Structural Equation Modeling R Demonstration Notes

Daniel J. Bauer & Patrick J. Curran

*This material (Version 2019.3) was developed in support of the workshop **Structural Equation Modeling** presented on May 13 - 17, 2019, by Dan Bauer and Patrick Curran in Chapel Hill NC.*

R, RStudio, lavaan, and all R-based packages undergo continual updating and expansion. These materials were developed using R version 3.5.2 (Windows) and lavaan version 0.6-3. The scripts may require modification when used with earlier or later versions to function correctly.

Although Curran-Bauer Analytics encourages the use of these materials in whatever way contributes to a research and teaching endeavor, we are unable to respond to specific questions that might arise in practice. However, please share any errors or omissions that you might find at info@curranbauer.org .

www.curranbauer.org

Citation Information

To cite R in publications, use:

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

To cite the lavaan package in publications, use:

Rosseel, Y. (2012). lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software, 48*, 1-36. URL http://www.jstatsoft.org/v48/i02/.

To cite this PDF, use:

Bauer, D.J. & Curran, P.J. (2019). *Structural equation modeling: R demonstration notes* (*Version 2019.3)*. Curran-Bauer Analytics, Durham: NC. URL https://curranbauer.org/sem-r-notes-2019-3/

Table of Contents

Chapter 5: Structural Equation Models with Latent Variables

Chapter 6: Multiple Groups Models

Chapter 7: SEM with Categorical Indicators

Chapter 8: Latent Growth Curve Analysis

Appendix A: Generating Path Diagrams in R

Preface: Instructions for Downloading R, RStudio, and R Packages

Downloading R

One of the nice things about R is that it is free to download and use. Just go to

`https://www.r-project.org/`

and click the "download R" link. Then run the installer.

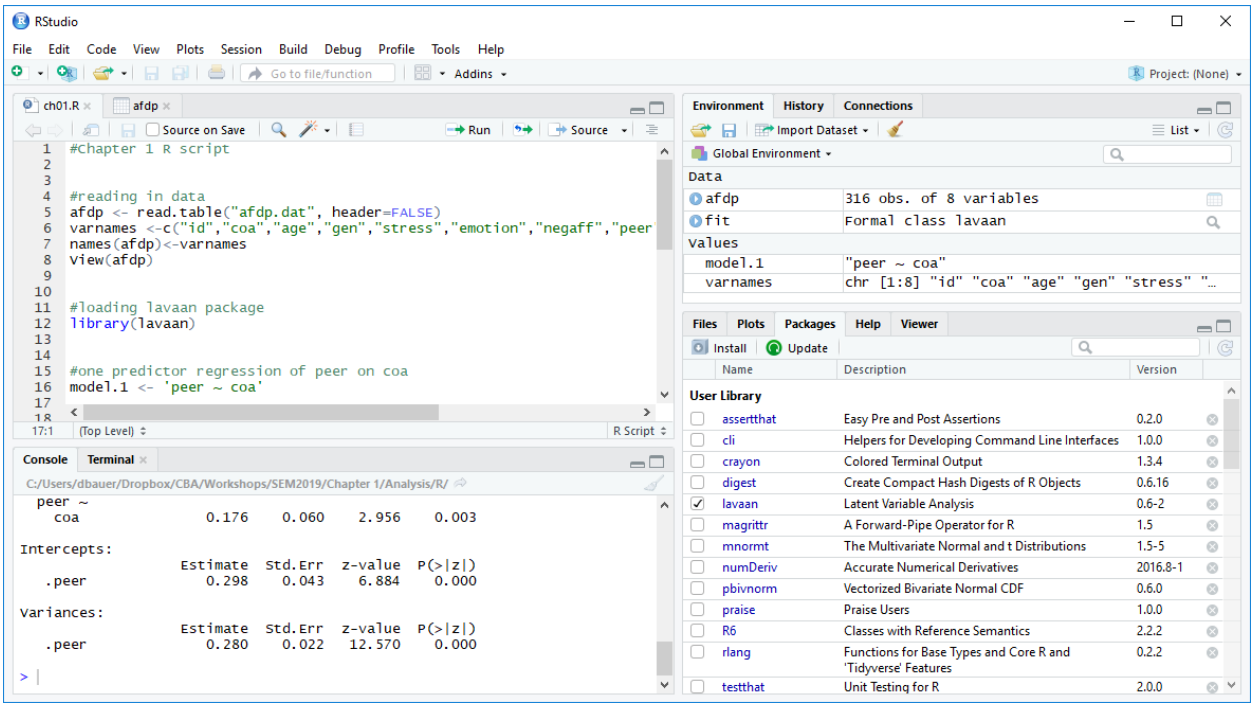
Downloading and Using the RStudio Interface

Another free program, RStudio, provides a convenient interface for using R. The installer for RStudio can be found at

`https://www.rstudio.com/products/rstudio/download/`

Select the RStudio Desktop / Open Source License for download and then run the installer.

The default layout for RStudio is shown below



The top left window shows an R script file. Saving R script in a file enables you to share code and re-run analyses quickly. You can highlight individual lines or blocks of lines and click "Run" to submit them to R and obtain the results.

The bottom left window shows the output and also displays a command prompt (the >). When you want to execute a function quickly and don't care to save it to an R script, you can simply type it at the command prompt.

The top right window shows the objects currently in working memory. Here we have two data objects, **afdp** (the data file) and **fit** (containing model results). We also have a model syntax object called **model.1** and a vector of variable labels called **varnames**.

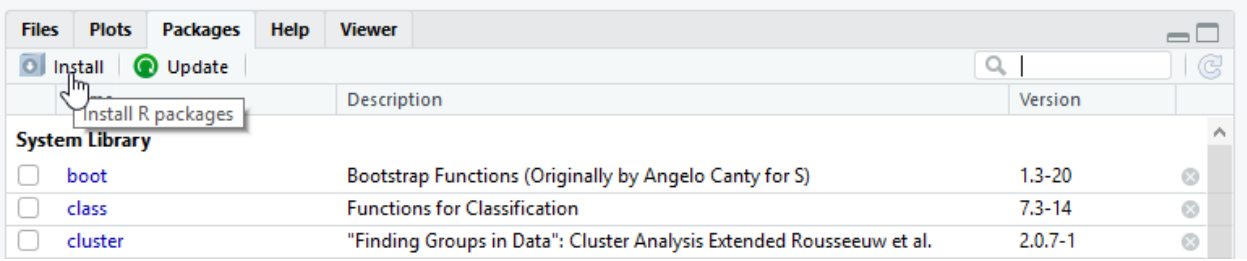
Finally, the bottom right window is useful for a variety of purposes, including viewing and installing packages, displaying plots, etc.

You can rearrange or resize these windows however you like.

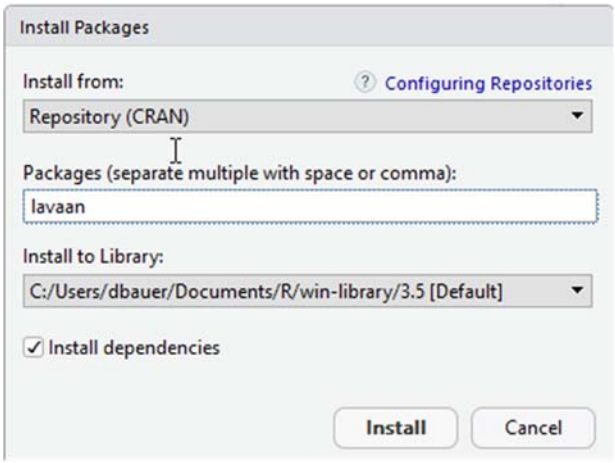
Installing and Using Lavaan and other R Packages

Although R comes with some built in functionality, much of what you can do with R comes through packages contributed by the scientific community. For this workshop, we will be primarily using the **lavaan** (**LA**tent **VA**riable **AN**alaysis) package developed by Yves Rosseel from Ghent University. Although **lavaan** is still considered to be in beta-testing (i.e., experimental, meaning there is no guarantee everything will work as it should), it is widely used and considered to generate accurate results.

There are a couple ways you can download this and other packages in R. Using RStudio, the easiest way to install a package is to click the **packages** tab (in the lower right window under the default configuration) and then click the **install** button, as shown



This will bring up the Install Packages dialogue box shown below. Just type “lavaan” on the Packages line, as shown, then click “Install”.



Alternatively, you can type the following at the command prompt:

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

Regardless of how you install **lavaan**, to actually use the package, you will need to also run the line

```
> library(lavaan)
```

The same steps are used for installing any other package in R (with the exception that some R packages are not provided through the CRAN repository accessed with these commands, in which case they must be downloaded directly from the developer).

Additional packages used in various demonstrations presented in these notes include **plyr**, **semTools**, and **psych**.

If you have questions about **lavaan**, you may wish to consult the discussion group. Go to **`https://groups.google.com/d/forum/lavaan/`** and join the group. Then you can email questions to **`lavaan@googlegroups.com`**.

Chapter 1
Introduction, Background, and Multiple Regression

Multiple Regression Analysis of Deviant Peer Affiliations

The data for this demonstration were provided by Dr. Chassin from the Adolescent and Family Development Project, housed at Arizona State University. ***Note that these data were generously provided for strictly pedagogical purposes and should not be used for any other purposes beyond this workshop.*** The sample includes 316 adolescents, between 10-16 years of age. The study was designed to assess the association between parental alcoholism and adolescent substance use and psychopathology. The data are in the text file **`afdp.dat`** and the accompanying R script is in the **`ch01.R`** file. The variables in the data set that we will use are

`peer`	adolescent report on peer substance use and peer tolerance of use
`coa`	parent report of alcoholism diagnosis where 0=non-alcoholic and 1=alcoholic
`gen`	gender where 0=girl and 1=boy
`age`	age measured in years at assessment
`stress`	self report measure of uncontrollable negative life stressful events
`emotion`	self report measure of temperamental emotional expressiveness
`negaff`	self report measure of depression and anxiety

Reading in the Data

To run our analyses in R, we must first load the data. The first five rows of the **`afdp.dat`** data file (for the first five participants), look like this:

```
1 1 14 0 2.35 1.37 4.4 0.49
2 1 12 0 0.55 1.46 2.34 0
3 1 14 1 2.37 2.12 2.11 1.73
4 1 15 1 1.14 2.83 2.6 1.86
5 1 12 1 1.37 2.11 2.04 0.36
```

The data is in free text format, meaning there are not set column positions for each variable. Instead, variable values are separated only by a space. With this kind of data format, we can use the **`read.table`** function to bring the data into R:

```
afdp <- read.table("afdp.dat", header=FALSE)
varnames <-c("id","coa","age","gen","stress","emotion","negaff","peer")
names(afdp)<-varnames
View(afdp)
```

Note that because we have not specified a full file path (e.g., "C:\Users\dbauer\CBA\SEM\R\afdp.dat"), R will look for **`afdp.dat`** in the current working directory. You can see which directory this is by typing **`getwd()`** at the prompt. You can also reset the working directory by running **`setwd("<new path>")`** or, within RStudio, by clicking **`Session -> Set Working Directory`**. We usually choose to set the working directory to the source file location, meaning the folder within which we have saved our R script file.

The **header** argument of the **read.table** function is set to **FALSE** here to indicate that there are no variable names at the top of the data file. If the first row of the data file included variable names then we would instead set this to **header=TRUE**. Because there are no names for the variables, they will initially have the default names **V1**, **V2**, **V3**, etc.

The next two lines of the script assign labels that are more informative to our variables. First, we define a vector, **varnames**, to hold the variable labels. Here we use the **c** function (for "combine") to put all the labels into one vector. Then we use the **names** function to replace the default variable labels with these new, more useful labels.

Finally, the line **View(afdp)** requests that the data be presented for viewing in spreadsheet form, as shown below.

ch01.R × afdp ×

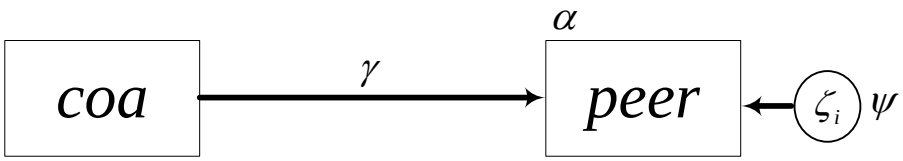
Filter

	id	coa	age	gen	stress	emotion	negaff	peer
1	1	1	14	0	2.35	1.37	4.40	0.49
2	2	1	12	0	0.55	1.46	2.34	0.00
3	3	1	14	1	2.37	2.12	2.11	1.73
4	4	1	15	1	1.14	2.83	2.60	1.86
5	5	1	12	1	1.37	2.11	2.04	0.36

We are now ready to run our analyses.

Single Predictor Regression[1]

The single-predictor regression model of COA status predicting deviant peer affiliations is shown in the diagram below:



The model is of the form $peer_i = \alpha + \gamma coa_i + \zeta_i$ where $peer_i$ is an individual's report of their level of association with deviant peers, coa_i is an adolescent's parent report of parent alcoholism status, α represents mean association with deviant peers for children of non-alcoholics (i.e., where $coa_i = 0$), and γ is the expected increase in deviant peer associations for children of alcoholics (i.e. a one unit increase in coa_i). We assume that both **coa** and **peer** contain no measurement error, that $\zeta_i \sim N(0, \psi)$ for all individuals i, and that ζ_i and coa_i are uncorrelated.

[1] The initial model building procedure presented in Chapter 1 of the lecture notes used OLS estimation in traditional regression whereas here we use ML as a way of introducing the SEM procedure in R. The results are virtually identical.

We will fit this regression model using the lavaan package, which we load with the library function as shown:

```
library(lavaan)
```

Under other circumstances, we would use a dedicated regression package to fit such a model, but our intent here is to gain familiarity with how to specify structural equation models within lavaan, starting with the very simple single-predictor regression model.

First, we need to create what's called a model syntax object. The syntax for specifying models in lavaan is quite straightforward and we will show many of the options as we proceed through the demonstrations in this book. For this simple model, we only require a single line:

```
model.1 <- 'peer ~ coa'
```

Here, we have defined the model syntax object **model.1**, in which we have specified that **peer** is predicted by **coa** using the regression operator **~**. That is, the line **peer ~ coa** signifies that **peer** is regressed on **coa**. In general, whatever variable is to the left of **~** is the dependent variable and whatever variable(s) are to the right of **~** are the predictor(s).

We now fit the model using the **sem** function, placing the results into the data object **fit**. Note that we include the **meanstructure** argument with the **sem** function so that we will obtain an estimate of the regression intercept. Last, we use the **summary** function here to display the primary output.

```
fit <- sem(model.1, data=afdp, meanstructure=TRUE)
summary(fit)
```

We now consider the output produced by the **summary** function. First, we obtain some basic information about the model estimation:

```
lavaan 0.6-2 ended normally after 11 iterations

  Optimization method                           NLMINB
  Number of free parameters                          3

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                       0.000
  Degrees of freedom                                 0
  Minimum Function Value               0.0000000000000
```

This shows us that we have 316 cases in our sample data and three parameters that we are estimating. Additionally, we are using **ML** (maximum likelihood) estimation with the **nlminb** optimizer (this is a built-in optimizer in R for finding the minimum of a function).

It is worth pausing here to note that lavaan counts parameters differently than we described in lecture. For this model, we would normally count five parameters, the mean and variance of **COA**, the intercept and slope of the regression of **Peer** on **COA**, and the residual variance of **Peer**. But lavaan does not count the mean or variance of the predictor, **COA**, as parameters of

the model. Nor does it count covariances among predictors (though there are no such covariances in the current model) as free parameters. Fortunately, lavaan also does not count these parameters when determining the number of observed moments, so the degrees of freedom for the test statistic work out regardless (e.g., here it neither counts the mean and variance of COA as observed moments nor does it count them as estimated moments, leaving a net difference of zero when calculating the degrees of freedom).

This output also provides some information about model fit. Multiple regression models are just identified (i.e., every piece of information provided by the sample is 'used up' to estimate model parameters so that no degrees of freedom remain). Therefore, the model fits the data perfectly and it is not worthwhile to interpret the test statistic (this and other fit indices will be discussed in later sections).

The estimates for the model are shown next:

```
Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  peer ~
    coa               0.176    0.060    2.956    0.003

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .peer              0.298    0.043    6.884    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .peer              0.280    0.022   12.570    0.000
```

Recall that our model is

$$peer_i = \alpha + \gamma coa_i + \zeta_i$$

The output is subdivided into three parts, regressions, intercepts, and variances. In the **Regressions** section, the **peer ~ coa** coefficient is the slope parameter estimate $\hat{\gamma}$. In the **Intercepts** section, the coefficient listed for **.peer** is the intercept $\hat{\alpha}$. Finally, in the Variances section, the coefficient listed for .peer is the represents represents $\hat{\psi}$, the estimated variance of ζ_i (i.e., residual variance). The **Estimate** column lists is the ML point estimate of each parameter. The **Std.Err** column gives the standard error for each estimate (where these are computed using the expected information matrix, as noted above). Next, the **z-value** column reports the Wald z-statistic for the null hypothesis test that the parameter is significantly different from zero in the population, and the **P(>|z|)** column reports the p-value associated with this z-statistic.

