Number
5 gs =~ Coding + Symbol.Search
6
7 g=~ NA*gf + gc  + gsm + gs
8 g~~1*g
9 '
```

Notice the `NA*` in front of the `gf` term in line 7. This tells `lavaan` to estimate this loading instead of constraining it to unity (i.e., 1), which is the default. The trade-off for doing this is that *g*'s variance must be constrained to unity, which is done with the `g~~1*g` syntax in line 8. This is done because it is better to estimate the (residual) variances of the first-order factors instead of constrain them to be unity.

Next, estimate the parameters and obtain the results and fit statistics.

```
 1 > WISC.higherOrder.fit<-cfa(model=WISC.higherOrder.model, sample.cov=
       WiscIV.cov, sample.nobs=550)
 2 > summary(WISC.higherOrder.fit, fit.measure=TRUE, standardized=TRUE)
 3 lavaan (0.5-9) converged normally after  56 iterations
 4
 5   Number of observations                           550
 6
 7   Estimator                                         ML
 8   Minimum Function Chi-square                   57.592
 9   Degrees of freedom                                31
10   P-value                                        0.003
11
12 Chi-square test baseline model:
13
14   Minimum Function Chi-square                 2552.014
15   Degrees of freedom                                45
16   P-value                                        0.000
17
18 Full model versus baseline model:
19
20   Comparative Fit Index (CFI)                    0.989
21   Tucker-Lewis Index (TLI)                       0.985
22
23 Loglikelihood and Information Criteria:
24
25   Loglikelihood user model (H0)             -12567.268
26   Loglikelihood unrestricted model (H1)     -12538.472
27
28   Number of free parameters                         24
```

```
  Akaike (AIC)                                   25182.537
  Bayesian (BIC)                                 25285.975
  Sample-size adjusted Bayesian (BIC)            25209.788

Root Mean Square Error of Approximation:

  RMSEA                                          0.039
  90 Percent Confidence Interval          0.023  0.055
  P-value RMSEA <= 0.05                          0.856

Standardized Root Mean Square Residual:

  SRMR                                           0.023

Parameter estimates:

  Information                                    Expected
  Standard Errors                                Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  gc =~
    Comprehension     1.000                               2.194    0.762
    Information       1.160    0.056   20.813    0.000    2.545    0.846
    Similarities      1.165    0.056   20.753    0.000    2.555    0.844
    Vocabulary        1.217    0.056   21.857    0.000    2.670    0.885
  gf =~
    Matrix.Resnng     1.000                               1.990    0.689
    Pictur.Cncpts     0.847    0.079   10.754    0.000    1.685    0.566
  gsm =~
    Digit.Span        1.000                               1.967    0.661
    Letter.Number     1.172    0.087   13.505    0.000    2.306    0.772
  gs =~
    Coding            1.000                               1.771    0.599
    Symbol.Search     1.441    0.142   10.141    0.000    2.551    0.818
  g =~
    gf                1.851    0.122   15.173    0.000    0.930    0.930
    gc                1.794    0.112   16.023    0.000    0.818    0.818
    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
```

```
84     gsm              0.547    0.241                  0.141    0.141
85     gs               1.464    0.263                  0.467    0.467
```

Notice the lack of `std.lv=TRUE` argument in the `cfa()` function, which is equivalent to including the argument `std.lv=FALSE` because that is the default value for the function.

To estimate the direct impact of *g* and the first-order factors on the WISC-IV subtests, use Wright's (Wright, 1934, 1968) rules (cf. (Loehlin, 2004)). Specifically, the factor loadings of the subtests on *g* are computed by multiplying the factor loading of each subtest on the corresponding first-order factor by the factor loading of this first-order factor on *g*. For example, the standardized loading of the Comprehension subtest score on *g* is computed as $.762 \times .818 = .623$. Further, the loadings of the subtests on a specific factor can be computed by multiplying the factor loading of each subtest on the corresponding first-order factor by the standard deviation of the corresponding specific factor. For example, the standardized loading of the Comprehension subtest score on *Gc* is $.762 \times .576 = .439$ (the [standardized] variance of *Gc* is .332 and $\sqrt{.332} = .576$).

This model fits the data very similarly to the four-factor model, although since the higher-order model is more parsimonious (it estimates 24 parameters instead of the 26 parameters the four-factor model estimates), it is probably a better model for this data. One other way to see how well this model fit the data is to inspect the residual correlations of the first-order factors (i.e., the difference between the model-implied correlations among the first-order constructs and the corresponding correlations in the first-order factor model (McDonald, 2011)). The actual and implied correlations are given in Table 3. The residuals are given in Table 4, and range from -.04 to .03, which are likely not of much concern. Another way to see how well the model fits is to examine the amount of variance in the first-order factors explained by the second order factor (see Table 5). With this data, *g* explains between 54 and 87% of the first order factors' variances.

Table 3. Correlations among first-order factors (Actual correlations are in upper triangle and implied correlations are in the lower triangle)

height	Gf	Gc	Gsm	Gs
Gf	1.00	0.78	0.82	0.71
Gc	0.76	1.00	0.77	0.56
Gsm	0.86	0.76	1.00	0.71
Gs	0.68	0.60	0.68	1.00

Table 4. Residual correlations of first-order factors

	Gf	Gc	Gsm	Gs
Gf				
Gc	0.02			
Gsm	-0.04	0.01		
Gs	0.03	-0.04	0.03	

Table 5. Variances of first-order factors

	Observed Variance	Residual Variance	R^2
Gc	4.81	1.60	0.67
Gf	4.00	0.53	0.87
Gsm	3.87	0.55	0.86
Gs	3.16	1.46	0.54

Schmid-Leiman Transformation (Schmid & Leiman, 1957) developed a transformation of the higher-order factor model to yield uncorrelated first-order factors that represent both the second-order and the first-order factors. This transformation of the factor loadings makes them reflect the incremental influence of both general and specific abilities on the indicator variable. As this procedure just transforms the higher order factor model, the "fit" of both models will be identical (Yung, Thissen, & McLeod, 1999).

The Schmid-Leiman (S-L) transformation can be used to estimate the direct impact of *g* and the first-order constructs on the subtests. You can calculate the factor loadings of the manifest subtest scores on *g* and the first order factors using Wright's (Wright, 1934, 1968) rules as was done in the previous section. This would be tedious to do by hand for each indicator, but can be carried out efficiently using matrices following the steps outlined in (Gorsuch, 1983) (see Figure 6).

- Exploratory Factor Models
  1. Conduct an EFA of the m MVs, extracting p factors with an oblique rotation.
     (a) Save the $m \times p$ first-order factor loading matrix, $\mathbf{\Lambda}_1$.
     (b) Save the $p \times p$ first-order inter-factor correlation matrix, $\mathbf{\Phi}_1$.
  2. Using $\mathbf{\Phi}_1$, conduct an EFA and extract the second order factor.
     (a) Save the $p \times 1$ second-order factor loading matrix, $\mathbf{\Lambda}_2$.
     (b) Save the $p \times 1$ second-order factor uniquenesses, $\mathbf{u}_2^2$.
  3. Create a $p \times p$ diagonal matrix using the square root of $\mathbf{u}_2^2$, $\mathbf{d}^* = \text{tr}(\mathbf{u}_2)\mathbf{I}$.
  4. Create an $p \times p+1$ augmented matrix, $\mathbf{A}$, where $\mathbf{A} = \left[\begin{array}{c|c} \mathbf{\Lambda}_2 & \mathbf{d}^* \end{array} \right]$.
  5. Estimate the $m \times p+1$ matrix of factor loadings for the second- and first-order factors by multiplying $\mathbf{\Lambda}_1$ by $\mathbf{A}$, $\mathbf{\Lambda}_{SL} = \mathbf{\Lambda}_1\mathbf{A}$.
- Confirmatory Factor Models
  1. Fit a second-order factor model, which will give $\mathbf{\Lambda}_1$, $\mathbf{\Phi}_1$, $\mathbf{\Lambda}_2$, and $\mathbf{u}_2^2$.
  2. Create a $p \times p$ diagonal matrix using the square root of $\mathbf{u}_2^2$, $\mathbf{d}^* = \text{tr}(\mathbf{u}_2)\mathbf{I}$.
  3. Create an $p \times p+1$ augmented matrix, $\mathbf{A}$, where $\mathbf{A} = \left[\begin{array}{c|c} \mathbf{\Lambda}_2 & \mathbf{d}^* \end{array} \right]$.
  4. Estimate the $m \times p+1$ matrix of factor loadings for the second- and first-order factors by multiplying $\mathbf{\Lambda}_1$ by $\mathbf{A}$, $\mathbf{\Lambda}_{SL} = \mathbf{\Lambda}_1\mathbf{A}$.

Figure 6. Steps for Conducting the Schmid-Leiman Transformation.

The psych package (Revelle, 2012) has the schmid() function that will conduct the S-L transformation in a EFA context (see also the bi-factor rotation in the GPArotation package (Bernaards & Jennrich, 2005; Jennrich & Bentler, 2011)). The syntax below will complete the S-L transformation using the output from the second-order CFA model.