Here, we see that the average non-COA has a score of .298 on **peer** and the average COA has a score that is .176 units higher than non-COAs on **peer** (.298 + .176 = .474). Both the intercept and slope are significantly different from zero. Finally, the variance in deviant peer association that is not explained by COA status is .280. This indicates that, although COA status is a significant predictor of **peer**, COA status does not fully account for affiliation with peers.

Because **peer** is not on an intrinsically meaningful scale, we cannot easily interpret the differences among the regression parameters or residual variances. To better interpret these results, we can request standardized estimates. Within lavaan, several methods of standardization are available within the **summary** function. The first type, obtained by including **standardized=TRUE**, is the typical fully standardized solution. In this case, both the independent and dependent variables are standardized to have a variance of 1 (note that lavaan does not, however, also make the means 0, as one might expect). We can see this by running the following line from the R script:

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

The relevant results are shown here:

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  peer ~
    coa               0.176    0.060    2.956    0.003    0.176    0.164

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .peer              0.298    0.043    6.884    0.000    0.298    0.554

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .peer              0.280    0.022   12.570    0.000    0.280    0.973
```

The **Std.lv** column refers to a solution in which only the latent variables are standardized and not the observed variables. Since the current model contains no latent variables, this is of little use here. The **Std.all** column contains the fully standardized estimates of interest, in which all variables (observed and latent) are standardized to have a variance of one. Thus, we say that a one standard deviation increase in COA status is associated with a .164 standard deviation increase in deviant peer associations.

It may be more useful to retain the original scaling of **COA** while standardizing **peer**. If we include **std.nox=TRUE** in the **summary** function, as shown below, we will obtain estimates in which the latent variables and endogenous observed variables are standardized but the exogenous observed variables are left in their raw scale. These are commonly known as partially standardized estimates.

```
summary(fit, std.nox=TRUE)
```

The relevant output is shown here:

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
  peer ~
    coa               0.176    0.060    2.956    0.003    0.176    0.329

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .peer              0.298    0.043    6.884    0.000    0.298    0.554

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .peer              0.280    0.022   12.570    0.000    0.280    0.973
```

We can interpret these results by saying that the average COA affiliates with deviant peers about .329 standard deviations more than the average non-COA.

Finally, we can also calculate how much variance is explained in peer by coa by requesting that lavaan output the r-squared statistic:

```
summary(fit, rsquare=TRUE)
```

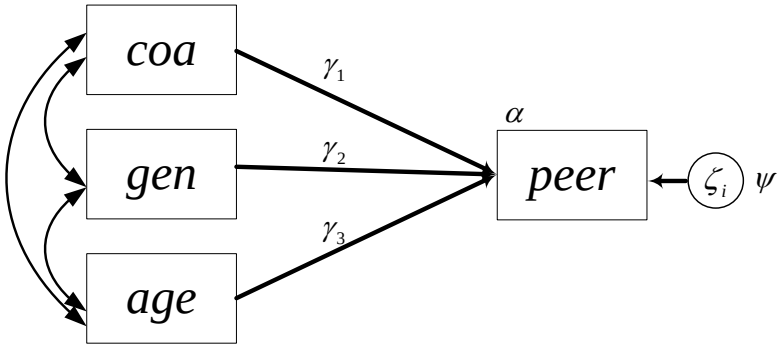
We now obtain a new section of output, shown here:

```
R-Square:
                   Estimate
    peer              0.027
```

R-Square is the estimated proportion of variance in **peer** that is accounted for by the models (i.e., **COA**, the only predictor included in this model). We have explained less than 3% of the total variance in **peer** with the **COA** predictor. Note, however, that measured variable regression models assume that no measurement error is present. If measurement error is present, the estimated relationship among the variables in the model may be attenuated and the R^2 value will also be underestimated.

Multiple Regression Model with COA Status, Gender, and Age as Predictors

We now expand the model to include gender and age as additional predictors of deviant peer relations as shown in the diagram below. All of the predictors are implicitly allowed to covary with one another.



The model is of the form $peer_i = \alpha + \gamma_1 coa_i + \gamma_2 gen_i + \gamma_3 age_i + \zeta_i$. Each γ is interpreted as the effect of the associated predictor on **peer**, holding the other predictors constant. Unless all predictors are uncorrelated with one another, these estimates will change depending on which other predictors are included in the model. α represents the expected value of **peer** when all predictors are equal to zero.

We assume that none of the variables in our model contain measurement error, that $\zeta_i \sim N(0, \psi)$ for all individuals *i,* that the error variance is constant for all predictors, and that ζ_i is uncorrelated with all of the predictors in the model.

The R script for fitting this model is shown below:

```
model.2 <- 'peer ~ coa + gen + age'

fit <- sem(model.2, data=afdp, meanstructure=TRUE)
summary(fit, std.nox=TRUE, rsquare=TRUE)
```

Here, we have defined a new model syntax object, **model.2**, in which we have specified that **peer** is regression on **coa**, **gen**, and **age**. Note that, again, we use the regression operator **~** to indicate the regression. Now that our model includes multiple predictors, we use the **+** operator in between each predictor.

The results output by the **summary** function (including both partially standardized estimates and r-square) are shown here:

```
lavaan 0.6-2 ended normally after 18 iterations

  Optimization method                           NLMINB
  Number of free parameters                          5

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                       0.000
  Degrees of freedom                                 0
  Minimum Function Value               0.0000000000000

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
  peer ~
    coa               0.210    0.054    3.858    0.000    0.210    0.391
    gen              -0.048    0.055   -0.877    0.381   -0.048   -0.089
    age               0.151    0.019    7.993    0.000    0.151    0.281
```

```
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .peer              -1.610    0.250   -6.453    0.000   -1.610   -2.999

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .peer               0.232    0.018   12.570    0.000    0.232    0.804

R-Square:
                   Estimate
    peer               0.196
```

We see that **gen** is not a significant predictor of **peer** when **COA** and **age** are also included in the model (p = .381). However, **age** is significantly related with **peer** such that older adolescents are more likely to associate with deviant peers. We estimate that a one-year increase in **age** is associated with a .151-increase in deviant peer association ratings. **COA** remains a significant predictor of **peer**, and the regression parameter estimate has not changed much from the single-predictor model to the multiple-predictor model relative to its standard error (i.e., it has increased from .18 to .21). The stability of this coefficient reflects the fact that **COA** is not highly correlated with **gen** or **age**.

The regression coefficients can be interpreted as follows. The slope of **peer** on **COA** is the average effect of being a child of an alcoholic, holding age and gender constant. The slope of **age** is the average effect of **age** on **peer**, holding COA status and gender constant. The intercept is less informative in this model because **age** has not been centered. It now represents the average deviant peer association score for female, non-COAs who are zero years old. This is obviously outside of the range of our data.

To better interpret these results, and to get a sense for the relative contribution of each predictor, we requested the partially standardized solution. We prefer this method of standardization for this example because all three predictors have natural scales. As can be seen in the **Std.nox** column of output, after controlling for **age** and **gender**, COAs affiliate with deviant peers about .391 standard deviation units more than non-COAs. Each additional year of age is associated with a .281 standard deviation increase in self-reported affiliation with deviant peers.

From the R-square output, we can see that the multiple predictor regression model explains more of the variance in peer than the single predictor model. Age and gender account for an additional 17% of the variance in **peer**, over and above COA status. Still, approximately 80% of the variance in **peer** remains unexplained by the variables in our model.

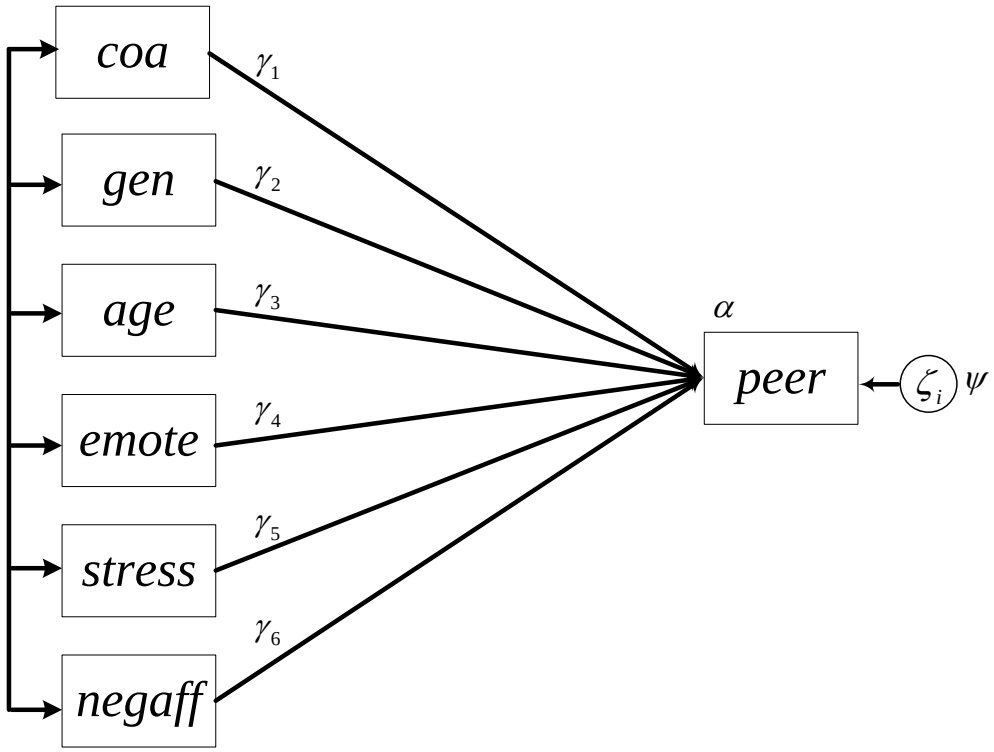
Multiple Regression Model with COA Status, Gender, Age, Stress, Emotion, and Negative Affect as Predictors

To see if we can explain more of the variance in deviant peer associations, we expand the model to include **stress**, **emotion**, and **negaff**, while retaining **COA**, **gen**, and **age**.

The model is now

$$peer_i = \alpha + \gamma_1 coa_i + \gamma_2 gen_i + \gamma_3 age_i + \gamma_4 emote_i + \gamma_5 stress_i + \gamma_6 negaff_i + \zeta_i$$

with the same assumptions as before. Although **gen** was not a significant predictor of deviant peer relations in the last model, we do not exclude it from this model because it may covary with the new predictors in such a way that it is important to include it as a control variable in the model. The full model is shown in the diagram.



The R script for fitting this model is shown below:

```
model.3 <- 'peer ~ coa + gen + age + emotion + stress + negaff'

fit <- sem(model.3, data=afdp, meanstructure=TRUE)
summary(fit, standardized=TRUE, rsquare=TRUE)
summary(fit, std.nox=TRUE, rsquare=TRUE)
```

As before, we simply defined a new model syntax object, **model.3**, which includes the new predictors of **peer**. In this case we requested both the fully standardized and partially standardized solutions in two calls of the **summary** function. For **coa**, **gen**, and **age**, the partially standardized coefficients are most interpretable. In contrast, **emotion**, **stress**, and

negaff do not have intrinsically meaningful scales, so the fully standardized estimates are better. Thus, we consider both. Additionally, by examining the fully standardized estimates across the full set of predictors we can gauge the relative contributions of the predictors, despite the fact that they were originally on quite different scales.

Here, we show the output from the first summary function requesting fully standardized effects:

```
lavaan 0.6-2 ended normally after 22 iterations

  Optimization method                           NLMINB
  Number of free parameters                          8

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                       0.000
  Degrees of freedom                                 0

Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  peer ~
    coa               0.137    0.055    2.492    0.013    0.137    0.127
    gen              -0.027    0.052   -0.509    0.611   -0.027   -0.025
    age               0.140    0.018    7.660    0.000    0.140    0.378
    emotion           0.030    0.058    0.520    0.603    0.030    0.028
    stress            0.111    0.044    2.546    0.011    0.111    0.140
    negaff            0.109    0.030    3.660    0.000    0.109    0.195

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .peer             -1.934    0.267   -7.233    0.000   -1.934   -3.602

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .peer              0.210    0.017   12.570    0.000    0.210    0.728

R-Square:
                   Estimate
    peer              0.272
```

We see that **gen** is still not a significant predictor of **peer** when other predictors are included in the model (p = .611), but that **COA** and **age** remain statistically significant even though the value of their regression coefficients have changed. Importantly, the effect of **COA** on **peer** is not as strong after controlling for **emotion**, **stress**, and **negaff**, indicating that these

variables are somewhat related to one another. These new variables could potentially mediate the relationship between **COA** and **peer**; we will later explore this possibility with a path analysis model. Of the new variables, it appears that stressful life events and negative affect are significant predictors of association with deviant peers, but self-reported emotional expressiveness is not significantly related to deviant peer association, after controlling for **age**, **gen**, **stress**, **negaff**, and **COA**.

Examining the standardized estimates in the **Std.all** column, we see that **age** is the strongest relative predictor of **peer**, followed by **negaff**, **stress**, **COA**, **emotion**, and then **gen**. However, determining the "strongest" predictors is always a tricky endeavor, particularly given that **gen** and **COA** are binary predictors for which fully standardized effects are less useful.

We can examine the results from the second **summary** call to see the partially standardized estimates.

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
  peer ~
    coa               0.137    0.055    2.492    0.013    0.137    0.255
    gen              -0.027    0.052   -0.509    0.611   -0.027   -0.049
    age               0.140    0.018    7.660    0.000    0.140    0.262
    emotion           0.030    0.058    0.520    0.603    0.030    0.056
    stress            0.111    0.044    2.546    0.011    0.111    0.206
    negaff            0.109    0.030    3.660    0.000    0.109    0.204

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .peer             -1.934    0.267   -7.233    0.000   -1.934   -3.602

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .peer              0.210    0.017   12.570    0.000    0.210    0.728

R-Square:
                   Estimate
    peer              0.272
```

Finally, examining the R-square output, we see that the final model explains about 27% of the total variance in **peer**.

Chapter 2
Path Analysis: Part I

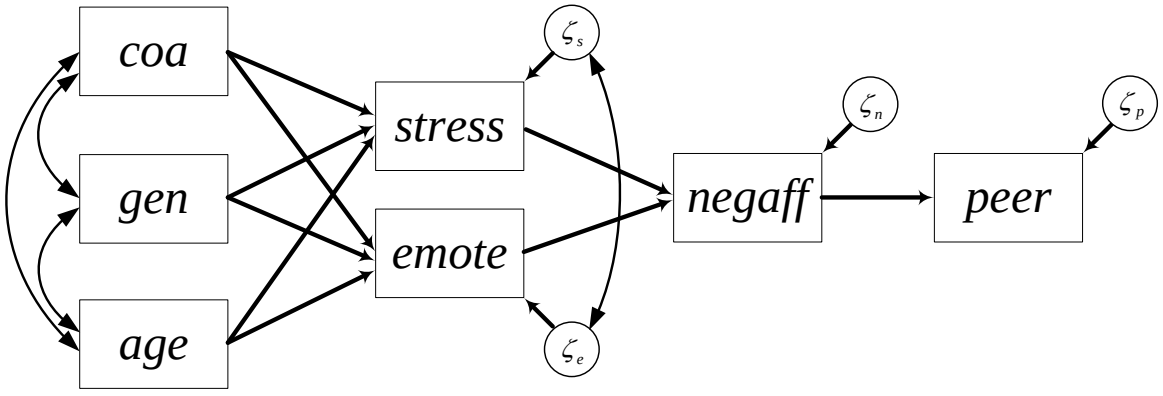
Path Analysis with Deviant Peer Affiliation Data

The data for this demonstration were provided by Dr. Laurie Chassin from the Adolescent and Family Development Project, housed at Arizona State University. ***Note that these data were generously provided for strictly pedagogical purposes and should not be used for any other purposes beyond this workshop.*** The sample includes 316 adolescents, between 10-16 years of age. The study was designed to assess the association between parental alcoholism and adolescent substance use and psychopathology. The data are in the text file **`afdp.dat`** and the accompanying R script is in the **`ch02.R`** file. The variables in the data set that we will use are

`peer`	adolescent report on peer substance use and peer tolerance of use
`coa`	parent report of alcoholism diagnosis where 0=non-alcoholic and 1=alcoholic
`gen`	gender where 0=girl and 1=boy
`age`	age measured in years at assessment
`stress`	self report measure of uncontrollable negative life stressful events
`emotion`	self report measure of temperamental emotional expressiveness
`negaff`	self report measure of depression and anxiety

Path Analysis of Theoretical Peer Affiliation Model

Theory dictates that alcoholic parents increase the number of stressful life events that their children experience, leading to an increase in child negative affect. Further, children of alcoholics are hypothesized to have higher levels of emotionality, leading to more negative affect. Negative affect is thought to be related to higher rates of affiliation with deviant peers. Stressful life events and emotionality should covary, but we hypothesize no directional relation among these variables. We allow **`age`** and **`gen`** to predict **`stress`** and **`emotion`**, and we allow all exogenous characteristics (**`coa`**, **`gen`**, and **`age`**) to covary.



The model is of the form

$$\begin{pmatrix} y_{stress_i} \\ y_{emote_i} \\ y_{negaff_i} \\ y_{peer_i} \end{pmatrix} = \begin{pmatrix} \alpha_{1i} \\ \alpha_{2i} \\ \alpha_{3i} \\ \alpha_{4i} \end{pmatrix} + \begin{pmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} coa_i \\ gen_i \\ age_i \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \beta_{31} & \beta_{32} & 0 & 0 \\ 0 & 0 & \beta_{43} & 0 \end{pmatrix} \begin{pmatrix} stress_i \\ emote_i \\ negaff_i \\ peer_i \end{pmatrix} + \begin{pmatrix} \zeta_{1i} \\ \zeta_{2i} \\ \zeta_{3i} \\ \zeta_{4i} \end{pmatrix}$$ where

$$COV(\zeta_i) = \begin{pmatrix} \psi_{11} & & & \\ \psi_{21} & \psi_{22} & & \\ 0 & 0 & \psi_{33} & \\ 0 & 0 & 0 & \psi_{44} \end{pmatrix}$$

The R script that fits this model with the lavaan package is shown below (note that this assumes the data file is in the same directory as the R script file, i.e., the source directory):