```
#Residual variances
> U2<-diag(c(.332, .135, .141, .467))
> U<-sqrt(U2)
> rownames(U)<-colnames(U)<-c("gc","gf", "gsm", "gs")
> #First Order loadings
> Loadings<-matrix(0,ncol=4,nrow=10)
> colnames(Loadings)<-c("gc","gf", "gsm", "gs")
> rownames(Loadings)<-c("Comprehension", "Information", "Similarities", "
    Vocabulary", "Matrix.Reasoning", "Picture.Concepts", "Digit.Span", "
    Letter.Number", "Coding",  "Symbol.Search")
> Loadings[1:4,1]<-c(0.762, 0.846, 0.844, 0.885) #Gc loadings
> Loadings[5:6,2]<-c(0.689, 0.566) #Gf loadings
> Loadings[7:8,3]<-c(0.661, 0.772) #Gsm loadings
> Loadings[9:10,4]<-c(0.599, 0.818) #Gs loadings
> #Second-order loadings
> Higher.Loadings<-matrix(c(.818,.930, .927, .730),ncol=1)
> colnames(Higher.Loadings)<-"g"
> rownames(Higher.Loadings)<-c("gc","gf", "gsm", "gs")
> A<-cbind(Higher.Loadings,U)
>
> #S-L transformed matrix
> Loadings%*%A
                         g        gc        gf       gsm        gs
Comprehension    0.623316 0.4390601 0.0000000 0.0000000 0.0000000
Information      0.692028 0.4874605 0.0000000 0.0000000 0.0000000
Similarities     0.690392 0.4863081 0.0000000 0.0000000 0.0000000
Vocabulary       0.723930 0.5099321 0.0000000 0.0000000 0.0000000
Matrix.Reasoning 0.640770 0.0000000 0.2531548 0.0000000 0.0000000
Picture.Concepts 0.526380 0.0000000 0.2079617 0.0000000 0.0000000
Digit.Span       0.612747 0.0000000 0.0000000 0.2482053 0.0000000
Letter.Number    0.715644 0.0000000 0.0000000 0.2898857 0.0000000
Coding           0.437270 0.0000000 0.0000000 0.0000000 0.4093410
Symbol.Search    0.597140 0.0000000 0.0000000 0.0000000 0.5589999
```

The S-L transformation orthogonalizes the relationship between the higher-order and lower-order factors. That is, first the highest order factor solution is determined, then the next highest order is determined based on the variance orthogonal to the highest order. Moreover, the S-L factors are proportionality constrained. This constraint affects the proportion of variance in the subtest scores explained by general and specific ability constructs. Specifically, for a given set of subtests, the ratios of variance attributable to the respective first-order ability to variance attributable to *g* are constrained to be the same. For example, the standardized factor loadings on the first order factor for Comprehension is .762 and for Information is .846. The standardized factor loadings of these subtests on *g* are $.762 \times .818 = .623$ and $.846 \times .818 = .692$, respectively. The variance ratio for the Comprehension subtest is $\frac{.762^2}{.623^2} = 1.50$ and for Information is $\frac{.846^2}{.692^2} = 1.50$

2.4. Hierarchical Model

An alternative to the higher-order model is a hierarchal model, which specifies that all the factors are first-order factors, only some of these first order factors are more general than others (Rindskopf & Rose, 1988). Sometimes hierarchical models are called a bi-factor models or nested-factor models (Chen, West, & Sousa, 2006; Brunner et al., 2012; Mulaik & Quartetti, 1997).

(Chen et al., 2006) write that hierarchal models should be investigated when (*a*) there is a general factor that is hypothesized to account for the commonality of the items; (*b*) there are multiple domain specific factors, each of which is hypothesized to account for the unique influence of the specific domain *over and above* the general factor; or (*c*) interest is in the domain specific factors as well as the common factor. They further argue that a hierarchal model is often better than a higher-order model as hierarchical models (*a*) can be used to study the role of domain specific factors that are independent of the general factor; (*b*) allow for the direct examination of the strength of the relationship between the first-order factors and their associated indicator variables via the factor loadings, whereas such relationships cannot be directly examined in the second-order factor models. (*c*) can be useful in testing whether a first-order factor predict external variables, over and above the general factor, as the domain specific factors are directly represented as independent factors; and (*d*) allow for the testing of measurement invariance of the domain specific factors, in addition to the general factor, whereas the second-order model only allows for invariance in the second-order factor to be directly tested for invariance because the first-order factors are represented by disturbances.

An example of a hierarchical model is shown in Figure 7. Notice that *g*, *Gc, Gf, Gsm*, and *Gs* are all first-order factors, but that *g* is uncorrelated with the other factors. Thus, this model is similar to the S-L transformation, but here there are no proportionality constraints. The syntax given below is for a hierarchical model with uncorrelated domain specific factors, but a more general model could allow them to be correlated (Rindskopf & Rose, 1988).

The specification of the hierarchical model in `lavaan` is