```
#reading in data
afdp <- read.table("afdp.dat", header=FALSE)
varnames <-c("id","coa","age","gen","stress","emotion","negaff","peer")
names(afdp)<-varnames

#loading lavaan package
library(lavaan)

#fitting path analysis model
pathmodel.1 <- '
              #regressions
              stress ~ coa + gen + age
              emotion ~ coa + gen + age
              negaff ~ stress + emotion
              peer ~ negaff

              #covariances
              stress ~~ emotion
              '
fit <- sem(pathmodel.1, data=afdp, meanstructure=TRUE)
summary(fit, standardized=TRUE, rsquare=TRUE)
summary(fit, std.nox=TRUE, rsquare=TRUE)

fitted(fit)
resid(fit, type="raw")
resid(fit, type="normalized")
```

We have seen most of this code in Chapter 1, so here we will focus only on what's new in the definition of the model syntax object. Note that some lines in this script, those that are preceded with a **#** character, are comments and are not part of the actual syntax.

In Chapter 1, **`peer`** was the only dependent variable and was regressed on all of the other variables in the model. Here, we have multiple endogenous variables and each has its own

regression equation, as specified in the section of the syntax beginning with the **#regressions** comment. Thus, we indicate that there are direct effects of **coa**, **gen**, and **age** on **stress** with the line **stress ~ coa + gen + age**. In each subsequent line, we indicate the direct effects of the predictors for each endogenous variable in the model, ending with **peer** being regressed on **negaff**.

As a default, lavaan allows all exogenous variables to covary but it fixes the residual covariances among endogenous variables to zero. The residual terms of **stress** and **emotion** are freed to covary within the section of the code marked **#covariances**. To designate a covariance (or variance) we use the **~~** operator. Thus, **stress ~~ emotion** tells lavaan to include a residual covariance for **stress** and **emotion**.

In the two **summary** function calls we requested standardized estimates and partially standardized estimates, respectively, to aid interpretation of the results. The final three lines request the model-implied covariance matrix (**fitted(fit)**) and the raw and standardized residual matrices (**resid(fit, type="raw")** and **resid(fit, type="normalized")**, respectively). These residuals represent the differences between the observed covariance matrix and the matrix that is implied by the structure of the model.

Typically, we would evaluate model fit prior to interpreting parameter estimates. For pedagogical purposes, however, we will put aside a discussion of model fit until Chapter 3 and move directly to parameter estimates for our model. Here we see the raw and fully standardized estimates produced by the first **summary** function call:

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  stress ~
    coa               0.451    0.072    6.223    0.000    0.451    0.331
    gen              -0.016    0.073   -0.215    0.830   -0.016   -0.011
    age               0.002    0.025    0.078    0.938    0.002    0.004
  emotion ~
    coa               0.110    0.056    1.963    0.050    0.110    0.110
    gen              -0.048    0.056   -0.843    0.399   -0.048   -0.047
    age              -0.027    0.019   -1.374    0.170   -0.027   -0.077
  negaff ~
    stress            0.246    0.078    3.134    0.002    0.246    0.175
    emotion           0.553    0.106    5.206    0.000    0.553    0.290
  peer ~
    negaff            0.176    0.030    5.892    0.000    0.176    0.315
Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .stress ~~
   .emotion           0.112    0.019    5.896    0.000    0.112    0.352
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .stress            0.687    0.333    2.066    0.039    0.687    1.010
   .emotion           2.341    0.258    9.088    0.000    2.341    4.663
   .negaff            1.527    0.207    7.373    0.000    1.527    1.594
   .peer             -0.118    0.091   -1.298    0.194   -0.118   -0.220
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .stress            0.412    0.033   12.570    0.000    0.412    0.890
   .emotion           0.247    0.020   12.570    0.000    0.247    0.979
   .negaff            0.778    0.062   12.570    0.000    0.778    0.848
   .peer              0.260    0.021   12.570    0.000    0.260    0.901

R-Square:
                   Estimate
    stress            0.110
    emotion           0.021
    negaff            0.152
    peer              0.099
```

The second summary function call reproduces these same raw parameter estimates but now computes the partially standardized estimates:

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
  stress ~
    coa               0.451    0.072    6.223    0.000    0.451    0.663
    gen              -0.016    0.073   -0.215    0.830   -0.016   -0.023
    age               0.002    0.025    0.078    0.938    0.002    0.003
  emotion ~
    coa               0.110    0.056    1.963    0.050    0.110    0.219
    gen              -0.048    0.056   -0.843    0.399   -0.048   -0.095
    age              -0.027    0.019   -1.374    0.170   -0.027   -0.053
  negaff ~
    stress            0.246    0.078    3.134    0.002    0.246    0.175
    emotion           0.553    0.106    5.206    0.000    0.553    0.290
  peer ~
    negaff            0.176    0.030    5.892    0.000    0.176    0.315

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
 .stress ~~
   .emotion           0.112    0.019    5.896    0.000    0.112    0.352

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .stress            0.687    0.333    2.066    0.039    0.687    1.010
   .emotion           2.341    0.258    9.088    0.000    2.341    4.663
   .negaff            1.527    0.207    7.373    0.000    1.527    1.594
   .peer             -0.118    0.091   -1.298    0.194   -0.118   -0.220

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
   .stress            0.412    0.033   12.570    0.000    0.412    0.890
   .emotion           0.247    0.020   12.570    0.000    0.247    0.979
   .negaff            0.778    0.062   12.570    0.000    0.778    0.848
   .peer              0.260    0.021   12.570    0.000    0.260    0.901
```

```
R-Square:
                  Estimate
    stress           0.110
    emotion          0.021
    negaff           0.152
    peer             0.099
```

We see that the hypothesized pathways to deviant peer affiliation do contain statistically significant components. To aid in interpretation, standardized values are included in the model path diagram, shown below. We report fully standardized effects (from **Std.all** column generated by the first **summary** function call) except for **coa**, **gen**, and **age**, for which we report partially standardized effects (from **Std.nox** column generated by the second **summary** function call). Recall that only the outcome variables are standardized in computing the partially standardized effects, which makes them particularly useful for examining the effects of coding variables (e.g., **coa** and **gen**) or predictors with natural metrics (e.g., **age**).

These partially standardized parameter estimates represent the expected change in standard deviation units in *y* given a one unit increase in *x*. By comparison, the fully standardized estimates are computed by standardizing both the predictors and the outcome variables so that parameter estimates represent the expected change in standard deviation units in *y* given a one standard deviation increase in *x*. These are most useful for interpreting effects when neither predictors nor outcomes have natural scales and when gauging relative effect sizes across predictors on different scales. For covariance parameters, fully standardized estimates can be interpreted as correlations.

From the **R-Square** section of output, we can also see that only a modest amount of the total variance in any of these variables has been explained by the model.

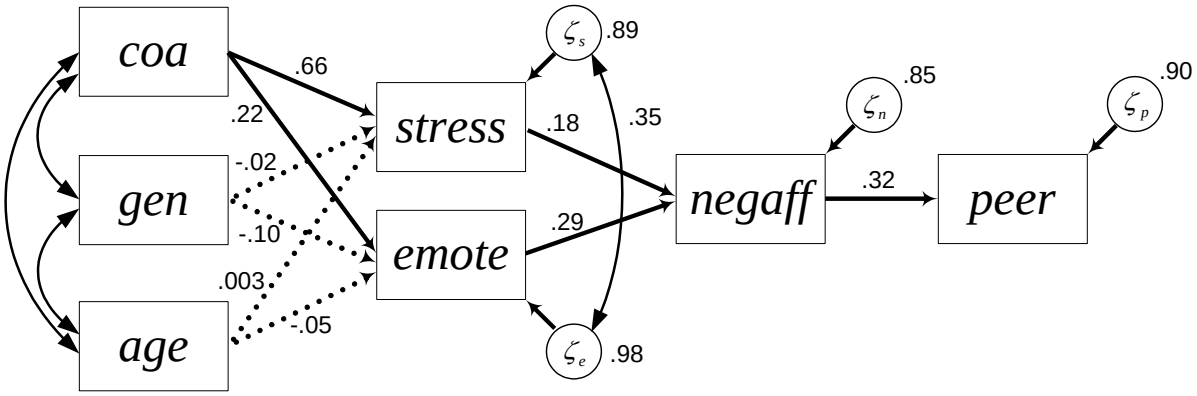
Finally, the model-implied covariance matrix and the raw and standardized residuals between the observed and model-implied covariance matrices are:

```
> fitted(fit)
$`cov`
        stress emotin negaff peer   coa    gen    age
stress   0.463
emotion  0.125  0.252
negaff   0.183  0.170  0.917
peer     0.032  0.030  0.162  0.288
coa      0.112  0.029  0.044  0.008  0.249
gen     -0.003 -0.010 -0.006 -0.001  0.002  0.249
age     -0.019 -0.059 -0.037 -0.007 -0.055 -0.070  2.095

$mean
 stress emotion  negaff    peer     coa     gen     age
  0.941   2.034   2.883   0.390   0.525   0.538  12.718
```

```
> resid(fit, type="raw")
$`type`
[1] "raw"

$cov
        stress emotin negaff peer   coa    gen    age
stress   0.000
emotion  0.000  0.000
negaff   0.000  0.000  0.000
peer     0.055  0.006  0.000  0.000
coa      0.000  0.000 -0.004  0.036  0.000
gen      0.000  0.000 -0.042 -0.021  0.000  0.000
age      0.000  0.000  0.247  0.314  0.000  0.000  0.000

$mean
 stress emotion  negaff    peer     coa     gen     age
      0       0       0       0       0       0       0

> resid(fit, type="normalized")
$`type`
[1] "normalized"

$cov
        stress emotin negaff peer   coa    gen    age
stress   0.000
emotion  0.000  0.000
negaff   0.000  0.000  0.000
peer     2.626  0.394  0.000  0.000
coa      0.000  0.000 -0.159  2.375  0.000
gen      0.000  0.000 -1.541 -1.383  0.000  0.000
age      0.000  0.000  3.130  6.681  0.000  0.000  0.000

$mean
 stress emotion  negaff    peer     coa     gen     age
      0       0       0       0       0       0       0
```

The normalized residuals are rescaled to follow a standard normal distribution and values exceeding plus or minus two are often taken as potentially meaningful in magnitude. The normalized residuals for this model suggest that the hypothesized structure is doing a poor job in reproducing the observed covariances between several variables, most notably between *age* and *peer*, between *age* and *negaff*, and between *stress* and *peer*. We will explore methods for more formally assessing overall model fit in the next chapter.

Finally, below is a path diagram denoting all estimated parameters. Significant effects are denoted with solid lines (p<.05) and non-significant with dashed lines. Partially standardized effects are presented for COA, gender, and age and fully standardized are presented for all other predictors.



Chapter 3
Path Analysis: Part II

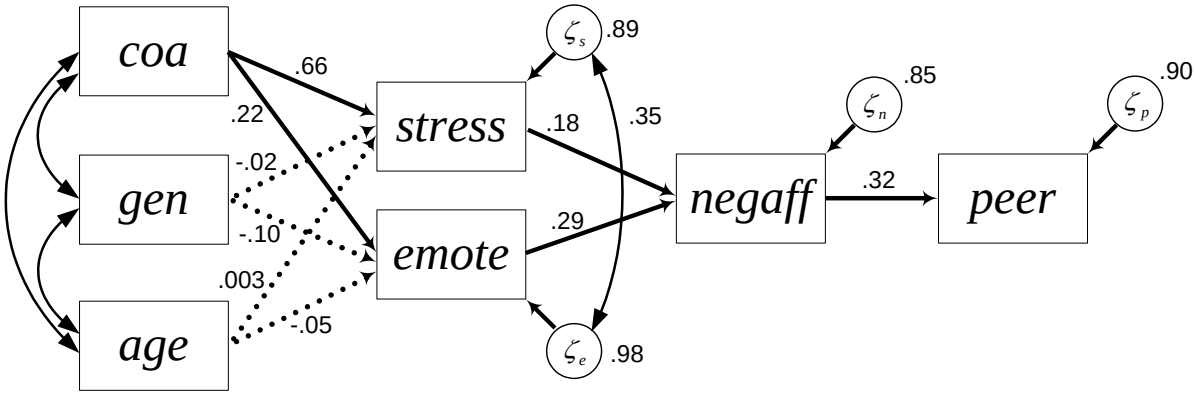
Path Analysis with Deviant Peer Affiliation Data

The data for this demonstration were provided by Dr. Laurie Chassin from the Adolescent and Family Development Project, housed at Arizona State University. ***Note that these data were generously provided for strictly pedagogical purposes and should not be used for any other purposes beyond this workshop.*** The sample includes 316 adolescents, between 10-16 years of age. The study was designed to assess the association between parental alcoholism and adolescent substance use and psychopathology. The data are in the text file **`afdp.dat`** and the accompanying R script is in the **`ch03.R`** file. The variables in the data set that we will use are

`peer`	adolescent report on peer substance use and peer tolerance of use
`coa`	parent report of alcoholism diagnosis where 0=non-alcoholic and 1=alcoholic
`gen`	gender where 0=girl and 1=boy
`age`	age measured in years at assessment
`stress`	self report measure of uncontrollable negative life stressful events
`emotion`	self report measure of temperamental emotional expressiveness
`negaff`	self report measure of depression and anxiety

Theoretical Model

We tested our hypothesized theoretical model of pathways to deviant peer affiliation for COAs and non-COAs. The original model is illustrated below with standardized parameter estimates overlaid on the path diagram (regression paths from **`coa`**, **`gen`**, **`age`** are partially standardized; all other estimates are fully standardized):



The R script for fitting this model was shown in Chapter 2 and is included here again for reference.