```
1 ## Hierarchical model
2 WISC.hierarchical.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=~ Comprehension + Information + Matrix.Reasoning + Picture.Concepts +
     Similarities + Vocabulary +  Digit.Span + Letter.Number + Coding +
     Symbol.Search
8 g~~0*gc + 0*gf + 0*gsm + 0*gs
9 gc ~~ 0*gf + 0*gsm + 0*gs
10 gf ~~ 0*gsm + 0*gs
11 gsm ~~ 0*gs
12 '
```

Lines 1-3 and 7 are very similar to the other models specified in this chapter. Lines 4-6 and lines 8-11 are different, though. Lines 4-6 specify that for *Gs, Gsm*, and *Gs*, both indicator variables are pre-multiplied by the same label, which constrains the parameter estimate to be the same (see Section 3. for more information on constriants). This has to be

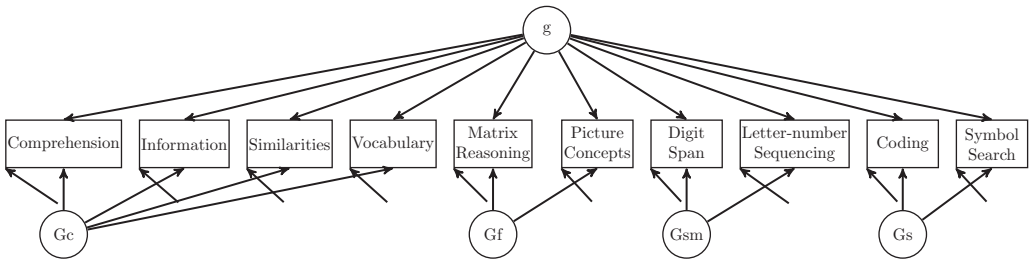

Figure 7. Hierarchical Model of the WISC-IV Subtests.

done for these latent variables or the model would be empirically underidentified (Kenny, 1979; Rindskopf, 1984).

The default in `lavaan` is for all exogenous variables to be correlated with each, so line 8 tells `lavaan` that *g* needs to be uncorrelated with *Gc, Gf, Gsm* and *Gs*, line 9 specifies that *Gc* need to be uncorrelated with *Gf, Gsm* and *Gs*, and so on. Notice that, for each variable, the parameter constraints are done through one statement instead of separate ones. The left hand side of the double tilde specifies one of the variables and the multiple + signs on the right hand side of the double tilde means that the the constraints are all going to apply to a variable's relationship with the variable of the left hand side.

The syntax to estimate the model's parameters and calculate the fit indices is the same as with the previous models, only replacing the `model` argument.

```
1 > WISC.hierarchical.fit<-cfa(model=WISC.hierarchical.model, sample.cov=
      WiscIV.cov, sample.nobs=550, std.lv=TRUE)
2 > summary(WISC.hierarchical.fit, fit.measure=TRUE, standardized=TRUE)
3 lavaan (0.5-9) converged normally after  48 iterations
4
5   Number of observations                           550
6
7   Estimator                                         ML
8   Minimum Function Chi-square                   50.345
9   Degrees of freedom                                28
10  P-value                                        0.006
11
12 Chi-square test baseline model:
13
14  Minimum Function Chi-square                 2552.014
15  Degrees of freedom                                45
16  P-value                                        0.000
17
18 Full model versus baseline model:
19
20  Comparative Fit Index (CFI)                    0.991
21  Tucker-Lewis Index (TLI)                       0.986
22
23 Loglikelihood and Information Criteria:
24
25  Loglikelihood user model (H0)              -12563.645
```

```
  Loglikelihood unrestricted model (H1)     -12538.472

  Number of free parameters                         27
  Akaike (AIC)                                25181.290
  Bayesian (BIC)                              25297.657
  Sample-size adjusted Bayesian (BIC)         25211.948

Root Mean Square Error of Approximation:

  RMSEA                                          0.038
  90 Percent Confidence Interval          0.020  0.055
  P-value RMSEA <= 0.05                          0.873

Standardized Root Mean Square Residual:

  SRMR                                           0.022

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  gc =~
    Comprhnsn         1.424    0.131   10.873    0.000    1.424    0.495
    Informatn         1.259    0.127    9.917    0.000    1.259    0.419
    Similarts         1.374    0.128   10.724    0.000    1.374    0.454
    Vocabulry         1.662    0.122   13.652    0.000    1.662    0.551
  gf =~
    Mtrx.Rsnn (a)     0.635    0.221    2.874    0.004    0.635    0.220
    Pctr.Cncp (a)     0.635    0.221    2.874    0.004    0.635    0.213
  gsm =~
    Digit.Spn (b)     0.867    0.163    5.321    0.000    0.867    0.291
    Lttr.Nmbr (b)     0.867    0.163    5.321    0.000    0.867    0.290
  gs =~
    Coding    (c)     1.460    0.114   12.794    0.000    1.460    0.494
    Symbl.Src (c)     1.460    0.114   12.794    0.000    1.460    0.468
  g =~
    Comprhnsn         1.709    0.127   13.496    0.000    1.709    0.594
    Informatn         2.190    0.124   17.674    0.000    2.190    0.728
    Mtrx.Rsnn         1.882    0.121   15.502    0.000    1.882    0.652
    Pctr.Cncp         1.568    0.132   11.898    0.000    1.568    0.527
    Similarts         2.131    0.126   16.862    0.000    2.131    0.704
    Vocabulry         2.135    0.126   16.961    0.000    2.135    0.708
    Digit.Spn         1.792    0.129   13.844    0.000    1.792    0.602
    Lttr.Nmbr         2.112    0.123   17.105    0.000    2.112    0.707
    Coding            1.280    0.133    9.614    0.000    1.280    0.433
    Symbl.Src         1.864    0.133   14.056    0.000    1.864    0.598

Covariances:
  gc ~~
    g                 0.000                               0.000    0.000
  gf ~~
    g                 0.000                               0.000    0.000
```

```
  gsm ~~
    g                 0.000                                0.000    0.000
  gs ~~
    g                 0.000                                0.000    0.000
  gc ~~
    gf                0.000                                0.000    0.000
    gsm               0.000                                0.000    0.000
    gs                0.000                                0.000    0.000
  gf ~~
    gsm               0.000                                0.000    0.000
    gs                0.000                                0.000    0.000
  gsm ~~
    gs                0.000                                0.000    0.000

Variances:
    Comprehension     3.330    0.251                       3.330    0.402
    Information       2.661    0.202                       2.661    0.294
    Similarities      2.735    0.212                       2.735    0.298
    Vocabulary        1.783    0.208                       1.783    0.196
    Matrix.Resnng     4.393    0.384                       4.393    0.527
    Pictur.Cncpts     6.004    0.444                       6.004    0.677
    Digit.Span        4.902    0.381                       4.902    0.553
    Letter.Number     3.711    0.345                       3.711    0.416
    Coding            4.977    0.409                       4.977    0.569
    Symbol.Search     4.110    0.392                       4.110    0.423
    gc                1.000                                1.000    1.000
    gf                1.000                                1.000    1.000
    gsm               1.000                                1.000    1.000
    gs                1.000                                1.000    1.000
    g                 1.000                                1.000    1.000
```

All of the fit indices indicate that this model fits the data fairly well, and arguably better than the other models of this data.

One nice feature of the hierarchical model that the higher-order model does not have is that it can be used to test if the domain specific factors (i.e., *Gf, Gc, Gsm , Gs*) explain any additional variance above and beyond the general factor, (i.e, *g*) by removing the domain specific factors from the model and examining how it fits the data. For example, the results from the hierarchical model indicate that factor loadings for the *Gf* and *Gsm* are relatively small (standardized loadings <0.30), so we can fit a model without these factors.