```
pathmodel.1 <-'
              #regressions
              stress ~ coa + gen + age
              emotion ~ coa + gen + age
              negaff ~ stress + emotion
              peer ~ negaff

              #covariances
              stress ~~ emotion
              '
fit.1 <- sem(pathmodel.1, data=afdp, meanstructure=TRUE)
```

The only modification to the script is that the object previously named **fit** has been renamed **fit.1**, as we will be comparing fit across multiple models in this chapter.

We obtain measures of fit, appropriately enough, by including **fit.measures=TRUE** as an argument to the summary function, as shown here:

```
summary(fit.1, fit.measures=TRUE)
```

Although the output includes the parameter estimates, we will focus here only on output related to model fit. Some of this information is already included in the default output header:

```
lavaan 0.6-2 ended normally after 43 iterations

  Optimization method                           NLMINB
  Number of free parameters                         18

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                      81.173
  Degrees of freedom                                 8
  P-value (Chi-square)                           0.000
```

Lavaan lists the number of free parameters as 18, but recall that it does not count the means, variances, or covariances of the exogenous predictors (**coa**, **gen**, and **age**) as estimated parameters. The number of free parameters reported by lavaan for this model thus does not equal what we computed by the *t*-rule (27 free parameters, including 3 means, 3 variances, and 3 covariances for **coa**, **gen**, and **age**; 16 + 9 = 27). The *t*-rule still works out to 8 degrees of freedom for the chi-square, however, because the observed means, variances, and covariances of the covariates are also not counted in the number of sample means and variances *k*.

As we can see, the chi square-distributed likelihood ratio test of model fit rejects the null hypothesis that the model fits the data. However, to get a full understanding of model fit we must also consider other fit indices.

The additional output generated by **fit.measures=TRUE** is shown below

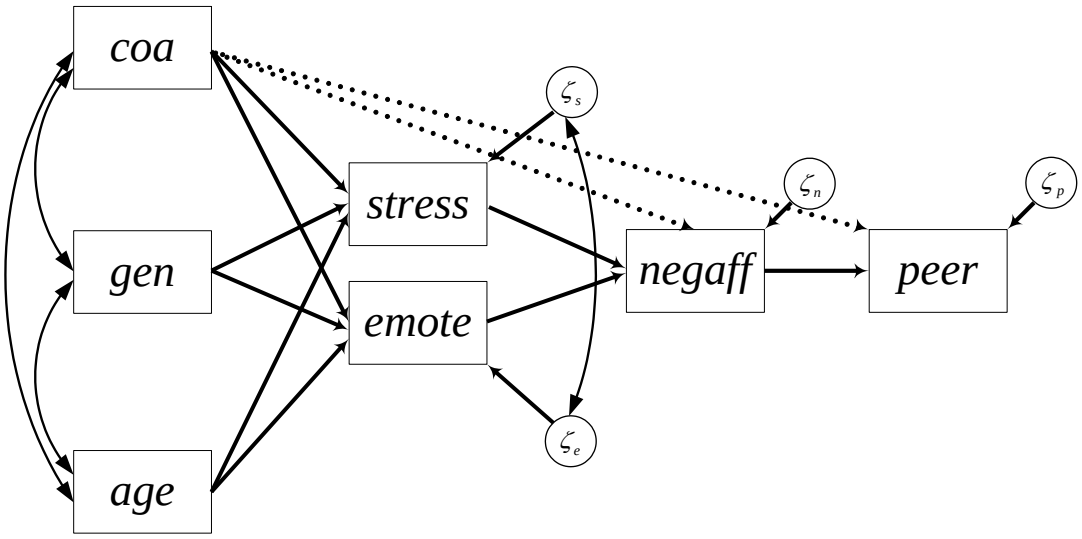
```
Model test baseline model:

  Minimum Function Test Statistic              251.027
  Degrees of freedom                                18
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.686
  Tucker-Lewis Index (TLI)                       0.293

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -1158.938
  Loglikelihood unrestricted model (H1)      -1118.351

  Number of free parameters                         18
  Akaike (AIC)                                2353.875
  Bayesian (BIC)                              2421.479
  Sample-size adjusted Bayesian (BIC)         2364.387

Root Mean Square Error of Approximation:

  RMSEA                                          0.170
  90 Percent Confidence Interval          0.138  0.205
  P-value RMSEA <= 0.05                          0.000

Standardized Root Mean Square Residual:

  SRMR                                           0.085
```

In addition to poor model chi-square, the CFI and the TLI are far lower than .9, the standard lower bound for a good-fitting model. Finally, the 90% confidence interval for the RMSEA does not even include .10 at the lower bound, indicating terrible fit. As such, we cannot interpret the obtained parameter estimates with any confidence given the severe misfit of the model.

Modification to Peer Affiliation Model: Likelihood Ratio Test

Since the theoretical model of deviant peer affiliation that was presented in Chapter 2 did not fit the data well, we will consider model modifications to improve our representation of the data. First, theory might suggest that it is an excessively severe restriction to require that the influence of parental alcoholism be conveyed entirely by the mediators. Thus we can allow **coa** to directly predict **negaff** and **peer**, over and above its indirect relationship with these variables via **stress** and **emotion**.

The new paths are illustrated with dotted lines in the path diagram below.



The model is now of the form:

$$\begin{pmatrix} y_{stress_i} \\ y_{emote_i} \\ y_{negaff_i} \\ y_{peer_i} \end{pmatrix} = \begin{pmatrix} \alpha_{1i} \\ \alpha_{2i} \\ \alpha_{3i} \\ \alpha_{4i} \end{pmatrix} + \begin{pmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & 0 & 0 \\ \gamma_{41} & 0 & 0 \end{pmatrix} \begin{pmatrix} coa_i \\ gen_i \\ age_i \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \beta_{31} & \beta_{32} & 0 & 0 \\ 0 & 0 & \beta_{43} & 0 \end{pmatrix} \begin{pmatrix} stress_i \\ emote_i \\ negaff_i \\ peer_i \end{pmatrix} + \begin{pmatrix} \zeta_{1i} \\ \zeta_{2i} \\ \zeta_{3i} \\ \zeta_{4i} \end{pmatrix}$$

where

$$COV(\zeta_i) = \begin{pmatrix} \psi_{11} & & & \\ \psi_{21} & \psi_{22} & & \\ 0 & 0 & \psi_{33} & \\ 0 & 0 & 0 & \psi_{44} \end{pmatrix}$$

The R script for this model is shown below:

```
pathmodel.2 <- '
               #regressions
               stress ~ coa + gen + age
               emotion ~ coa + gen + age
               negaff ~ stress + emotion + coa
               peer ~ negaff + coa

               #covariances
               stress ~~ emotion
               '
fit.2 <- sem(pathmodel.2, data=afdp, meanstructure=TRUE)
summary(fit.2, fit.measures=TRUE)
```

The only difference from the **pathmodel.1** model syntax object is that **coa** is included as a predictor in the lines for **peer** and **negaff**.

Let us now turn to the model fit to determine whether freeing these two regression parameters has led to a significant improvement in model fit.

```
lavaan 0.6-2 ended normally after 47 iterations

  Optimization method                           NLMINB
  Number of free parameters                         20

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                      74.321
  Degrees of freedom                                 6
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              251.027
  Degrees of freedom                                18
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.707
  Tucker-Lewis Index (TLI)                       0.120

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -1155.511
  Loglikelihood unrestricted model (H1)      -1118.351

  Number of free parameters                         20
  Akaike (AIC)                                2351.023
  Bayesian (BIC)                              2426.138
  Sample-size adjusted Bayesian (BIC)         2362.703

Root Mean Square Error of Approximation:

  RMSEA                                          0.190
  90 Percent Confidence Interval          0.153  0.230
  P-value RMSEA <= 0.05                          0.000

Standardized Root Mean Square Residual:

  SRMR                                           0.081
```

The likelihood for the original model (Model A) was $T_A = 81.17$ on 8 degrees of freedom. We have used two addition model parameters for the modified model (Model B), and the likelihood test statistic for this model is 74.32. These models are nested, so we can conduct a Likelihood Ratio Test (LRT) to determine whether Model B fits significantly better than Model A. Lavaan nicely provides a function for conducting the LRT, **lavTestLRT**, in which you simply provide the fit objects from Models B and A (in that order, where A is more restricted than B), as shown here for the present example:

```
lavTestLRT(fit.2, fit.1)
```

This generates the output

```
Chi Square Difference Test

      Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.2  6 2351.0 2426.1 74.321
fit.1  8 2353.9 2421.5 81.173     6.8522       2    0.03251 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Thus we have

$$T_\Delta = T_A - T_B = 81.17 - 74.32 = 6.85$$
$$df_\Delta = df_A - df_B = 8 - 6 = 2$$
$$T_\Delta \sim \chi^2(df_\Delta) \rightarrow p = .033$$

Given that $p < .05$, we can reject the null hypothesis that there is no difference in model fit between Model A and Model B. There is thus support for including the two additional paths. However, given that the sample size is rather large and the effect size (gain in model improvement) is rather small, the LRT suggests only a trivial gain in model fit associated with this modification.

Examining relative tests of model fit for this new model, however, we see that the fit is still terrible. Our attempt to modify the model based upon a priori hypotheses has failed to produce an acceptable model.

Modification to Peer Affiliation Model: Modification Indices

Returning to the original hypothesized model, we will take a different approach to model modification. We can request that lavaan suggest changes to our model based on the expected change in model chi-square if a fixed parameter were freed; these are the modification indices (MIs). We can obtain MIs with the **modindices** function, as shown here:

```
modindices(fit.1, sort.=TRUE, minimum.value=10)
```

The first argument is the fit/results object generated by lavaan for the model of interest, here **fit.1**. The **sort.** argument is set to **TRUE** so that lavaan will present the largest MIs first. Additionally, the **minimum.value** argument is set to suppress printing of any MIs less than 10.

The **minimum.value** argument can be changed to a smaller or larger number, as desired, or the omitted entirely to look at all possible MIs.

Model modification indices are shown below:

```
      lhs op    rhs     mi   epc sepc.lv sepc.all sepc.nox
63    age  ~   peer 53.942 1.114   1.114    0.413    0.413
47   peer  ~    age 42.540 0.129   0.129    0.348    0.241
42 negaff  ~    age 11.889 0.119   0.119    0.179    0.124
62    age  ~ negaff 10.877 0.303   0.303    0.201    0.201
```

Here, **lhs** stands for "on the left-hand side of the operator" and **rhs** stands for "on the right-hand side of the operator." The operator, denoted **op**, for these parameters is **~**, indicating these are regression slopes. **mi** denotes the expected improvement in the model chi square test statistic if the modification is accepted. **epc** denotes "expected parameter change (EPC)" and gives the expected parameter value if the modification is implemented (since it is changing from zero). **sepc.lv** re-scales the EPC by standardizing any latent variables in the model, **sepc.all** re-scales the EPC by standardizing all variables (observed and latent), and **sepc.nox** re-scales the EPC by standardizing all variables except exogenous predictors. Here, **sepc.lv** is equivalent to the EPC because there are no latent variables in the model. It can be helpful to rely on standardized EPC values in order to get a sense of the relative magnitude of each potential modification.

Note the overlap in suggested modification indices. This tells us that our original model does not allow for a significant relation between **age** and **negaff** or between **age** and **peer**. These suggestions are not independent from one another; allowing **negaff** to relate directly with **age** would imply an increased correlation between **negaff** and **peer**.

It is more theoretically justifiable to regress **peer** on **age** than to regress **negaff** on **age**; thus, we will proceed with this model modification, as shown in the code below.

```
pathmodel.3 <-'
              #regressions
              stress ~ coa + gen + age
              emotion ~ coa + gen + age
              negaff ~ stress + emotion
              peer ~ negaff + age

              #covariances
              stress ~~ emotion
              '
fit.3 <- sem(pathmodel.3, data=afdp, meanstructure=TRUE)
summary(fit.3, fit.measures=TRUE)
```

Relative to the model syntax object for our first model, all we have done is add **age** as a predictor to the right of the **~** statement for **peer**.

We obtain the following model fit:

```
lavaan 0.6-2 ended normally after 49 iterations

  Optimization method                           NLMINB
  Number of free parameters                         19

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                      34.368
  Degrees of freedom                                 7
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              251.027
  Degrees of freedom                                18
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.883
  Tucker-Lewis Index (TLI)                       0.698

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -1135.535
  Loglikelihood unrestricted model (H1)      -1118.351

  Number of free parameters                         19
  Akaike (AIC)                                2309.070
  Bayesian (BIC)                              2380.429
  Sample-size adjusted Bayesian (BIC)         2320.166

Root Mean Square Error of Approximation:

  RMSEA                                          0.111
  90 Percent Confidence Interval          0.076  0.150
  P-value RMSEA <= 0.05                          0.003

Standardized Root Mean Square Residual:

  SRMR                                           0.057
```

We can statistically compare the fit of this model with that of the original model using a LRT:

```
lavTestLRT(fit.3, fit.1)
```

```
Chi Square Difference Test

      Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.3  7 2309.1 2380.4 34.368
fit.1  8 2353.9 2421.5 81.173     46.805       1  7.841e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

In other words, we have

$$T_\Delta = T_A - T_B = 81.17 - 34.37 = 46.80$$
$$df_\Delta = df_A - df_B = 8 - 7 = 1$$
$$T_\Delta \sim \chi^2(df_\Delta) \rightarrow p < .001$$

Thus, including age as a predictor of deviant peer affiliation has significantly improved the fit of the model. However, the relative model fit is still poor. We can again examine the modification indices to determine whether a justifiable model modification would lead to further improvements in model fit.

```
modindices(fit.3, sort.=TRUE, minimum.value=10)
```

```
      lhs op    rhs     mi   epc sepc.lv sepc.all sepc.nox
44   peer  ~ stress 13.381 0.149   0.149    0.192    0.192
63    age  ~   peer 12.858 1.977   1.977    0.721    0.721
43 negaff  ~    age 11.889 0.119   0.119    0.179    0.124
46   peer  ~    coa 11.853 0.185   0.185    0.175    0.350
62    age  ~ negaff 10.859 0.303   0.303    0.200    0.200
```

The MIs continue to suggest that **negaff** be directly regressed on **age**, even after we have regressed **peer** on **age**. However, the MIs also suggest regressing **peer** on **coa**, a modification that we consider to be the most theoretically defensible of the suggested modifications.

To make this modification, we run the following script:

```
pathmodel.4 <-'
              #regressions
              stress ~ coa + gen + age
              emotion ~ coa + gen + age
              negaff ~ stress + emotion
              peer ~ negaff + age + coa

              #covariances
              stress ~~ emotion
              '
fit.4 <- sem(pathmodel.4, data=afdp, meanstructure=TRUE)
summary(fit.4, fit.measures=TRUE)
```

We obtain the following model fit:

```
lavaan 0.6-2 ended normally after 50 iterations

  Optimization method                           NLMINB
  Number of free parameters                         20

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                      22.274
  Degrees of freedom                                 6
  P-value (Chi-square)                           0.001

Model test baseline model:

  Minimum Function Test Statistic              251.027
  Degrees of freedom                                18
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.930
  Tucker-Lewis Index (TLI)                       0.790

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -1129.488
  Loglikelihood unrestricted model (H1)      -1118.351

  Number of free parameters                         20
  Akaike (AIC)                                2298.977
  Bayesian (BIC)                              2374.092
  Sample-size adjusted Bayesian (BIC)         2310.657

Root Mean Square Error of Approximation:

  RMSEA                                          0.093
  90 Percent Confidence Interval          0.054  0.135
  P-value RMSEA <= 0.05                          0.038

Standardized Root Mean Square Residual:

  SRMR                                           0.044
```

A LRT would show that the inclusion of a direct path from **coa** to **peer** results in a significant improvement to the model fit; however, we are still not satisfied with the relative model fit. Once again, we turn to the modification indices.

```
modindices(fit.4, sort.=TRUE, minimum.value=10)
```

```
      lhs op    rhs     mi   epc sepc.lv sepc.all sepc.nox
44 negaff  ~    age 11.889 0.119   0.119    0.179    0.124
62    age  ~ negaff 10.875 0.303   0.303    0.201    0.201
```

Modification indices persist in suggesting that **age** is directly related with **negaff**. This is the only remaining modification that has an expected change in model fit that is greater than 10. Because this is not an unreasonable modification, we will make this change as well:

```
pathmodel.5 <-'
              #regressions
              stress ~ coa + gen + age
              emotion ~ coa + gen + age
              negaff ~ stress + emotion + age
              peer ~ negaff + age + coa

              #covariances
              stress ~~ emotion
              '
fit.5 <- sem(pathmodel.5, data=afdp, meanstructure=TRUE)
summary(fit.5, fit.measures=TRUE)
```

And we obtain the following fit to the data:

```
lavaan 0.6-2 ended normally after 55 iterations

  Optimization method                           NLMINB
  Number of free parameters                         21

  Number of observations                           316

  Estimator                                         ML
  Model Fit Test Statistic                      10.156
  Degrees of freedom                                 5
  P-value (Chi-square)                           0.071

Model test baseline model:

  Minimum Function Test Statistic              251.027
  Degrees of freedom                                18
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.978
  Tucker-Lewis Index (TLI)                       0.920

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -1123.429
  Loglikelihood unrestricted model (H1)      -1118.351

  Number of free parameters                         21
```

```
  Akaike (AIC)                                2288.858
  Bayesian (BIC)                              2367.728
  Sample-size adjusted Bayesian (BIC)         2301.122

Root Mean Square Error of Approximation:

  RMSEA                                          0.057
  90 Percent Confidence Interval          0.000  0.108
  P-value RMSEA <= 0.05                          0.345

Standardized Root Mean Square Residual:

  SRMR                                           0.027
```

By including three data-driven but theoretically acceptable modifications to our original model, we have obtained good model fit. The CFI and the TLI are both above .9 and the 90% confidence interval for the RMSEA includes 0. Note, however, that the confidence interval also includes values greater than .10, so the model fit is not outstanding.

We turn now to the raw and standardized parameter estimates associated with our final model, again computing both fully and partially standardized estimates.

```
summary(fit.5, standardized=TRUE, rsquare=TRUE)
summary(fit.5, std.nox=TRUE, rsquare=TRUE)
```

Here we see the raw and fully standardized estimates:

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  stress ~
    coa               0.451    0.072    6.223    0.000    0.451    0.331
    gen              -0.016    0.073   -0.215    0.830   -0.016   -0.011
    age               0.002    0.025    0.078    0.938    0.002    0.004
  emotion ~
    coa               0.110    0.056    1.963    0.050    0.110    0.110
    gen              -0.048    0.056   -0.843    0.399   -0.048   -0.047
    age              -0.027    0.019   -1.374    0.170   -0.027   -0.077
  negaff ~
    stress            0.243    0.077    3.157    0.002    0.243    0.173
    emotion           0.582    0.105    5.568    0.000    0.582    0.305
    age               0.119    0.034    3.515    0.000    0.119    0.179
  peer ~
    negaff            0.137    0.028    4.942    0.000    0.137    0.244
    age               0.138    0.018    7.525    0.000    0.138    0.372
    coa               0.185    0.053    3.512    0.000    0.185    0.172

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .stress ~~
   .emotion           0.112    0.019    5.896    0.000    0.112    0.352
```

```
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .stress            0.687    0.333    2.066    0.039    0.687    1.010
   .emotion           2.341    0.258    9.088    0.000    2.341    4.663
   .negaff           -0.038    0.489   -0.078    0.938   -0.038   -0.040
   .peer             -1.856    0.239   -7.771    0.000   -1.856   -3.457

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .stress            0.412    0.033   12.570    0.000    0.412    0.890
   .emotion           0.247    0.020   12.570    0.000    0.247    0.979
   .negaff            0.749    0.060   12.570    0.000    0.749    0.816
   .peer              0.216    0.017   12.570    0.000    0.216    0.748
```

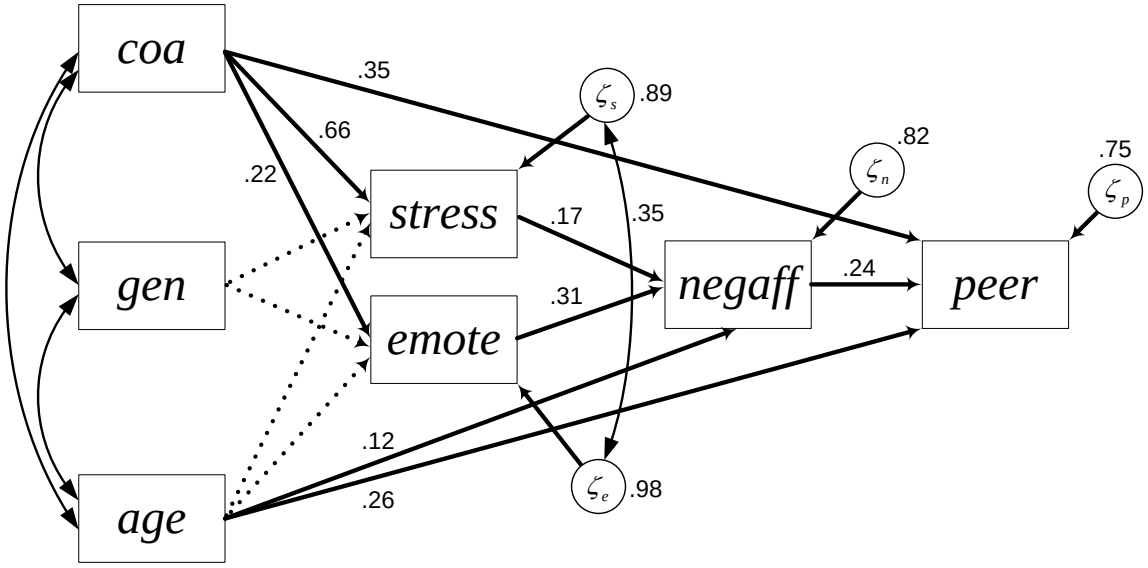
And here we see the raw and partially standardized estimates (just shown for the regression slopes):

```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.nox
  stress ~
    coa               0.451    0.072    6.223    0.000    0.451    0.663
    gen              -0.016    0.073   -0.215    0.830   -0.016   -0.023
    age               0.002    0.025    0.078    0.938    0.002    0.003
  emotion ~
    coa               0.110    0.056    1.963    0.050    0.110    0.219
    gen              -0.048    0.056   -0.843    0.399   -0.048   -0.095
    age              -0.027    0.019   -1.374    0.170   -0.027   -0.053
  negaff ~
    stress            0.243    0.077    3.157    0.002    0.243    0.173
    emotion           0.582    0.105    5.568    0.000    0.582    0.305
    age               0.119    0.034    3.515    0.000    0.119    0.124
  peer ~
    negaff            0.137    0.028    4.942    0.000    0.137    0.244
    age               0.138    0.018    7.525    0.000    0.138    0.257
    coa               0.185    0.053    3.512    0.000    0.185    0.345
```

Finally, the r-square values for the model are given here:

```
R-Square:
                   Estimate
    stress            0.110
    emotion           0.021
    negaff            0.184
    peer              0.252
```

The final path diagram with standardized estimates overlaid is shown below. Note that, as before, partially standardized estimates (from the **std.nox** output) are reported for regression paths emanating from the exogenous variables, **COA**, **gen**, and **age**, as all of these predictors have meaningful metrics. All other values are fully standardized.



We originally hypothesized that **coa** would lead to an increase in uncontrolled stressful life events, and this was supported by the model. COA status leads to a moderately high increase in **stress**. We hypothesized that being a COA would lead to an increase in emotionality, and this was supported: **coa** leads to a small-to-moderate increase in **emotion**. We hypothesized that stress and **emotion** would increase **negaff**, and model results suggest a small, positive effect of **stress** on **negaff** and a moderate effect of **emotion** on **negaff**. Finally, we hypothesized that negative affect would increase affiliation with deviant peers, and we estimated a moderately positive direct relation between **negaff** and **peer**.

Additionally, we found support for a direct effect of **coa** on **peer**, suggesting that **stress**, **emotion**, and **negaff** do not fully account for the total relation between COA status and affiliation with deviant peers. Furthermore, age appears to account for a significant amount of the variation in negative affect and affiliation with deviant peers, but age does not appear to be related to uncontrolled stressful life events or emotionality.

Tests of Direct, Indirect, and Specific Effects

Significant links in a mediational pathway are not sufficient to infer mediation. In order to formally test the full mediation effect, we need an inferential test of the entire specific indirect effect in question. Here, we want to know whether the specific mediational pathway of COA status on deviant peer affiliation via stressful events and negative affect is statistically significant, whether the specific mediational effect of **coa** on **peer** via emotionality and negative affect is significant, and whether the overall indirect effect of **coa** on **peer** is

significant. Finally, we would like to have an estimate of the total effect of **coa** on **peer** considering both direct and indirect pathways.

The first thing we need to do is define the effects of interest in a new model syntax object, as shown below:

```
effects.5 <-'
            #regressions
            stress ~ c_s*coa + gen + age
            emotion ~ c_e*coa + gen + age
            negaff ~ s_n*stress + e_n*emotion + age
            peer ~ n_p*negaff + age + c_p*coa

            #covariances
            stress ~~ emotion

            #direct effect
            dir := c_p

            #specific indirect effects
            ind_s := c_s*s_n*n_p
            ind_e := c_e*e_n*n_p

            #total indirect effect
            tot_ind := c_s*s_n*n_p + c_e*e_n*n_p

            #total effect
            tot := c_s*s_n*n_p + c_e*e_n*n_p + c_p
```

This is precisely the same model as was defined in the **pathmodel.5** syntax object. What's new here is that we've added parameter labels for the paths involved in the direct and indirect effects of **coa** on **peer**. For instance, in the line **stress ~ c_s*coa + gen + age**, we have supplied the label **c_s** for the effect of **coa** on **stress**. Likewise, the other paths involved in computing the effects of interest have also been labeled. To help keep track of things, our labels consist of the first letter of the predictor, an underscore, and first letter of outcome, but any labels are fine. We can now refer to these labels when defining direct, indirect, and total effects.

Following the core model specification, we have new lines to define the direct, specific indirect, total indirect, and total effects. This is done using the "define" operator **:=**. For instance, **ind_s := c_s*s_n*n_p** requests that lavvan compute the quantity **ind_s** as the product of the three paths previously labeled **c_s**, **s_n**, and **n_p**.

If we were to fit this model using the default options, lavaan would generate point estimates for all of the newly defined effects as well as delta-method standard errors. As we noted in lecture, however, the current best practices is to conduct inferential tests of indirect effects using bootstrapped confidence intervals. Fortunately, it is straightforward to compute these using lavaan.

The script for computing bootrapped confidence intervals is shown below:

```
set.seed(62973)
fit.5b <- sem(effects.5, data=afdp, meanstructure=TRUE, se="bootstrap",
bootstrap=1000)
parameterEstimates(fit.5b, boot.ci.type = "perc")
parameterEstimates(fit.5b, boot.ci.type = "bca.simple")
```

Note that we have used the **set.seed** function to set the random number seed for selecting the bootstrapped samples so that we can reproduce exactly these same results each time we run the script.

Then, in the **sem** function, we specified **se="bootstrap", bootstrap=1000** to request that lavaan compute bootstrapped standard errors with 1000 bootstrapped samples. We actually don't care much about the standard errors. What we really want are bootstrapped confidence intervals, which are generated in the **parameterEstimates** function with the **boot.ci.type** argument. The **parameterEstimates** function produces a simple list of the parameter estimates from the model. By including the **boot.ci.type** argument, we request that this output include 95% boostrapped confidence intervals.

Note that there exist multiple methods for calculating bootrapped CIs. The percentile method is probably easiest to understand – the 1000 boostrapped estimates are ordered by magnitude and the 25th estimate (2.5th percentile) and 975th estimate (97.5th percentile) are taken as the confidence limits, yielding a 95% confidence interval. This method, which is widely available in many software programs, is obtained via **boot.ci.type = "perc"**. That is the method we used in the lecture notes. A slightly preferable approach, however, is to use bias-corrected bootstrapped confidence intervals, and these are readily available in lavaan. To obtain the bias-corrected confidence intervals, we simply use **boot.ci.type = "bca.simple"**. The bias-corrected intervals are shown in the output below. In terms of null hypothesis testing, the same conclusions are generated with the bias-corrected CIs as with the percentile CIs presented in lecture, but there are some slight differences in the specific values of the confidence limits.

```
       lhs op                     rhs   label    est    se      z pvalue ci.lower ci.upper
1   stress  ~                     coa     c_s  0.451 0.071  6.385  0.000    0.319    0.597
2   stress  ~                     gen         -0.016 0.075 -0.209  0.835   -0.163    0.126
3   stress  ~                     age          0.002 0.026  0.075  0.940   -0.050    0.055
4  emotion  ~                     coa     c_e  0.110 0.058  1.908  0.056   -0.004    0.226
5  emotion  ~                     gen         -0.048 0.056 -0.843  0.399   -0.167    0.062
6  emotion  ~                     age         -0.027 0.022 -1.215  0.225   -0.072    0.018
7   negaff  ~                  stress     s_n  0.243 0.091  2.674  0.007    0.080    0.430
8   negaff  ~                 emotion     e_n  0.582 0.109  5.324  0.000    0.354    0.787
9   negaff  ~                     age          0.119 0.030  3.891  0.000    0.062    0.180
10    peer  ~                  negaff     n_p  0.137 0.030  4.526  0.000    0.077    0.196
11    peer  ~                     age          0.138 0.016  8.636  0.000    0.108    0.169
12    peer  ~                     coa     c_p  0.185 0.052  3.529  0.000    0.069    0.280
13  stress ~~                 emotion          0.112 0.019  6.006  0.000    0.076    0.150
14  stress ~~                  stress          0.412 0.041 10.132  0.000    0.343    0.507
15 emotion ~~                 emotion          0.247 0.017 14.356  0.000    0.217    0.286
16  negaff ~~                  negaff          0.749 0.066 11.352  0.000    0.637    0.896
17    peer ~~                    peer          0.216 0.029  7.426  0.000    0.162    0.277
18     coa ~~                     coa          0.249 0.000     NA     NA    0.249    0.249
19     coa ~~                     gen          0.002 0.000     NA     NA    0.002    0.002
20     coa ~~                     age         -0.055 0.000     NA     NA   -0.055   -0.055
21     gen ~~                     gen          0.249 0.000     NA     NA    0.249    0.249
```

```
22     gen ~~                            age          -0.070 0.000     NA     NA   -0.070   -0.070
23     age ~~                            age           2.095 0.000     NA     NA    2.095    2.095
24  stress ~1                                          0.687 0.339  2.028  0.043    0.016    1.346
25 emotion ~1                                          2.341 0.291  8.033  0.000    1.777    2.939
26  negaff ~1                                         -0.038 0.449 -0.085  0.932   -0.950    0.880
27    peer ~1                                         -1.856 0.195 -9.514  0.000   -2.236   -1.474
28     coa ~1                                          0.525 0.000     NA     NA    0.525    0.525
29     gen ~1                                          0.538 0.000     NA     NA    0.538    0.538
30     age ~1                                         12.718 0.000     NA     NA   12.718   12.718
31     dir :=                            c_p     dir  0.185 0.052  3.527  0.000    0.069    0.280
32   ind_s :=                  c_s*s_n*n_p   ind_s  0.015 0.007  2.024  0.043    0.005    0.036
33   ind_e :=                  c_e*e_n*n_p   ind_e  0.009 0.005  1.647  0.100    0.000    0.022
34 tot_ind :=     c_s*s_n*n_p+c_e*e_n*n_p tot_ind  0.024 0.010  2.477  0.013    0.009    0.047
35     tot := c_s*s_n*n_p+c_e*e_n*n_p+c_p     tot  0.209 0.054  3.900  0.000    0.094    0.306
```

Here, we are principally concerned with the estimates and bias-adjusted bootstrapped confidence intervals given in lines 31-35.

The total effect of **coa** on **peer** is equal to .209 and represents a combination of the direct effect (.185) and the total indirect effect (.024). The 95% CI is equal to .094 and .306; because this does not contain zero, the total effect is deemed to be significant.

Examining the mediational pathways, we see that the specific indirect effect **coa→stress→negaff→peer**, labeled **ind_s** to indicate it is the indirect effect through **stress**, is equal to .015 (95% CI=.005, .036) and is significant (does not include zero). The biological pathway from **coa→emotion→negaff→peer**, labeled **ind_e** to indicate it is the indirect effect through **emotion**, is equal to .009 (95% CI=0, .022) and thus does not reach statistical significance (because the lower CI is equal to zero).

In sum, examining the specific indirect mediational pathways has provided a more nuanced understanding of how parental alcoholism is related to children's affiliation with deviant peers.

Chapter 4
Confirmatory Factor Analysis

Confirmatory Factor analysis of Holzinger-Swineford Data

The data for this demonstration were provided by Holzinger & Swineford in their 1939 monograph *A Study in Factor Analysis: The Stability of a Bi-Factor Solution*. The sample includes 301 7th and 8th grade students, between 11-16 years of age, drawn from two schools. The data is in the text file **hs.dat** and the accompanying R-script file is **ch04.R**. The variables in the data set that we will use are

visperc	visual perception test in which participants select the next image in a series
cubes	visual perception test in which participants must mentally rotate a cube
lozenges	visual perception test involving mental "flipping" of a parallelogram ("lozenge")
parcomp	paragraph comprehension test
sencomp	sentence completion task in which participants select most appropriate word to put at the end of a sentence
wordmean	verbal ability test in which participants must select a word most similar in meaning to a word used in a sentence.
addition	participants have 2 minutes to complete as many 2-number addition problems as they can
countdot	participants have 4 minutes to count the number of dots in each of a series of dot pictures
sccaps	participants have 3 minutes to indicate whether capital letters are composed entirely of straight lines or include curved lines.

Other variables in the data not included in the models fit here are **school**, **female**, **age**, and **month**.

Preliminary Steps

We begin by reading in the data and assigning variable names, as follows (where the working directory has been set to the source file directory as described in Chapter 1):

```
hs <- read.table("hs.dat", header=FALSE)
names(hs) <- c("school", "female", "age", "month",
               "visperc", "cubes", "lozenges",
               "parcomp", "sencomp", "wordmean",
               "addition", "countdot", "sccaps")
```

Next, we rescaled the variables **addition**, **countdot**, and **sccaps** by dividing by four:

```
hs$addition <- hs$addition/4
hs$countdot <- hs$countdot/4
hs$sccaps <- hs$sccaps/4
```

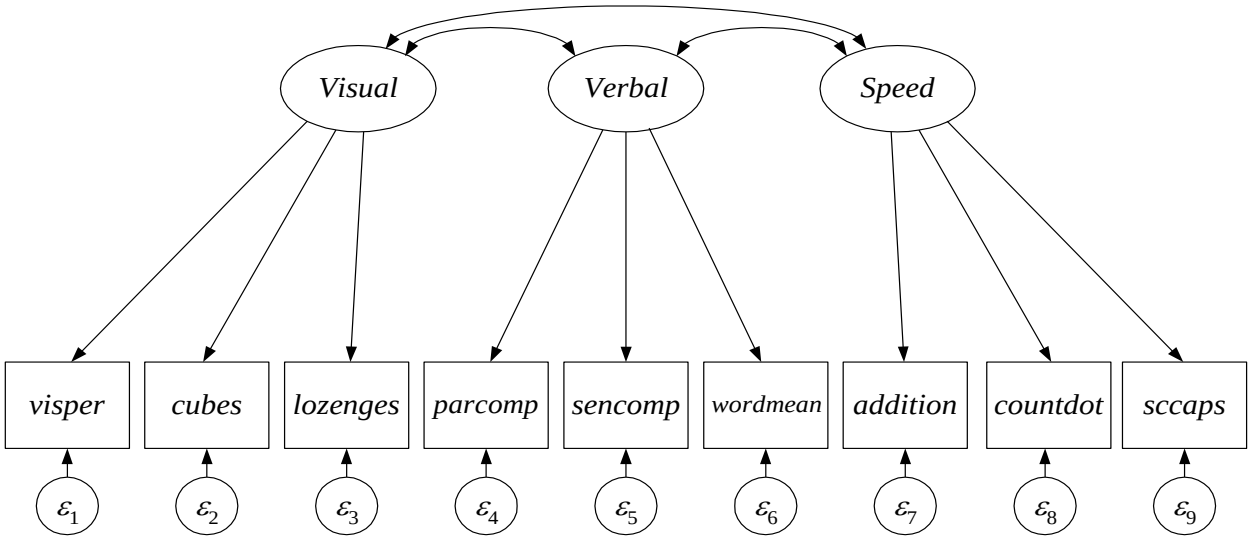
This was done to facilitate model estimation, as numerical instability problems can arise when using variables on widely differing scales. Dividing these variables by four brings their standard deviations closer to the standard deviations of the other observed variables.