```
> WISC.hierarchical2.model<-'
+ gc =~ Comprehension + Information +  Similarities + Vocabulary
+ gs =~ c*Coding + c*Symbol.Search
+
+ g=~ Comprehension + Information + Matrix.Reasoning + Picture.Concepts +
      Similarities + Vocabulary +  Digit.Span + Letter.Number + Coding +
     Symbol.Search
+ g~~0*gc + 0*gs
+ gc ~~ 0*gs
+ '
>
> WISC.hierarchical2.fit<-cfa(model=WISC.hierarchical2.model, sample.cov=
    WiscIV.cov, sample.nobs=550, std.lv=TRUE)
> summary(WISC.hierarchical2.fit, fit.measure=TRUE, standardized=TRUE)
```

```
lavaan (0.5-9) converged normally after  36 iterations

  Number of observations                           550

  Estimator                                         ML
  Minimum Function Chi-square                   60.949
  Degrees of freedom                                30
  P-value                                        0.001

Chi-square test baseline model:

  Minimum Function Chi-square                 2552.014
  Degrees of freedom                                45
  P-value                                        0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.988
  Tucker-Lewis Index (TLI)                       0.981

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -12568.947
  Loglikelihood unrestricted model (H1)      -12538.472

  Number of free parameters                         25
  Akaike (AIC)                               25187.894
  Bayesian (BIC)                             25295.642
  Sample-size adjusted Bayesian (BIC)        25216.281

Root Mean Square Error of Approximation:

  RMSEA                                          0.043
  90 Percent Confidence Interval          0.027  0.059
  P-value RMSEA <= 0.05                          0.744

Standardized Root Mean Square Residual:

  SRMR                                           0.023

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  gc =~
    Comprhnsn         1.435    0.125   11.516    0.000    1.435    0.499
    Informatn         1.337    0.119   11.228    0.000    1.337    0.445
    Similarts         1.436    0.121   11.896    0.000    1.436    0.474
    Vocabulry         1.694    0.115   14.734    0.000    1.694    0.561
  gs =~
    Coding    (c)     1.460    0.113   12.905    0.000    1.460    0.494
    Symbl.Src (c)     1.460    0.113   12.905    0.000    1.460    0.468
```

```
 67   g =~
 68     Comprhnsn          1.695    0.124   13.625    0.000    1.695    0.589
 69     Informatn          2.140    0.122   17.472    0.000    2.140    0.712
 70     Mtrx.Rsnn          1.873    0.119   15.699    0.000    1.873    0.649
 71     Pctr.Cncp          1.592    0.128   12.414    0.000    1.592    0.535
 72     Similarts          2.092    0.125   16.780    0.000    2.092    0.691
 73     Vocabulry          2.106    0.124   17.009    0.000    2.106    0.698
 74     Digit.Spn          1.878    0.124   15.161    0.000    1.878    0.631
 75     Lttr.Nmbr          2.178    0.119   18.248    0.000    2.178    0.729
 76     Coding             1.285    0.132    9.738    0.000    1.285    0.435
 77     Symbl.Src          1.857    0.132   14.106    0.000    1.857    0.596
 78
 79 Covariances:
 80   gc ~~
 81     g                  0.000                              0.000    0.000
 82   gs ~~
 83     g                  0.000                              0.000    0.000
 84   gc ~~
 85     gs                 0.000                              0.000    0.000
 86
 87 Variances:
 88     Comprehension      3.346    0.248                     3.346    0.404
 89     Information        2.677    0.204                     2.677    0.296
 90     Similarities       2.727    0.213                     2.727    0.298
 91     Vocabulary         1.799    0.204                     1.799    0.198
 92     Coding             4.963    0.408                     4.963    0.567
 93     Symbol.Search      4.139    0.391                     4.139    0.426
 94     Matrix.Resnng      4.828    0.341                     4.828    0.579
 95     Pictur.Cncpts      6.331    0.415                     6.331    0.714
 96     Digit.Span         5.336    0.371                     5.336    0.602
 97     Letter.Number      4.179    0.328                     4.179    0.468
 98     gc                 1.000                              1.000    1.000
 99     gs                 1.000                              1.000    1.000
100     g                  1.000                              1.000    1.000
```

While difference in $\chi - square$ is moderately large ($\chi^2_{df=1} = 10.60$, $p = .01$), none of the other measures of model fit are appreciably different between the two models. Moreover, the change in R^2 values for the four subtests affected by the removal of the two factors (Matrix Reasoning, Picture Concepts, Digit Span, and Letter-Number Sequencing) only change by .04-.05 units. Consequently, it appears that the covariance among the *Gf* and *Gsm* subtests are mostly explained by *g*, alone.

3. Behavior Genetic Analysis

Behavior genetics is the study of the genetic and environmental influences on psychological traits (Plomin, DeFries, McClearn, & McGuffin, 2008). It can answer many questions that traditional research designs cannot, such as the interaction of genotype with sex, age and lifestyle factors, cultural transmission, genetic and environmental stability over time, and the causes of co-morbidity between traits and diseases (Boomsma, Busjahn, & Peltonen, 2002). Traditionally, these analysis have used "natural experiments" to study these relationships (Rutter, 2007), such as twins reared in differing environments (Bouchard, 1991)

and adoption studies (Jensen, 1997), as well as the general study of sibling similarities and differences (Jensen, 1980; Dunn & Plomin, 1992).

Initially, studies in this field used variations of regression models in conjunction with the kinship of individuals within the sample to partition the phenotypic variance into genetic and environmental components (DeFries & Fulker, 1985; Falconer, 1960), and such models are still used (Lynch & Walsh, 1998; Falconer & Mackay, 1996). With the advent of more powerful computers and data analysis programs, the field has moved to using latent variable models for much of its analyses (Neale & Maes, 1992). That trend is especially evident by the development of the the *Mx* program (Neale, Boker, Xie, & Maes, 2003) and its translation into the `R` language via *OpenMx* (Boker et al., 2011).

The use of latent variable models for behavioral genetic data requires two extensions from the models this chapter has covered thus far. First, the ability to estimate parameters for different groups concurrently, and, second, the ability to constrain parameter estimates within and between groups.

To input the data from multiple groups in `lavaan`, the data from the different groups needs to be combined into a single `list`. In `R`, a list is a single objected made up of an ordered (and possibly named) collection of objects (technically called *components*). As an example, say we obtained Full Scale IQ (FSIQ) scores in a set of monozygotic (MZ) and dizygotic (DZ) twins. The correlation of the scores between the MZ twins was .86, but for the DZ the correlation was .60 ((Bouchard & McGue, 1981) p. 1056). To enter both datasets in `R` as a list, first we need to enter the correlation matrices for the MZ and DZ twins, separately.

```
1 > MZ.cor<-matrix(c(1,.86,.86,1),nrow=2)
2 > DZ.cor<-matrix(c(1,.60,.60,1),nrow=2)
```

Then, we need to name the rows and columns in each matrix

```
1 > rownames(MZ.cor)<-c("P1", "P2")
2 > colnames(MZ.cor)<-c("P1", "P2")
3 > rownames(DZ.cor)<-c("P1", "P2")
4 > colnames(DZ.cor)<-c("P1", "P2")
```

where "P1" stands for the measured phenotype from twin 1, and "P2" stands for the measured phenotype from twin 2.

Assuming the standard deviation is 15 and does not differ between groups (Koeppen-Schomerus, Spinath, & Plomin, 2003), `lavaan`'s `cor2cov()` function can convert each correlation matrix to a covariance matrix.

```
1 > DZ.cov<-cor2cov(DZ.cor, c(15,15))
2 > MZ.cov<-cor2cov(MZ.cor, c(15,15))
```

To combine the two matrices into a list, use the `list()` function.