Finally, we load the lavaan package that we will use to fit the CFA models:

```
library(lavaan)
```

Initial Model with Standardized Factors

The hypothesized model for the data includes three factors and is shown in the diagram below:



The model is of the form

$$\begin{pmatrix} visper_i \\ cubes_i \\ lozenges_i \\ parcomp_i \\ sencomp_i \\ wordmean_i \\ addition_i \\ countdot_i \\ sccaps_i \end{pmatrix} = \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \\ \nu_6 \\ \nu_7 \\ \nu_8 \\ \nu_9 \end{pmatrix} + \begin{pmatrix} \lambda_{11} & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & \lambda_{42} & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & \lambda_{73} \\ 0 & 0 & \lambda_{83} \\ 0 & 0 & \lambda_{93} \end{pmatrix} \begin{pmatrix} visual_i \\ verbal_i \\ speed_i \end{pmatrix} + \begin{pmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{pmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \theta_{33}, \theta_{44}, \theta_{55}, \theta_{66}, \theta_{77}, \theta_{88}, \theta_{99})$$

$$E(\boldsymbol{\eta}_i) = \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}; \quad COV(\boldsymbol{\eta}_i) = \boldsymbol{\Psi} = \begin{pmatrix} \psi_{11} & & \\ \psi_{21} & \psi_{22} & \\ \psi_{31} & \psi_{32} & \psi_{33} \end{pmatrix}$$

Recall that to identify the model we must set the scale of the latent variables. Two options for doing so are to (1) standardize the latent variables or (2) set the intercept and factor loading of one item per factor to zero and one, respectively. We shall begin with the standardized scaling by setting these parameters as fixed:

$$E(\boldsymbol{\eta}_i) = \boldsymbol{\alpha} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \; COV(\boldsymbol{\eta}_i) = \boldsymbol{\Psi} = \begin{pmatrix} 1 & & \\ \psi_{21} & 1 & \\ \psi_{31} & \psi_{32} & 1 \end{pmatrix}$$

The R script to specify and fit this model is shown below:

```
cfa.1a <-'#factor loadings
          visual =~ visperc + cubes + lozenges
          verbal =~ parcomp + sencomp + wordmean
          speed =~ addition + countdot + sccaps
         '

fit.1a <- cfa(cfa.1a, data=hs, meanstructure=TRUE, std.lv=TRUE)
summary(fit.1a, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)
```

We have seen most of this syntax before, so here we will highlight only portions of the syntax, especially new syntax that involves the use of latent variables.

First, we will use the **cfa** function in lavaan to fit the model. This function imposes a number of defaults consistent with how CFA models are typically specified and enables us to specify models with compact model syntax objects. The default scaling for the latent variables is a hybrid of the approaches we described in lecture. By default, the latent variables will have means set to zero but freely estimated variance. Additionally, the factor loading of the first indicator for each latent variable will be set to one but the intercept for this indicator will be freely estimated. We will override these defaults to implement the scaling options described in lecture.

Second, in defining the model syntax object, we implement the **=~** operator, which is used to define latent variables (left hand side) and to specify the variables that load on them (right hand side). Thus, **visual =~ visperc + cubes + lozenges** defines a new latent variable, **visual**, and indicates that **visperc**, **cubes**, and **lozenges** are indicator variables.

Third, in fitting the model, we specify **std.lv=TRUE** within the **cfa** function call. This overrides the default hybrid scaling and instead asks lavaan to (1) standardize the factors and (2) freely estimate all loadings.

Let us now turn to the output, considering first the fit indices for the model:

```
lavaan 0.6-2 ended normally after 28 iterations

  Optimization method                           NLMINB
  Number of free parameters                         30
  Number of observations                           301
  Estimator                                         ML
  Model Fit Test Statistic                      85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              918.852
  Degrees of freedom                                36
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.931
  Tucker-Lewis Index (TLI)                       0.896

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -8326.241
  Loglikelihood unrestricted model (H1)      -8283.589

  Number of free parameters                         30
  Akaike (AIC)                               16712.483
  Bayesian (BIC)                             16823.696
  Sample-size adjusted Bayesian (BIC)        16728.553

Root Mean Square Error of Approximation:

  RMSEA                                          0.092
  90 Percent Confidence Interval          0.071  0.114
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.060
```

We see here that the fit indices indicate that the model does not fit the data particularly well.

The estimates (both raw and fully standardized) for the model are shown next. These values must be interpreted cautiously, given the lack of fit of the model.

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    visperc           5.398    0.485   11.128    0.000    5.398    0.772
    cubes             1.992    0.310    6.429    0.000    1.992    0.424
    lozenges          5.249    0.595    8.817    0.000    5.249    0.581
  verbal =~
    parcomp           2.969    0.170   17.474    0.000    2.969    0.852
    sencomp           4.406    0.251   17.576    0.000    4.406    0.855
    wordmean          6.416    0.376   17.082    0.000    6.416    0.838
  speed =~
    addition          3.562    0.400    8.903    0.000    3.562    0.570
    countdot          3.655    0.330   11.090    0.000    3.655    0.723
    sccaps            6.030    0.585   10.305    0.000    6.030    0.665

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    verbal            0.459    0.064    7.189    0.000    0.459    0.459
    speed             0.471    0.073    6.461    0.000    0.471    0.471
  verbal ~~
    speed             0.283    0.069    4.117    0.000    0.283    0.283

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          29.615    0.403   73.473    0.000   29.615    4.235
   .cubes            24.352    0.271   89.855    0.000   24.352    5.179
   .lozenges         18.003    0.521   34.579    0.000   18.003    1.993
   .parcomp           9.183    0.201   45.694    0.000    9.183    2.634
   .sencomp          17.362    0.297   58.452    0.000   17.362    3.369
   .wordmean         15.299    0.441   34.667    0.000   15.299    1.998
   .addition         24.069    0.360   66.766    0.000   24.069    3.848
   .countdot         27.635    0.291   94.854    0.000   27.635    5.467
   .sccaps           48.367    0.523   92.546    0.000   48.367    5.334
    visual            0.000                             0.000    0.000
    verbal            0.000                             0.000    0.000
    speed             0.000                             0.000    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          19.766    4.090    4.833    0.000   19.766    0.404
   .cubes            18.141    1.628   11.146    0.000   18.141    0.821
   .lozenges         54.037    5.800    9.317    0.000   54.037    0.662
   .parcomp           3.341    0.429    7.779    0.000    3.341    0.275
   .sencomp           7.140    0.934    7.642    0.000    7.140    0.269
   .wordmean         17.454    2.109    8.277    0.000   17.454    0.298
   .addition         26.430    2.691    9.823    0.000   26.430    0.676
   .countdot         12.192    1.855    6.573    0.000   12.192    0.477
   .sccaps           45.857    5.730    8.003    0.000   45.857    0.558
```

```
    visual            1.000                           1.000    1.000
    verbal            1.000                           1.000    1.000
    speed             1.000                           1.000    1.000
```

Note that the standard error, z-statistic, and p-value for the factor means and variances are all missing. This simply indicates that these parameters were not estimated, and hence no inferential tests were conducted on their values.

Because the items are on different scales, we cannot easily interpret the differences among the raw estimates for the factor loadings or residual variances. To better interpret these results, we requested the standardized solution. Note that because we have already standardized the latent variables, the column labeled **Std.lv** is redundant with the raw estimates. We are more interested in the fully-standardized **Std.all** values. The fully-standardized factor loadings can be compared directly (e.g., **visperc** is a better indicator than **cubes** for the **visual** factor).

One odd thing to note is that the indicator intercepts are not zero in the standardized output, as you would expect if the observed variables had been standardized in the usual way of deviating the mean and dividing by the standard deviation. In lavaan, the standardized solution merely rescales the observed variables to have standard deviations of one, and does not center the variables to have means of zero. This is of no consequence here, as the mean structure is saturated and of little interest.

Last, the standardized residual variances of the indicators are interpretable as the proportion of variance unexplained by the latent factors (or 1 – communality). The communalities themselves were obtained by setting the **rsquare** argument of the **summary** function to **TRUE**:

```
R-Square:
                   Estimate
    visperc           0.596
    cubes             0.179
    lozenges          0.338
    parcomp           0.725
    sencomp           0.731
    wordmean          0.702
    addition          0.324
    countdot          0.523
    sccaps            0.442
```

Items with high communality are generally regarded as better items. Given the poor model fit, we may wish to examine modification indices to get an idea of where the model may be misspecified.

The modification indices for the model are obtained from the **modindices** function as follows:

```
modindices(fit.1a, sort.=TRUE, minimum.value=10)
```

```
        lhs op      rhs     mi     epc sepc.lv sepc.all sepc.nox
42   visual =~   sccaps 36.411   4.672   4.672    0.515    0.515
88 addition ~~ countdot 34.145  15.423  15.423    0.859    0.859
40   visual =~ addition 18.631  -2.182  -2.182   -0.349   -0.349
90 countdot ~~   sccaps 14.946 -19.039 -19.039   -0.805   -0.805
```

Here we see the largest modification indices are associated with a cross-loading for **sccaps** on **visual**, and a correlated uniqueness for **countdot** with **addition**. It is important to keep in mind that both modification indices may be related to the same misspecification.

Before proceeding to respecify the model, let us also consider how the model would be input into lavaan if we chose to scale the latent variables using scaling items rather than standardizing.

Initial Model with Scaling Items

We shall choose the first indicator for each factor to be the scaling item. The intercept and factor loading for each scaling item is set to zero and one, respectively. This scaling option permits the means and variances of the latent factors to be estimated. The model is thus now specified as

$$\begin{pmatrix} visper_i \\ cubes_i \\ lozenges_i \\ parcomp_i \\ sencomp_i \\ wordmean_i \\ addition_i \\ countdot_i \\ sccaps_i \end{pmatrix} = \begin{pmatrix} 0 \\ \nu_2 \\ \nu_3 \\ 0 \\ \nu_5 \\ \nu_6 \\ 0 \\ \nu_8 \\ \nu_9 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \lambda_{83} \\ 0 & 0 & \lambda_{93} \end{pmatrix} \begin{pmatrix} visual_i \\ verbal_i \\ speed_i \end{pmatrix} + \begin{pmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{pmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \theta_{33}, \theta_{44}, \theta_{55}, \theta_{66}, \theta_{77}, \theta_{88}, \theta_{99})$$

$$E(\boldsymbol{\eta}_i) = \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}; \quad COV(\boldsymbol{\eta}_i) = \boldsymbol{\Psi} = \begin{pmatrix} \psi_{11} & & \\ \psi_{21} & \psi_{22} & \\ \psi_{31} & \psi_{32} & \psi_{33} \end{pmatrix}$$

and all elements in $\boldsymbol{\alpha}$ and $\boldsymbol{\Psi}$ are now estimated.

The corresponding R script for fitting the model and displaying the results is given here:

```
cfa.1b <-'#factor loadings
          visual =~ visperc + cubes + lozenges
          verbal =~ parcomp + sencomp + wordmean
          speed =~ addition + countdot + sccaps

          #means/intercepts
          visual ~ 1
          verbal ~ 1
          speed ~ 1
          visperc ~ 0*1
          parcomp ~ 0*1
          addition ~ 0*1
         '

fit.1b <- cfa(cfa.1b, data=hs, meanstructure=TRUE)
summary(fit.1b, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)
```

In this case, we will not invoke the **std.lv** option when fitting the model as we do not wish to standardize the factors. Here, we wish to use a scaling indicator instead. We can rely on the default that the factor loading for the first indicator per factor will be fixed to one (this need not be specified explicitly). We must, however, assign a numeric value of zero to the intercept for each of the anchor items, e.g., via **visperc ~ 0*1**, to override the default of freely estimating intercepts for all observed variables.

Similarly, we can rely on the default that the factor variances will be freely estimated but we need to indicate that lavvan should estimate the factor means, e.g., via **visual ~ 1**. (Since there are no predictors of the factors, their intercepts are interpretable as means).

The model fit output is shown here:

```
lavaan 0.6-2 ended normally after 180 iterations

  Optimization method                           NLMINB
  Number of free parameters                         30

  Number of observations                           301

  Estimator                                         ML
  Model Fit Test Statistic                      85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              918.852
  Degrees of freedom                                36
  P-value                                        0.000
```

```
User model versus baseline model:

  Comparative Fit Index (CFI)                    0.931
  Tucker-Lewis Index (TLI)                       0.896

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -8326.241
  Loglikelihood unrestricted model (H1)      -8283.589

  Number of free parameters                         30
  Akaike (AIC)                               16712.483
  Bayesian (BIC)                             16823.696
  Sample-size adjusted Bayesian (BIC)        16728.553

Root Mean Square Error of Approximation:

  RMSEA                                          0.092
  90 Percent Confidence Interval          0.071  0.114
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.060
```

Note that the tests of model fit are identical to the standardized factor model presented previously. The two models are equivalent, and merely scaled differently.

The difference in scales is apparent when considering the model estimates:

```
Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model          Structured
  Standard Errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    visperc           1.000                               5.398    0.772
    cubes             0.369    0.066    5.554    0.000    1.992    0.424
    lozenges          0.972    0.145    6.685    0.000    5.249    0.581
  verbal =~
    parcomp           1.000                               2.969    0.852
    sencomp           1.484    0.087   17.014    0.000    4.406    0.855
    wordmean          2.161    0.129   16.703    0.000    6.416    0.838
  speed =~
    addition          1.000                               3.562    0.570
    countdot          1.026    0.143    7.152    0.000    3.655    0.723
    sccaps            1.693    0.237    7.155    0.000    6.030    0.665
```

```
Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    verbal            7.348    1.323    5.552    0.000    0.459    0.459
    speed             9.047    1.942    4.660    0.000    0.471    0.471
  verbal ~~
    speed             2.993    0.851    3.518    0.000    0.283    0.283

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    visual           29.615    0.403   73.473    0.000    5.487    5.487
    verbal            9.183    0.201   45.694    0.000    3.093    3.093
    speed            24.069    0.360   66.766    0.000    6.757    6.757
   .visperc           0.000                             0.000    0.000
   .parcomp           0.000                             0.000    0.000
   .addition          0.000                             0.000    0.000
   .cubes            13.424    1.985    6.762    0.000   13.424    2.855
   .lozenges        -10.797    4.336   -2.490    0.013  -10.797   -1.195
   .sencomp           3.734    0.831    4.496    0.000    3.734    0.725
   .wordmean         -4.545    1.233   -3.685    0.000   -4.545   -0.594
   .countdot          2.940    3.472    0.847    0.397    2.940    0.582
   .sccaps            7.622    5.730    1.330    0.183    7.622    0.841

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          19.766    4.090    4.833    0.000   19.766    0.404
   .cubes            18.141    1.628   11.146    0.000   18.141    0.821
   .lozenges         54.037    5.800    9.317    0.000   54.037    0.662
   .parcomp           3.341    0.429    7.779    0.000    3.341    0.275
   .sencomp           7.140    0.934    7.642    0.000    7.140    0.269
   .wordmean         17.454    2.109    8.277    0.000   17.454    0.298
   .addition         26.430    2.691    9.823    0.000   26.430    0.676
   .countdot         12.192    1.855    6.573    0.000   12.192    0.477
   .sccaps           45.857    5.730    8.003    0.000   45.857    0.558
    visual           29.135    5.237    5.564    0.000    1.000    1.000
    verbal            8.815    1.009    8.737    0.000    1.000    1.000
    speed            12.688    2.850    4.451    0.000    1.000    1.000