```
1 sib.cov<-list(MZ=MZ.cov,DZ=DZ.cov)
```

In the previous syntax, the list components were entered in the form of *name=object*. Giving the list elements names will help in parameter interpretation, as `lavaan` will use the names in the `summary()` output.

To examine a list in the `R` console, type the list's name.

```
> sib.cov
$MZ
      P1    P2
P1 225.0 193.5
P2 193.5 225.0

$DZ
    P1  P2
P1 225 135
P2 135 225
```

The same syntax can be used to enter the twin groups' sample sizes, although the data can be entered as scalars (i.e., one-dimensional matrices) instead of a matrix.

```
> sib.n<-list(MZ=4672,DZ=5546)

> sib.n
$MZ
[1] 4672

$DZ
[1] 5546
```

Because the covariances from both groups are combined into a single list object, the `lavaan` syntax to estimate the parameters the will look the same as with the other CFA models, e.g., `cfa(model, sample.cov=sib.cor, sample.nobs=sib.n)`. The only thing left to do is to specify the model. In order to do so, though, certain parameters need constrained.

Constraining parameters in `lavaan` can take one of two forms. First, to constrain a parameter to a specific value, just premultiply the parameter by that value, e.g., `g =~3*Comprehension`. To constrain a parameter estimate to be the same for $m > 1$ groups, then the constrained parameter needs to have the same label in all the groups. This can be done in one of two ways. The first way is by specifying *m* separate models and giving the parameters to be constrained the same label. The second way is by specifying one model, but using R's concatenate function, `c()`, to apply multiple labels, e.g., for two groups: `g = c(a,a)*Comprehension`.

3.1. Behavior Genetic Models

3.1.1. Twin Models

Behavior genetic models can be quite complex, depending the degree(s) of kinship present in the data, as well as the number of phenotypes (i.e., observable behavior) measured. For the "classic" model though, we will assume there is just one phenotype measured in both MZ and DZ twins. (For some multivariate extensions of this model, see, e.g. (Loehlin, 2004; Neale et al., 2003)). The genetic influence on the phenotype can be decomposed into (*a*) additive effects of alleles at various loci, (*b*) dominance effects of alleles at various loci, and (*c*) epistatic interactions between loci (Plomin et al., 2008). Often with human samples, epistatic and dominance effects are confounded, so are lumped into a single *non-additive* genetic effects category. The environmental influence on the phenotype can be

decomposed into (*a*) effects due to a *shared* environment, such as being raised by the same parents in the same house (aka "between-family" effects); and (*b*) effects due to an *unshared* environment, such as having different peers or attending different schools (aka "within-family" effects). These unshared effects also include random environmental events, such as getting into automobile accident, as well as random measurement events (i.e, measurement error).

For relatives *i* and *j*, their phenotypes, P_i and P_j, are assumed to be a linear function of the additive genetic influence (A_i and A_j), non-additive influence (D_i and D_j), shared environmental influence (C_i and C_j) and unshared environmental variance (E_i and E_j). Thus,

$$\begin{aligned} P_1 &= a_1 A_1 + d_1 D_1 + c_1 C_1 + e_1 E_1 \\ P_2 &= a_2 A_2 + d_2 D_2 + c_2 C_2 + e_2 E_2 \end{aligned} \tag{2}$$

For a given set of twins, the influences of A, C, D, and E are not expected to differ. That is, we would not expect, say, the additive genetic influence estimates (a_1 and a_2) or the unshared environmental influence estimates (e_1 and e_2) to differ. Thus, Equation 2 can be simplified to

$$\begin{aligned} P_1 &= aA_1 + dD_1 + cC_1 + eE_1 \\ P_2 &= aA_2 + dD_2 + cC_2 + eE_2 \end{aligned} \tag{3}$$

If twins (no matter the zygosity) are reared together, the shared environment influence (C) is going to be the same. Likewise, by its definition, the non-shared environment (E) is going to be completely different. From quantitative genetic theory, we know that for MZ twins, the genetic influence (i.e., A and D) on a trait will be the same for both twins. For DZ twins, on average, the additive genetic influence will only be 1/2 the same and the non-additive genetic influence will be be 1/4 the same.[1] From this information, we can build a path model showing the relationship on a given phenotype between two twin siblings, as is shown in Figure 8.

Using the labels in Figure 8, the expected correlations on the measured phenotype for MZ and DZ twins reared together are

$$_{MZ}r_{P_1,P_2} = a^2 + d^2 + c^2 \tag{4}$$

and

$$_{DZ}r_{P_1,P_2} = 0.5a^2 + 0.25d^2 + c^2 \tag{5}$$

respectively. The variance for the trait (for either sibling) is

$$\sigma_P^2 = a^2 + d^2 + c^2 + e^2 \tag{6}$$

Together, equations 4, 5, and 6 represent four unknown parameters (*a, b, c* and *d*), but use input from only three known statistics ($_{MZ}r_{P_1,P_2}$, $_{DZ}r_{P_1,P_2}$, and σ_P^2). Thus, a "classic" twin design can only estimate three of the four parameters (Martin, Eaves, Kearsey, &

[1]This assumes random mating and that the additive genetic influence is uncorrelated with shared environmental or dominance influence.

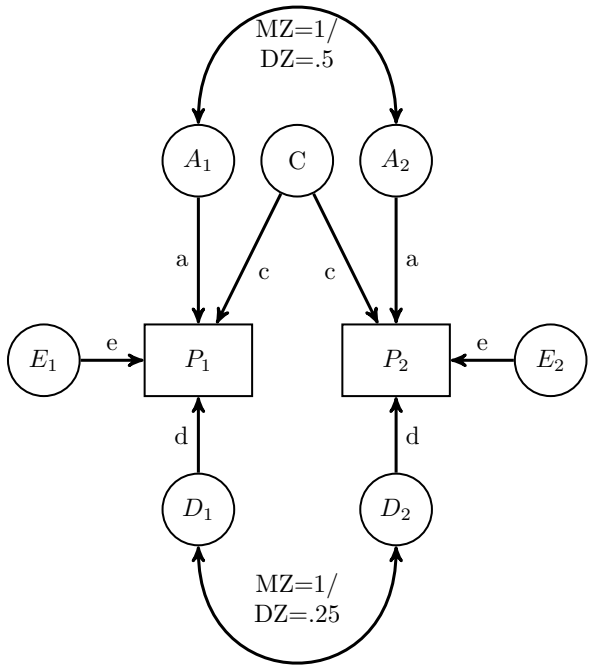

Figure 8. ACDE Model.

Davies, 1978). To estimate all the parameters within the same model would require additional data (e.g., twins separated at birth, relatives of twins (Neale & Maes, 1992)). Not all the parameters need to be estimated, however, if, say, the hypothesis is that only additive genetic components and random environmental events are influencing the phenotype. In fact, with twin designs it is typical to test the the following series of models that postulate different genetic and environmental components are influencing behavior: ACE, ADE, AE, CE and E (Maes, 2005).

Below is the `lavaan` syntax for the full ACDE model for a single measured variable, using MZ and DZ twins raised together. The model assumes the MZ twins are the first set of data entered into the R list. To estimate other models, just comment out the unnecessary parameters.

```
ACDE.model<-'
#Latent Variables
A1=~ NA*P1 + c(a,a)*P1
A2=~ NA*P2 + c(a,a)*P2
C=~ NA*P1 + c(c,c)*P1 + c(c,c)*P2
D1 =~ NA*P1 + c(d,d)*P1
D2 =~ NA*P2 +  c(d,d)*P2

#Variances
A1 ~~ 1*A1
A2 ~~ 1*A2
C~~ 1*C
D1 ~~ 1*D1
```

```
14 D2 ~~ 1*D2
15 P1~~c(e2,e2)*P1
16 P2~~c(e2,e2)*P2
17
18 #covariances
19 A1 ~~ c(1,.5)*A2 + 0*C + 0*D1 + 0*D2
20 A2 ~~ 0*C + 0*D1 + 0*D2
21 C ~~ 0*D1 + 0*D2
22 D1 ~~ c(1,.25)*D2
23 '
```

Lines 3-6 define the latent variables *A1*, *A2*, *C*, *D1*, and *D2* respectively. The `c(a,a)*P1` syntax tells `lavaan` to use the label *a* for the parameter estimating the influence from A1 to P1 in both the MZ and DZ groups. As the parameter estimating the influence from A2 to P2 has the same label, `lavaan` will constrain it to be the same value as well. A similar interpretation follows for the other `c(·)*` terms.

This model does not create a *E1* or *E2* latent variable because, at least for a design with MZ and DZ twins raised together, these terms represent error/residual variance. Consequently, this model constrains the error variances to be the same for for the MZ and DZ twins (meaning e^2 is estimated instead of *e* – lines 15-16). The estimate for *e* can be obtained via $e = \sqrt{e^2}$.

Lines 10-14 constrain the variances of *A1*, *A2*, and *C* to unity, while lines 19-22 constrain the correlation of both *A* variables with *C* and both *D* variables to zero. The correlation of all residual terms in `lavaan` are already constrained to zero; thus, there is no need to specify anything about *E1* and *E2*'s covariance. Lines 19 and 22 serve a dual purpose. In addition to making correlations equal zero, line 19 also constrains the between-twin *A* correlations to be 1 in the first group (MZ twins) and .5 in the second group (DZ twins). Likewise, line 22 constrains the between-twin *D* correlations to be 1 for MZ twins and .25 in DZ twins.

The results for the AE model are given below.

```
 1 > AE.fit<-cfa(AE.model, sample.cov=sib.cov, sample.nobs=sib.n)
 2 > summary(AE.fit, fit.measures=TRUE, standardized=TRUE)
 3 lavaan (0.5-9) converged normally after  27 iterations
 4
 5   Number of observations per group
 6   MZ                                              4672
 7   DZ                                              5546
 8
 9   Estimator                                         ML
10   Minimum Function Chi-square                  321.458
11   Degrees of freedom                                 4
12   P-value                                        0.000
13
14 Chi-square for each group:
15
16   MZ                                            11.971
17   DZ                                           309.487
18
19 Chi-square test baseline model:
20
21   Minimum Function Chi-square                 8761.454
```

```
  Degrees of freedom                                 2
  P-value                                        0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.964
  Tucker-Lewis Index (TLI)                       0.982

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -80117.143
  Loglikelihood unrestricted model (H1)     -79956.414

  Number of free parameters                          2
  Akaike (AIC)                              160238.287
  Bayesian (BIC)                            160252.751
  Sample-size adjusted Bayesian (BIC)       160246.395

Root Mean Square Error of Approximation:

  RMSEA                                          0.125
  90 Percent Confidence Interval          0.113  0.136
  P-value RMSEA <= 0.05                          0.000

Standardized Root Mean Square Residual:

  SRMR                                           0.085

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

Group 1 [MZ]:

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  A1 =~
    P1        (a)    13.570    0.091  148.344    0.000   13.570    0.927
  A2 =~
    P2        (a)    13.570    0.091  148.344    0.000   13.570    0.927

Covariances:
  A1 ~~
    A2                1.000                              1.000    1.000

Variances:
    A1                1.000                              1.000    1.000
    A2                1.000                              1.000    1.000
    P1       (e2)    29.930    0.611                    29.930    0.140
    P2       (e2)    29.930    0.611                    29.930    0.140


Group 2 [DZ]:
```

```
77
78                    Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
79 Latent variables:
80   A1 =~
81     P1         (a)    13.570    0.091  148.344    0.000   13.570    0.927
82   A2 =~
83     P2         (a)    13.570    0.091  148.344    0.000   13.570    0.927
84
85 Covariances:
86   A1 ~~
87     A2                 0.500                              0.500    0.500
88
89 Variances:
90     A1                 1.000                              1.000    1.000
91     A2                 1.000                              1.000    1.000
92     P1         (e2)    29.930    0.611                   29.930    0.140
93     P2         (e2)    29.930    0.611                   29.930    0.140
```

The results look a little different than previous output. First, the model's parameter estimates are given in two separate sections, one for Group 1 and the other for Group 2. These groups are labeled MZ and DZ, respectively, because that is what we named them in the `list()` function. To get percent of variance in cognitive ability explained by additive genetic influences from AE model (i.e., the narrow heritability), square the standardized a term (i.e., the term in the `Std.all` column): $a^2 = .927^2 \approx .86$. Likewise, the amount of variance explained by the unshared environment is $e^2 = .14$. These values should sum up to one, as they do in this model.

Table 6 has the fit values for all the models. The AE and CE models did not appear to fit the data very well, and the E model (i.e., the null model), fit the data horribly. The ADE model shows the exact same fit as the AE model, which normally would be odd, but can be explained by examining the estimated value for d: 0.00. The fact that there was no dominance influence is not surprising, given that 1/2 of the MZ correlation ($\frac{.86}{2} = .43$) is less than the DZ twin correlation (.60) (Evans, Gillespie, & Martin, 2002). Thus, it appears that the ACE model is the best fitting model of the five, and in fact the ACE model fit the data extremely well, as its estimated values for a, c and e were able to re-create the covariances perfectly.

Table 6. Goodness-of-fit statistics and parameter estimates for MZ and DZ twin models of cognitive ability

Model	χ^2	*df*	SRMR	RMSEA	a^2	c^2	e^2	d^2
ACE	$< 0.00^a$	3	0.00	0.00	0.52	0.34	0.14	–
AE	321.46	4	0.09	0.13	0.86	–	0.14	–
CE	1329.95	4	0.07	0.26	–	0.72	0.28	–
E	8761.45	5	0.42	0.59	–	–	1.00	–
ADE	321.46	3	0.09	0.14	0.86	–	0.14	0.00

Note. SRMR: standardized root mean square error; RMSEA: root mean square error of approximation.
[a] $\chi^2 = 0.000001843065$.

3.1.2. Non-Twin Sibling Models

While twin data can be very powerful, it can be difficult to collect such data. Collecting sibling data, however, is much easier, and can also give interesting information about environmental influences (Plomin, 1994; Jensen, 1980), specifically the power of the shared and non-shared environment for a given trait (Turkheimer & Waldron, 2000).

A non-twin sibling model for a single phenotype is shown in Figure 9. In such analyses, the goal is to examine the influence of shared versus non-shared environmental influences, so there are only two parameters estimated: *e* and *c*. As with the twin models, e^2 is the amount of variance explained by the non-shared environment, while c^2 is the amount of variance explained by the shared environment. For an example of a multivariate non-twin sibling analysis, see, e.g., (Kretschmer & Pike, 2010).

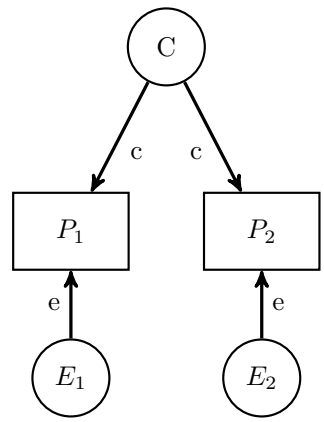

Figure 9. Univariate Sibling Analysis.

Some example data come from (Foong, Beaujean, Frisby, & Knoop, 2012), who gathered cognitive ability information on a group of siblings ($n = 17$) in middle school using the Woodcock Johnson-Third Edition Tests of Cognitive Abilities (Woodcock, McGrew, & Mather, 2001). For this data, the siblings were spilt by age into into older (*P1*) and younger (*P2*) sibling categories. The `lavaan` syntax to input and name the covariance matrix is given below.