R-Square:
                   Estimate
    visperc           0.596
    cubes             0.179
    lozenges          0.338
    parcomp           0.725
    sencomp           0.731
    wordmean          0.702
    addition          0.324
    countdot          0.523
    sccaps            0.442
```

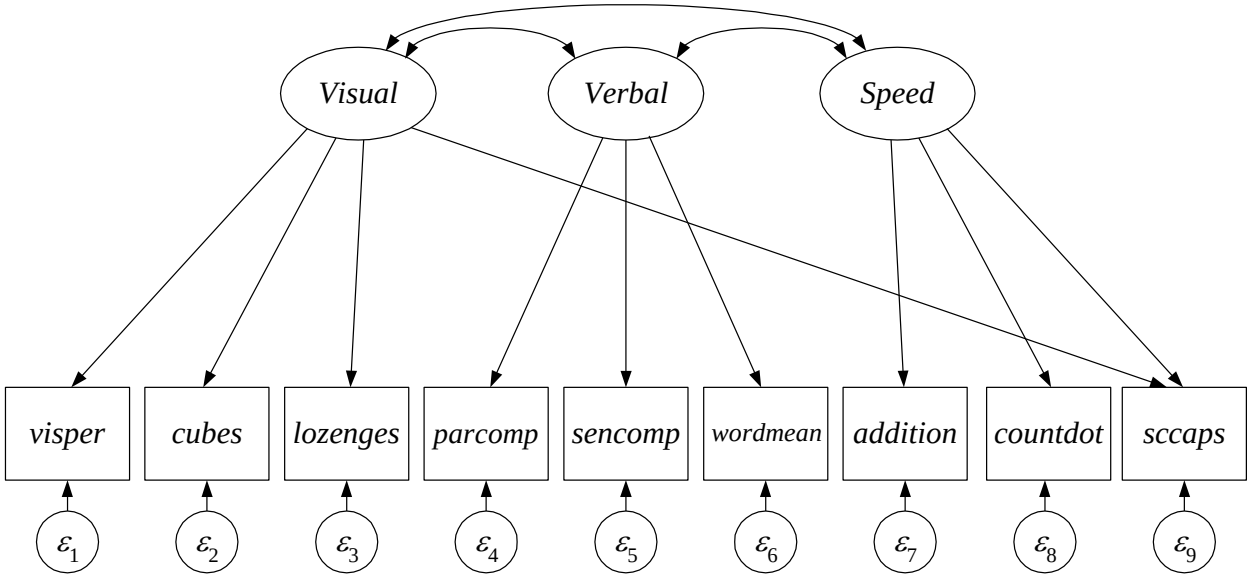
The raw estimates for the factor loadings and intercepts are now interpreted in a relative scale, with reference to the scaling item. Further, because the factor variances are no longer set to one, the covariances among the factors can no longer be interpreted as correlations.

Again, the difference in metric across the tests makes interpretation of differences in intercepts and loadings somewhat difficult. The standardized solution can again be considered to aid in interpretation. Note that it is identical to the standardized solution shown previously. Similarly, modification indices are no different with this scaling option and are not repeated here.

We now consider respecification of the model, returning to the standardized scaling option to set the metric of the latent variables.

Model Modification

We first introduce a cross loading of **`sccaps`** on **`visual`**, as shown in the diagram below:



The model is now of the form

$$\begin{pmatrix} visper_i \\ cubes_i \\ lozenges_i \\ parcomp_i \\ sencomp_i \\ wordmean_i \\ addition_i \\ countdot_i \\ sccaps_i \end{pmatrix} = \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \\ \nu_6 \\ \nu_7 \\ \nu_8 \\ \nu_9 \end{pmatrix} + \begin{pmatrix} \lambda_{11} & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & \lambda_{42} & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & \lambda_{73} \\ 0 & 0 & \lambda_{83} \\ \lambda_{91} & 0 & \lambda_{93} \end{pmatrix} \begin{pmatrix} visual_i \\ verbal_i \\ speed_i \end{pmatrix} + \begin{pmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{pmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \theta_{33}, \theta_{44}, \theta_{55}, \theta_{66}, \theta_{77}, \theta_{88}, \theta_{99})$$

$$E(\boldsymbol{\eta}_i) = \boldsymbol{\alpha} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \; COV(\boldsymbol{\eta}_i) = \boldsymbol{\Psi} = \begin{pmatrix} 1 & & \\ \psi_{21} & 1 & \\ \psi_{31} & \psi_{32} & 1 \end{pmatrix}$$

The difference between the modified model and the original CFA is that the element in 9th row (corresponding to **sccaps**) and 1st column (corresponding to **visual**) of the factor loading matrix has been freed from zero to λ_{91}.

The new cross-loading can be estimated by defining and fitting a slightly modified model syntax object, **cfa.2**, as follows:

```
cfa.2 <- '#factor loadings
          visual =~ visperc + cubes + lozenges + sccaps
          verbal =~ parcomp + sencomp + wordmean
          speed =~ addition + countdot + sccaps
         '
fit.2 <- cfa(cfa.2, data=hs, meanstructure=TRUE, std.lv=TRUE)
summary(fit.2, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)
```

Notice that **sccaps** now appears twice: once on **speed** and once on **visual**. Factor loadings for **sccaps** are freely estimated for both factors.

The model fit information is shown here:

```
lavaan 0.6-2 ended normally after 29 iterations

  Optimization method                           NLMINB
  Number of free parameters                         31

  Number of observations                           301

  Estimator                                         ML
  Model Fit Test Statistic                      52.382
  Degrees of freedom                                23
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              918.852
  Degrees of freedom                                36
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.967
  Tucker-Lewis Index (TLI)                       0.948
```

```
Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -8309.780
  Loglikelihood unrestricted model (H1)      -8283.589

  Number of free parameters                         31
  Akaike (AIC)                               16681.560
  Bayesian (BIC)                             16796.480
  Sample-size adjusted Bayesian (BIC)        16698.166

Root Mean Square Error of Approximation:

  RMSEA                                          0.065
  90 Percent Confidence Interval          0.042  0.089
  P-value RMSEA <= 0.05                          0.133

Standardized Root Mean Square Residual:

  SRMR                                           0.041
```

Freeing the cross-loading significantly improved the model fit according to a likelihood ratio test of the original CFA and the model estimated above:

```
lavTestLRT(fit.2, fit.1a)
```

```
Chi Square Difference Test

       Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.2  23 16682 16797 52.382
fit.1a 24 16713 16824 85.305     32.923       1  9.586e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Thus we have:

$$\chi^2\left(df_{Original} - df_{CrossLoad}\right) = \chi^2{}_{Original} - \chi^2{}_{CrossLoad}$$

$$\chi^2\left(24-23\right) = 85.305 - 52.382$$

$$\chi^2\left(1\right) = 32.923,\ p < .001$$

Other fit indices suggest that the modified model may have satisfactory fit to the data.

The estimates for the model are shown next. To the extent that we are justified in including the new cross loading, we can have more faith in these estimates due to improved model fit.

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    visperc           5.308    0.454   11.685    0.000    5.308    0.759
    cubes             2.046    0.305    6.698    0.000    2.046    0.435
    lozenges          5.333    0.577    9.241    0.000    5.333    0.590
    sccaps            3.480    0.577    6.034    0.000    3.480    0.384
```

```
  verbal =~
    parcomp           2.967    0.170   17.455    0.000    2.967    0.851
    sencomp           4.411    0.251   17.602    0.000    4.411    0.856
    wordmean          6.414    0.376   17.073    0.000    6.414    0.838
  speed =~
    addition          3.830    0.419    9.134    0.000    3.830    0.612
    countdot          4.021    0.368   10.934    0.000    4.021    0.795
    sccaps            4.049    0.593    6.824    0.000    4.049    0.447

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    verbal            0.453    0.062    7.242    0.000    0.453    0.453
    speed             0.301    0.080    3.763    0.000    0.301    0.301
  verbal ~~
    speed             0.206    0.070    2.937    0.003    0.206    0.206

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          29.615    0.403   73.473    0.000   29.615    4.235
   .cubes            24.352    0.271   89.855    0.000   24.352    5.179
   .lozenges         18.003    0.521   34.579    0.000   18.003    1.993
   .sccaps           48.367    0.523   92.546    0.000   48.367    5.334
   .parcomp           9.183    0.201   45.694    0.000    9.183    2.634
   .sencomp          17.362    0.297   58.452    0.000   17.362    3.369
   .wordmean         15.299    0.441   34.667    0.000   15.299    1.998
   .addition         24.069    0.360   66.766    0.000   24.069    3.848
   .countdot         27.635    0.291   94.854    0.000   27.635    5.467
    visual            0.000                             0.000    0.000
    verbal            0.000                             0.000    0.000
    speed             0.000                             0.000    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          20.726    3.616    5.731    0.000   20.726    0.424
   .cubes            17.924    1.607   11.153    0.000   17.924    0.811
   .lozenges         53.148    5.586    9.515    0.000   53.148    0.651
   .sccaps           45.234    4.845    9.336    0.000   45.234    0.550
   .parcomp           3.353    0.430    7.800    0.000    3.353    0.276
   .sencomp           7.098    0.934    7.602    0.000    7.098    0.267
   .wordmean         17.483    2.110    8.285    0.000   17.483    0.298
   .addition         24.450    2.845    8.595    0.000   24.450    0.625
   .countdot          9.383    2.362    3.973    0.000    9.383    0.367
    visual            1.000                             1.000    1.000
    verbal            1.000                             1.000    1.000
    speed             1.000                             1.000    1.000
```

```
R-Square:
                   Estimate
    visperc           0.576
    cubes             0.189
    lozenges          0.349
    sccaps            0.450
    parcomp           0.724
    sencomp           0.733
    wordmean          0.702
    addition          0.375
    countdot          0.633
```

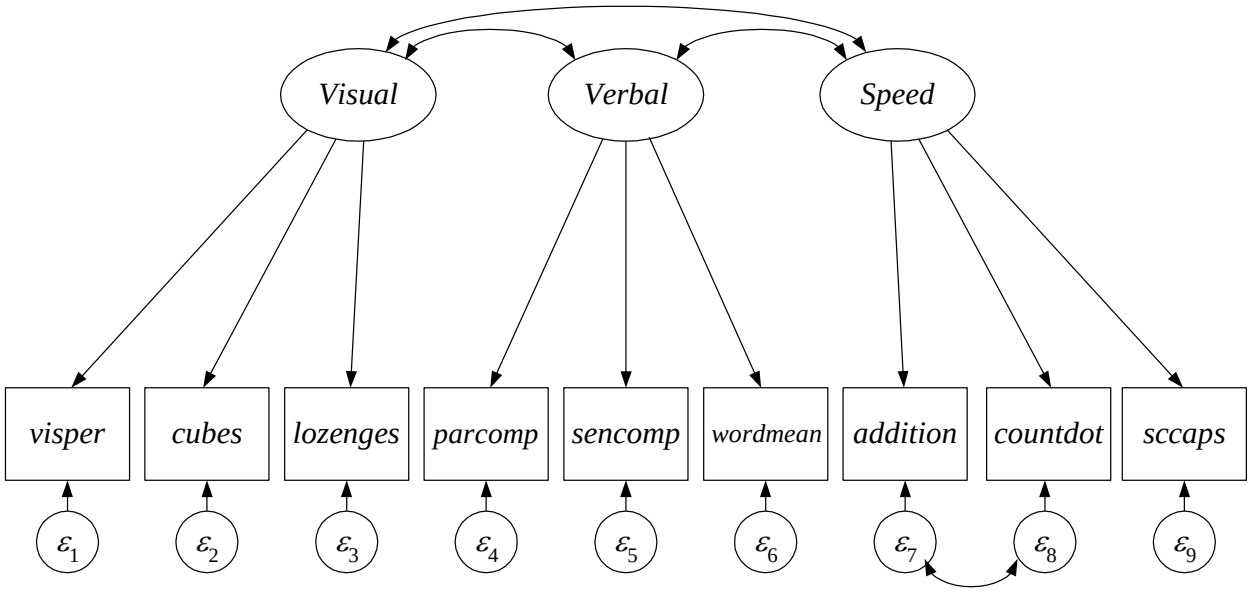
In the results, note that **sccaps** loads significantly on **visual**. The residual variance of **sccaps** remains fairly high and the correlation between **visual** and **speed** is still statistically significant. The new modification indices exceeding 10 are reported below:

```
modindices(fit.2, sort.=TRUE, minimum.value=10)
```

```
[1] lhs      op       rhs      mi       epc      sepc.lv  sepc.all sepc.nox
<0 rows> (or 0-length row.names)
```

Note that no further changes are suggested for the model.

Next, we illustrate how implementing a different model modification might affect conclusions drawn from the model. We allow the residual errors for **addition** and **countdot** to correlate, as shown in the diagram below:



The model is now of the form

$$\begin{pmatrix} visper_i \\ cubes_i \\ lozenges_i \\ parcomp_i \\ sencomp_i \\ wordmean_i \\ addition_i \\ countdot_i \\ sccaps_i \end{pmatrix} = \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \\ \nu_6 \\ \nu_7 \\ \nu_8 \\ \nu_9 \end{pmatrix} + \begin{pmatrix} \lambda_{11} & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & \lambda_{42} & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & \lambda_{73} \\ 0 & 0 & \lambda_{83} \\ 0 & 0 & \lambda_{93} \end{pmatrix} \begin{pmatrix} visual_i \\ verbal_i \\ speed_i \end{pmatrix} + \begin{pmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{pmatrix}$$

where

$$COV\left(\boldsymbol{\varepsilon}_i\right) = \boldsymbol{\Theta} = \begin{pmatrix} \theta_{11} & & & & & & & & \\ 0 & \theta_{22} & & & & & & & \\ 0 & 0 & \theta_{33} & & & & & & \\ 0 & 0 & 0 & \theta_{44} & & & & & \\ 0 & 0 & 0 & 0 & \theta_{55} & & & & \\ 0 & 0 & 0 & 0 & 0 & \theta_{66} & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \theta_{77} & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \theta_{87} & \theta_{88} & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta_{99} \end{pmatrix}$$

$$E\left(\boldsymbol{\eta}_i\right) = \boldsymbol{\alpha} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \quad COV\left(\boldsymbol{\eta}_i\right) = \boldsymbol{\Psi} = \begin{pmatrix} 1 & & \\ \psi_{21} & 1 & \\ \psi_{31} & \psi_{32} & 1 \end{pmatrix}$$

The difference between the modified model and the original CFA is that $\boldsymbol{\Theta}$ is no longer diagonal; it contains a covariance between the residual terms for **`addition`** and **`countdot`**. This change can be included in the model by defining and fitting a new model syntax object, **`cfa.3`**, as follows:

```
cfa.3 <- '#factor loadings
          visual =~ visperc + cubes + lozenges
          verbal =~ parcomp + sencomp + wordmean
          speed =~ addition + countdot + sccaps

          #covariances
          addition ~~ countdot
         '
fit.3 <- cfa(cfa.3, data=hs, meanstructure=TRUE, std.lv=TRUE)
summary(fit.3, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)
```

The difference between this input and the original input is the final statement: **addition ~~ countdot**. In lavaan, when observed variable names are joined by the ~~ operator, it indicates that the residuals associated with those variables should covary (i.e., be correlated).

We first consider the model fit:

```
lavaan 0.6-2 ended normally after 51 iterations

  Optimization method                           NLMINB
  Number of free parameters                         31

  Number of observations                           301

  Estimator                                         ML
  Model Fit Test Statistic                      53.272
  Degrees of freedom                                23
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic              918.852
  Degrees of freedom                                36
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.966
  Tucker-Lewis Index (TLI)                       0.946

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -8310.225
  Loglikelihood unrestricted model (H1)      -8283.589

  Number of free parameters                         31
  Akaike (AIC)                               16682.450
  Bayesian (BIC)                             16797.370
  Sample-size adjusted Bayesian (BIC)        16699.056

Root Mean Square Error of Approximation:

  RMSEA                                          0.066
  90 Percent Confidence Interval          0.043  0.090
  P-value RMSEA <= 0.05                          0.118

Standardized Root Mean Square Residual:

  SRMR                                           0.043
```

Allowing a single residual correlation among these items resulted in a large improvement in model fit, which we can evaluate via a likelihood ratio test (i.e., chi-square difference test):

```
lavTestLRT(fit.3, fit.1a)
```

```
Chi Square Difference Test

       Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.3  23 16682 16797 53.272
fit.1a 24 16713 16824 85.305     32.033       1  1.516e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The improvement in model fit is highly significant:

$$\chi^2\left(df_{Original} - df_{ResCorr}\right) = \chi^2{}_{Original} - \chi^2{}_{ResCorr}$$
$$\chi^2\left(24-23\right) = 85.305 - 53.272$$
$$\chi^2\left(1\right) = 32.033,\ p < .001$$

Other fit indices suggest that the modified model may also have satisfactory overall fit to the data.

The estimates for the model are shown next. To the extent that we are justified in including the unhypothesized cross loading, we can have more faith in these estimates due to improved model fit.