```
#Sibling data
BIA.cov<-matrix(c(293.4706, 119.1287, 119.1287, 220.7574),ncol=2)
colnames(BIA.cov)<-rownames(BIA.cov)<-c("P1", "P2")
```

Specifying the model is very similar to the ACDE type models, only here we are just specifying the C and E components of that model.

```
sibling.model<-'
Shared =~ NA*P1 + c*P1 + c*P2
P1~~e2*P1
P2~~e2*P2
Shared~~1*Shared
'
```

The results from this model are given below.

```
sibling.fit<-cfa(sibling.model, sample.cov=BIA.cov, sample.nobs=17)
summary(sibling.fit, standardized=TRUE)
lavaan (0.5-9) converged normally after  27 iterations

  Number of observations                            17

  Estimator                                         ML
  Minimum Function Chi-square                    0.438
  Degrees of freedom                                 1
  P-value                                        0.508

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  Shared =~
    P1        (c)    10.589    3.054    3.467    0.001   10.589    0.681
    P2        (c)    10.589    3.054    3.467    0.001   10.589    0.681

Variances:
    P1       (e2)   129.869   44.545                    129.869    0.537
    P2       (e2)   129.869   44.545                    129.869    0.537
    Shared            1.000                              1.000    1.000
```

The results indicate that the model fit the data relatively well ($\chi^2 = .438$). According to this model, the shared environment explains approximately 46% of the variance (i.e., $.681^2 = .46$), while the unique environment explains approximately 54% of the variance.

4. Conclusion

This chapter has shown how `R` (R Development Core Team, 2011) can be a powerful tool in the analysis of latent variable models of cognitive ability data, specifically focussing on the `lavaan` (Rosseel, 2012) package, which is designed especially for structural equation modeling with latent variables. With such tools, the analysis of latent variables models can be accessible to any who are interested in such models. When added to fact that the only cost involved in using `R` is the time it takes to learn the language, it is easy to see why (Kelley, Lai, & Wu, 2008) write that "there is no time like the present to begin incorporating `R` into one's set of statistical tools" (p. 569).

References

Beaujean, AA. 2012. BaylorEdPsych: R package for Baylor University Educational Psychology quantitative courses (Version 0.5) BaylorEdPsych: R package for Baylor University Educational Psychology quantitative courses (Version 0.5) [Computer software]. Waco, TXBaylor University.

Bernaards, CA., & Jennrich, RI. 2005. Gradient projection algorithms and software for arbitrary rotation criteria in factor analysis Gradient projection algorithms and software for arbitrary rotation criteria in factor analysis. Educational and Psychological Measurement655676-696. 10.1177/0013164404272507

Boker, S., Neale, M., Maes, H., Wilde, M., Spiegel, M., Brick, T., Spies, J., Estabrook, R., Kenny, S., Bates, T., Mehta, P., & Fox, J. 2011. OpenMx: An open source extended structural equation modeling framework OpenMx: An open source extended structural equation modeling framework. Psychometrika762306-317. 10.1007/s11336-010-9200-6

Boomsma, D., Busjahn, A., & Peltonen, L. 2002. Classical twin studies and beyond Classical twin studies and beyond. Nature Reviews Genetics311872-882. 10.1038/nrg932

Bouchard, TJ. 1991. A twice-told tale: Twins reared apart A twice-told tale: Twins reared apart. In D. Cicchetti & WM. Grove (Eds.), Thinking clearly about psychology: Essays in honor of Paul E. Meehl, Vol. 1: Matters of public interest Vol. 2: Personality and psychopathology. Thinking clearly about psychology: Essays in honor of Paul E. Meehl, Vol. 1: Matters of public interest Vol. 2: Personality and psychopathology. (p. 188-215). Minneapolis, MNUniversity of Minnesota Press.

Bouchard, TJ., & McGue, M. 1981. Familial studies of intelligence: A review Familial studies of intelligence: A review. Science21244981055-1059. 10.1126/science.7195071

Brunner, M., Nagy, G., & Wilhelm, O. 2012. A tutorial on hierarchically structured constructs A tutorial on hierarchically structured constructs. Journal of Personality804796-846. 10.1111/j.1467-6494.2011.00749.x

Carroll, JB. 1993. Human cognitive abilities: A survey of factor-analytic studies Human cognitive abilities: A survey of factor-analytic studies. New YorkCambridge University Press.

Chen, FF., West, SG., & Sousa, KH. 2006. A comparison of bifactor and second-order models of quality of life A comparison of bifactor and second-order models of quality of life. Multivariate Behavioral Research412189-225. 10.1207/s15327906mbr4102$_5$

Cudeck, R. 1989. Analysis of correlation matrices using covariance structure models Analysis of correlation matrices using covariance structure models. Psychological Bulletin1052317-327. 10.1037/0033-2909.105.2.317

DeFries, JC., & Fulker, DW. 1985. Multiple regression analysis of twin data Multiple regression analysis of twin data. Behavior Genetics155467-473. 10.1007/bf01066239

Dunn, J., & Plomin, R. 1992. Separate lives: Why siblings sre so different Separate lives: Why siblings sre so different. New YorkWiley.

Evans, DM., Gillespie, NA., & Martin, NG. 2002. Biometrical genetics Biometrical genetics. Biological Psychology611-233-51. 10.1016/s0301-0511(02)00051-0

Falconer, DS. 1960. Introduction to quantitative genetics Introduction to quantitative genetics. LondonOliver and Boyd.

Falconer, DS., & Mackay, TFC. 1996. Introduction to quantitative genetics Introduction to quantitative genetics (4th ed.). Harlow, Essex, UKAddison Wesley Longman.

Foong, C., Beaujean, AA., Frisby, CL., & Knoop, A. 2012. Environmental influence on mathematical reaction time: A sibling study. Manuscript submitted for publication Environmental influence on mathematical reaction time: A sibling study. Manuscript

submitted for publication.

Gorsuch, RL. 1983. Factor analysis Factor analysis. Hillsdale, NJLawrence Erlbaum.

Jennrich, R., & Bentler, P. 2011. Exploratory bi-factor analysis Exploratory bi-factor analysis. Psychometrika764537-549. 10.1007/s11336-011-9218-4

Jensen, AR. 1980. Uses of sibling data in educational and psychological research Uses of sibling data in educational and psychological research. American Educational Research Journal172153-170. 10.2307/1162480

Jensen, AR. 1997. Adoption data and two g-related hypotheses Adoption data and two g-related hypotheses. Intelligence2511-6. 10.1016/s0160-2896(97)90003-9

Jensen, AR. 1998. The g factor: The science of mental ability The g factor: The science of mental ability. Westport, CTPraeger Publishers/Greenwood Publishing Group. Print

Kelley, K., Lai, K., & Wu, PJ. 2008. Using R for data analysis: A best practice for research Using R for data analysis: A best practice for research. In JW. Osborne (Ed.), Best practices in quantitative methods Best practices in quantitative methods (p. 535-572). Thousand Oaks, CASage.

Kenny, DA. 1979. Correlation and causality Correlation and causality. New YorkJohn Wiley.

Koeppen-Schomerus, G., Spinath, FM., & Plomin, R. 2003. Twins and non-twin siblings: Different estimates of shared environmental influence in early childhood Twins and non-twin siblings: Different estimates of shared environmental influence in early childhood. Twin Research and Human Genetics60297-105. doi:10.1375/twin.6.2.97

Kretschmer, T., & Pike, A. 2010. Links between nonshared friendship experiences and adolescent siblings' differences in aspirations Links between nonshared friendship experiences and adolescent siblings' differences in aspirations. Journal of Adolescence331101-110. 10.1016/j.adolescence.2009.05.001

Loehlin, JC. 2004. Latent variable models: An introduction to factor, path, and structural equation analysis Latent variable models: An introduction to factor, path, and structural equation analysis (4th ed.). Mahwah, NJErlbaum.

Lynch, M., & Walsh, B. 1998. Genetics and analysis of quantitative traits Genetics and analysis of quantitative traits. Sunderland, MASinauer.

Maes, HH. 2005. ACE Model ACE Model. In BS. Everitt & DC. Howell (Eds.), Encyclopedia of Statistics in Behavioral Science. Encyclopedia of Statistics in Behavioral Science. John Wiley and Sons, Ltd.

Martin, NG., Eaves, LJ., Kearsey, MJ., & Davies, P. 1978. The power of the classical twin study The power of the classical twin study. Heredity40197-116. 10.1038/hdy.1978.10

McDonald, RP. 1989. An index of goodness-of-fit based on noncentrality An index of goodness-of-fit based on noncentrality. Journal of Classification6197-103. 10.1007/bf01908590

McDonald, RP. 2011. Measuring latent quantities Measuring latent quantities. Psychometrika764511-536. 10.1007/s11336-011-9223-7

Mulaik, SA., & Quartetti, DA. 1997. First order or higher order general factor? First order or higher order general factor? Structural Equation Modeling: A Multidisciplinary Journal43193-211. 10.1080/10705519709540071

Neale, MC., Boker, SM., Xie, G., & Maes, HH. 2003. Mx: Statistical modeling Mx: Statistical modeling (6th ed.). Richmond, VAVirginia Commonwealth University. http://www.vipbg.vcu.edu/ vipbg/software/mxmanual.pdf

Neale, MC., & Maes, HHM. 1992. Methodology for genetic studies of twins and families Methodology for genetic studies of twins and families. Dordrecht, The NetherlandsKluwer Academic Publishers.

Newton, JH., & McGrew, KS. 2010. Introduction to the special issue: Current research in Cattell-Horn-Carroll-based assessment Introduction to the special issue: Current research in Cattell-Horn-Carroll-based assessment. Psychology in the Schools477621-634. 10.1002/pits.20495

Parkin, J., & Beaujean, AA. 2012. The effects of Wechsler Intelligence Scale for Children-Fourth Edition cognitive abilities on math achievement The effects of Wechsler Intelligence Scale for Children-Fourth Edition cognitive abilities on math achievement. Journal of School Psychology50113-128. 10.1016/j.jsp.2011.08.003

Plomin, R. 1994. The Emanuel Miller Memorial Lecture 1993: Genetic research and identification of environmental influences The Emanuel Miller Memorial Lecture 1993: Genetic research and identification of environmental influences. Journal of Child Psychology and Psychiatry355817-834. 10.1111/j.1469-7610.1994.tb02297.x

Plomin, R., DeFries, JC., McClearn, GE., & McGuffin, P. 2008. Behavioral genetics Behavioral genetics (5th ed.). New YorkWorth.

R Development Core Team. 2011. R: A language and environment for statistical computing R: A language and environment for statistical computing. Vienna, AustriaR Foundation for Statistical Computing. Available from http://www.R-project.org.

Revelle, W. 2012. psych: Procedures for psychological, psychometric, and personality research (Version 1.2.4) [Computer software]. psych: Procedures for psychological, psychometric, and personality research (Version 1.2.4) [Computer software]. Evanston, ILNorthwestern University.

Rindskopf, D. 1984. Structural equation models: Empirical identification, Heywood cases, and related problems Structural equation models: Empirical identification, Heywood cases, and related problems. Sociological Methods and Research131109-119. 10.1177/0049124184013001004

Rindskopf, D., & Rose, T. 1988. Some theory and applications of confirmatory second-order factor analysis Some theory and applications of confirmatory second-order factor analysis. Multivariate Behavioral Research23151-67. 10.1207/s15327906mbr2301$_3$

Rosseel, Y. 2012. lavaan: An R package for structural equation modeling lavaan: An R package for structural equation modeling. Journal of Statistical Software4821-36.

Rutter, M. 2007. Proceeding from observed correlation to causal inference: The use of natural experiments Proceeding from observed correlation to causal inference: The use of natural experiments. Perspectives on Psychological Science24377-395. 10.1111/j.1745-6916.2007.00050.x

Schmid, J., & Leiman, J. 1957. The development of hierarchical factor solutions The development of hierarchical factor solutions. Psychometrika22153-61. 10.1007/bf02289209

Spearman, CE. 1927. The abilities of man: Their nature and measurement The abilities of

man: Their nature and measurement. New YorkBlackburn Press.

Turkheimer, E., & Waldron, MC. 2000. Nonshared environment: A theoretical, methodological and quantitative review Nonshared environment: A theoretical, methodological and quantitative review. Psychological Bulletin126178-108. 10.1037/0033-2909.126.1.78

Vance, A. 2009Jan 6. Data analysts captivated by R's power Data analysts captivated by R's power. New York Times.

Wechsler, D. 2003a. Wechsler intelligence scale for children Wechsler intelligence scale for children (4th ed.). San Antonio, TXThe Psychological Corporation.

Wechsler, D. 2003b. WISC-IV technical and interpretive manual WISC-IV technical and interpretive manual. San Antonio, TXThe Psychological Corporation.

Woodcock, RW., McGrew, KS., & Mather, N. 2001. Woodcock-Johnson tests of cognitive abilities Woodcock-Johnson tests of cognitive abilities (3rd ed.). Itasca, ILRiverside.

Wright, S. 1934. The method of path coefficients The method of path coefficients. The Annals of Mathematical Statistics53161-215.

Wright, S. 1968. Evolution and the genetics of populations: Genetics and biometric foundations Evolution and the genetics of populations: Genetics and biometric foundations (Vol. 1). ChicagoUniversity of Chicago Press.

Yung, YF., Thissen, D., & McLeod, LD. 1999. On the relationship between the higher-order factor model and the hierarchical factor model On the relationship between the higher-order factor model and the hierarchical factor model. Psychometrika642113-128. 10.1007/bf02294531