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    visperc           5.308    0.457   11.620    0.000    5.308    0.759
    cubes             2.037    0.305    6.673    0.000    2.037    0.433
    lozenges          5.323    0.576    9.238    0.000    5.323    0.589
  verbal =~
    parcomp           2.967    0.170   17.458    0.000    2.967    0.851
    sencomp           4.411    0.251   17.601    0.000    4.411    0.856
    wordmean          6.413    0.376   17.068    0.000    6.413    0.838
  speed =~
    addition          2.202    0.421    5.229    0.000    2.202    0.352
    countdot          2.383    0.366    6.505    0.000    2.383    0.471
    sccaps            8.667    0.958    9.051    0.000    8.667    0.956

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .addition ~~
   .countdot         10.140    1.936    5.239    0.000   10.140    0.389
  visual ~~
    verbal            0.457    0.064    7.142    0.000    0.457    0.457
    speed             0.544    0.078    6.965    0.000    0.544    0.544
  verbal ~~
    speed             0.270    0.065    4.141    0.000    0.270    0.270
```

```
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          29.615    0.403   73.473    0.000   29.615    4.235
   .cubes            24.352    0.271   89.855    0.000   24.352    5.179
   .lozenges         18.003    0.521   34.579    0.000   18.003    1.993
   .parcomp           9.183    0.201   45.694    0.000    9.183    2.634
   .sencomp          17.362    0.297   58.452    0.000   17.362    3.369
   .wordmean         15.299    0.441   34.667    0.000   15.299    1.998
   .addition         24.069    0.360   66.766    0.000   24.069    3.848
   .countdot         27.635    0.291   94.854    0.000   27.635    5.467
   .sccaps           48.367    0.523   92.546    0.000   48.367    5.334
    visual            0.000                             0.000    0.000
    verbal            0.000                             0.000    0.000
    speed             0.000                             0.000    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .visperc          20.730    3.651    5.678    0.000   20.730    0.424
   .cubes            17.960    1.608   11.171    0.000   17.960    0.812
   .lozenges         53.255    5.576    9.552    0.000   53.255    0.653
   .parcomp           3.350    0.430    7.791    0.000    3.350    0.276
   .sencomp           7.099    0.934    7.600    0.000    7.099    0.267
   .wordmean         17.496    2.111    8.287    0.000   17.496    0.298
   .addition         34.268    2.980   11.501    0.000   34.268    0.876
   .countdot         19.870    1.989    9.988    0.000   19.870    0.778
   .sccaps            7.090   15.210    0.466    0.641    7.090    0.086
    visual            1.000                             1.000    1.000
    verbal            1.000                             1.000    1.000
    speed             1.000                             1.000    1.000

R-Square:
                   Estimate
    visperc           0.576
    cubes             0.188
    lozenges          0.347
    parcomp           0.724
    sencomp           0.733
    wordmean          0.702
    addition          0.124
    countdot          0.222
    sccaps            0.914
```

Note that the newly freed parameter, **addition ~~ countdot** is statistically significantly different from zero. The standardized estimate indicates that **addition** and **countdots** are moderately correlated above and beyond the correlation implied by the common **speed** factor (r = .389). Note further the substantial reduction in residual variance estimates for items loading on **speed**. Specifically, even though **sccaps** is not directly involved in the residual correlation between **addition** and **countdot**, its residual variance was substantially reduced. By accounting for the local dependence between **addition** and **countdot** with the residual covariance, **speed** was able to account for more of the variance in **sccaps**. This is further

evidenced by the fact that **sccaps** is now the highest loading item on the factor. It now appears that **sccaps** is almost definitional of the **speed** factor, with a communality of .914, whereas the communalities of **countdots** and **addition** are much lower.

As with the previous example, introducing this single new parameter resulted in no further sizeable modification indices.

```
modindices(fit.3, sort.=TRUE, minimum.value=10)
```

```
[1] lhs      op       rhs      mi       epc      sepc.lv  sepc.all sepc.nox
<0 rows> (or 0-length row.names)
```

This model is not nested with the alternative modified model; however, the two models provide nearly equivalent fit (RMSEA=.065 versus .066, TLI=.948 versus .946, etc.). Thus, model selection should be based on which modification is most plausible, and upon the interpretability of parameter estimates.

Chapter 5
Structural Equation Models with Latent Variables

Structural Equation Modeling of Şenol-Durak and Ayvaşik's Posttraumatic Growth Data

The data for this demonstration were provided by Şenol-Durak & Ayvaşik in their 2010 *Journal of Health Psychology* manuscript, "Factors associated with posttraumatic growth among the spouses of myocardial infraction patients." The sample includes 132 spouses of myocardial infarction patients. The correlation matrix is in the text file **`mip.dat`**. We will enter the means and standard deviations into R directly. The R script for this chapter is contained in the file **`ch05.R`**.

The variables in the data set that we will use are

`fa`	social support from family	*Environmental Resources*
`fr`	social support from friends	
`si`	social support from significant others	
`lo`	Locus of Control Scale score	*Individual Resources*
`comt`	Commitment score	
`con`	Control score	
`cha`	Challenge score	
`es`	Rosenberg Self Esteem Scale score	
`pro`	subjective evaluation of prognosis	*Event Related Factors*
`th`	threat to future health	
`time`	time since diagnosis	
`em`	emotion focused coping	*Cognitive Process Coping*
`ind`	indirect coping (labeled **`in`** in diagrams)	
`ru`	rumination	
`av`	avoidance	
`hy`	hypervigilance	
`rb`	religious beliefs	
`ptgi1`	improved relationships	*Posttraumatic Growth*
`ptgi2`	new possibilities for one's life	
`ptgi3`	greater appreciation of life	
`ptgi4`	greater sense of personal strength	
`ptgi5`	spiritual development	

Refer to the article for definitions of variables not included in the model.

Preliminary Steps

This example illustrates how summary statistics can be used to fit structural equation models in lavaan. To do so, we require the covariance matrix of the observed variables. For generality, we will also bring in the mean vector, although it plays no part in the fit of this particular model (because the model is saturated in the means). In this case, we have the correlation matrix and thus must construct the covariance matrix from the correlations and standard deviations.

To begin, we load the lavaan package

```
library(lavaan)
```

We then define the variable names (this includes variables not used in the models fit here):

```
names <- c("ptgi","ptgi1", "ptgi2", "ptgi3", "ptgi4", "ptgi5",
           "marital", "fa", "fr", "si", "child", "child18",
           "age", "gender", "depres", "comt", "con", "cha",
           "es", "lo", "pro", "th", "diord", "time", "problem",
           "em", "ind", "ru", "av", "hy", "relipart", "rb")
```

As a side note, here we used the name **`ind`** for indirect coping rather than **`in`** because the latter has a special meaning and thus cannot be used as a variable name.

Next, we enter the means and standard deviations reported in the manuscript (in the same order as the variable names):

```
mip.mns <- c(59.18, 19.84, 10.53, 12.65, 9.80, 6.35,
             9.53, 24.96, 19.89, 18.24, 2.68, .45,
             52.04, .11, 9.37, 10.36, 10.35, 9.93,
             30.33, 80.47, 2.67, 1.78, .47, 3.91, 70.58,
             38.87, 24.34, 12.26, 11.80, 9.17, .81, 2.74)

mip.sds <- c(24.24, 9.00, 6.89, 5.22, 3.73, 3.05,
             2.59, 3.87, 6.07, 7.48, 1.07, .81,
            11.04, .318, 6.57, 2.65, 3.27, 2.45,
            5.92, 18.12, .74, 1.14, .50, 8.05, 11.94,
            11.41, 6.83, 7.51, 5.51, 5.94, 1.24, .92)
```

Finally, we read in the correlations from the file **`mip.dat`** using the **`read.table`** function and convert the correlation matrix to a covariance matrix with the following commands:

```
mip.cor <- read.table("mip.dat", header=FALSE, row.names=names, col.names=names)
mip.cor <- data.matrix(mip.cor)
mip.cov <- cor2cov(mip.cor,sds=mip.sds)
```

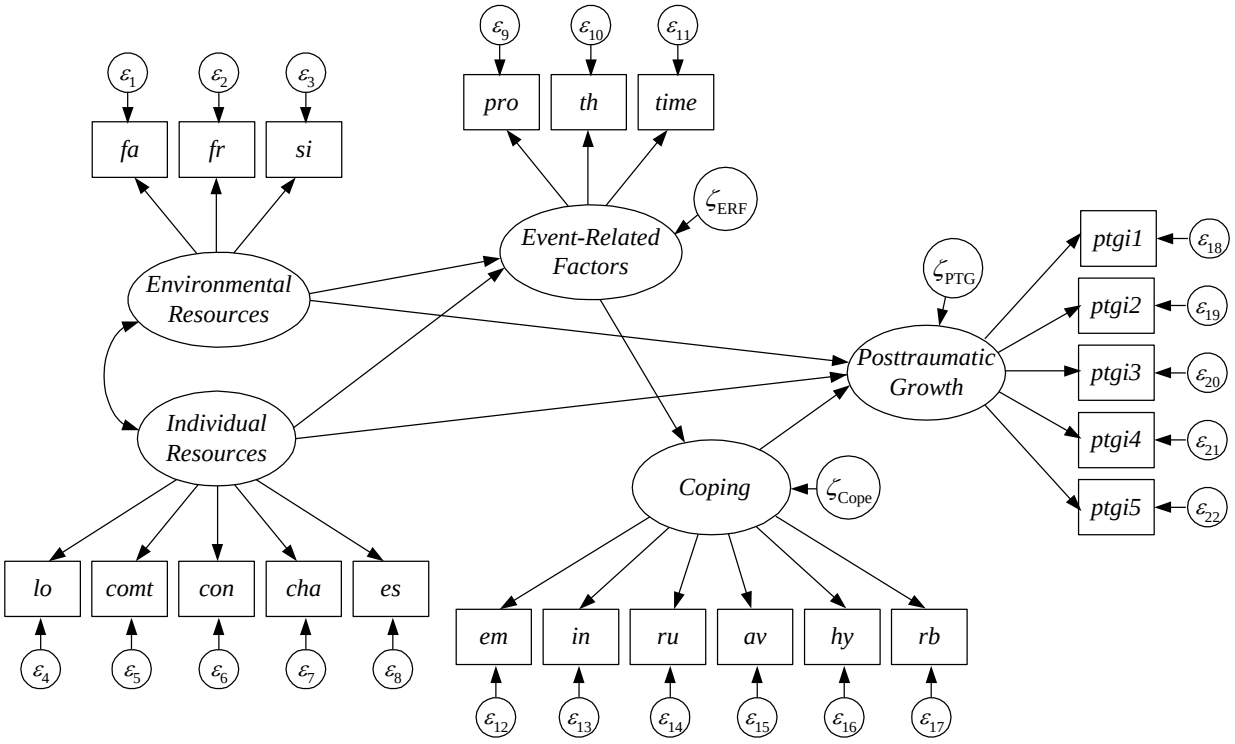
The **`mip.dat`** file is a space-delimited text file that contains a full correlation matrix (diagonal of 1's as well as all elements above and below the diagonal). The **`read.table`** function creates a data frame from this file called **`mip.cor`** with rows and columns labeled by the variable names we input previously. We then convert **`mip.cor`** to a numeric matrix using the **`data.matrix`** function. Finally, we use the **`cor2cov`** function of lavaan to convert the correlation matrix into a covariance matrix, utilizing the information we provided on the standard deviations in the vector **`mip.sds`**.

Note that it is also possible to read in a lower triangular correlation matrix and convert it to a full covariance matrix using the **`getCov`** function in lavaan. Here, however, the data was originally entered as a full correlation matrix.

We now have the sufficient statistics, the covariance matrix **`mip.cov`** and the mean vector **`mip.mns`**, ready for use with lavaan. We can thus proceed to specify and fit the hypothesized structural equation model.

Initial Hypothesized Model

The hypothesized model for the data predicts that both individual and environmental resources directly lead to increased posttraumatic growth, but also indirectly lead to somewhat decreased posttraumatic growth by reducing event-related hardship, thus decreasing the need for coping and reducing opportunities for posttraumatic growth. The hypothesized model also predicts that neither environmental nor individual resources have a direct impact on cognitive coping. Further, the effect of event-related factors on posttraumatic growth is hypothesized to be purely mediated by coping.



We can also express the model using matrix algebra, as shown on the next page.

The measurement model is:

$$\begin{bmatrix} fa_i \\ fr_i \\ si_i \\ lo_i \\ comt_i \\ con_i \\ cha_i \\ es_i \\ pro_i \\ th_i \\ time_i \\ em_i \\ in_i \\ ru_i \\ av_i \\ hy_i \\ rb_i \\ ptgi_{1i} \\ ptgi_{2i} \\ ptgi_{3i} \\ ptgi_{4i} \\ ptgi_{5i} \end{bmatrix} = \begin{bmatrix} \nu_{1i} \\ \nu_{2i} \\ \nu_{3i} \\ \nu_{4i} \\ \nu_{5i} \\ \nu_{6i} \\ \nu_{7i} \\ \nu_{8i} \\ \nu_{9i} \\ \nu_{10i} \\ \nu_{11i} \\ \nu_{12i} \\ \nu_{13i} \\ \nu_{14i} \\ \nu_{15i} \\ \nu_{16i} \\ \nu_{17i} \\ \nu_{18i} \\ \nu_{19i} \\ \nu_{20i} \\ \nu_{21i} \\ \nu_{22i} \end{bmatrix} + \begin{bmatrix} \lambda_{11} & 0 & 0 & 0 & 0 \\ \lambda_{21} & 0 & 0 & 0 & 0 \\ \lambda_{31} & 0 & 0 & 0 & 0 \\ 0 & \lambda_{42} & 0 & 0 & 0 \\ 0 & \lambda_{52} & 0 & 0 & 0 \\ 0 & \lambda_{62} & 0 & 0 & 0 \\ 0 & \lambda_{72} & 0 & 0 & 0 \\ 0 & \lambda_{82} & 0 & 0 & 0 \\ 0 & 0 & \lambda_{93} & 0 & 0 \\ 0 & 0 & \lambda_{10,3} & 0 & 0 \\ 0 & 0 & \lambda_{11,3} & 0 & 0 \\ 0 & 0 & 0 & \lambda_{12,4} & 0 \\ 0 & 0 & 0 & \lambda_{13,4} & 0 \\ 0 & 0 & 0 & \lambda_{14,4} & 0 \\ 0 & 0 & 0 & \lambda_{15,4} & 0 \\ 0 & 0 & 0 & \lambda_{16,4} & 0 \\ 0 & 0 & 0 & \lambda_{17,4} & 0 \\ 0 & 0 & 0 & 0 & \lambda_{18,5} \\ 0 & 0 & 0 & 0 & \lambda_{19,5} \\ 0 & 0 & 0 & 0 & \lambda_{20,5} \\ 0 & 0 & 0 & 0 & \lambda_{21,5} \\ 0 & 0 & 0 & 0 & \lambda_{22,5} \end{bmatrix} \begin{bmatrix} \eta_{ERi} \\ \eta_{IRi} \\ \eta_{ERFi} \\ \eta_{COPE_i} \\ \eta_{PTGi} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \\ \varepsilon_{10i} \\ \varepsilon_{11i} \\ \varepsilon_{12i} \\ \varepsilon_{13i} \\ \varepsilon_{14i} \\ \varepsilon_{15i} \\ \varepsilon_{16i} \\ \varepsilon_{17i} \\ \varepsilon_{18i} \\ \varepsilon_{19i} \\ \varepsilon_{20i} \\ \varepsilon_{21i} \\ \varepsilon_{22i} \end{bmatrix}$$

where $\boldsymbol{\Theta} = DIAG\left(\theta_{11}, \theta_{22}, \ldots, \theta_{22,22}\right)$

The latent variable model is:

$$\begin{bmatrix} \eta_{ERi} \\ \eta_{IRi} \\ \eta_{ERFi} \\ \eta_{COPE_i} \\ \eta_{PTGi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \beta_{31} & \beta_{32} & 0 & 0 & 0 \\ 0 & 0 & \beta_{43} & 0 & 0 \\ \beta_{51} & \beta_{52} & 0 & \beta_{54} & 0 \end{bmatrix} \begin{bmatrix} \eta_{ERi} \\ \eta_{IRi} \\ \eta_{ERFi} \\ \eta_{COPE_i} \\ \eta_{PTGi} \end{bmatrix} + \begin{bmatrix} \zeta_{ERi} \\ \zeta_{IRi} \\ \zeta_{ERFi} \\ \zeta_{COPE_i} \\ \zeta_{PTGi} \end{bmatrix}$$

where $\boldsymbol{\Psi} = \begin{bmatrix} 1 & & & & \\ \psi_{21} & 1 & & & \\ 0 & 0 & 1 & & \\ 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Note that the means/intercepts and (residual) variances of the factors have been fixed to 0 and 1, respectively to scale the latent variables. In class, we used the *t*-rule and the two-step rule to verify that the model is identified. We can thus go on to specify the model in lavaan.

Here we specify the model syntax object for the hypothesized SEM and fit the model.

```
mod.1 <-'#specifying measurement model portion
         ER =~ fa + fr + si
         IR =~ comt + con + cha + es + lo
         ERF =~ pro + th + time
         CPP =~ em + ind + ru + av + hy + rb
         PTG =~ ptgi1 + ptgi2 + ptgi3 + ptgi4 + ptgi5

         #specifying structural model portion
         ERF ~ ER + IR
         CPP ~ ERF
         PTG ~ ER + IR + CPP
        '

fit.1 <- sem(mod.1, sample.cov=mip.cov, sample.mean=mip.mns, sample.nobs=132,
             meanstructure=TRUE, std.lv=TRUE)
summary(fit.1, fit.measures=TRUE, estimates=FALSE)
```

We have seen most of these commands before, so here we will highlight only portions of the code.

Lavaan accepts either raw data or summary data (i.e., means, standard deviations, and correlations among variables). As a default, lavaan assumes that data are in raw format. Since the data from this example are in summary form, we used the **sample.cov**, **sample.mean**, and **sample.nobs** arguments of the **sem** function, indicating the covariance matrix, mean vector, and number of observations in the data, respectively.

As in Chapter 4, the measurement model for the latent variables is specified using the **=~** operator. As in Chapters 1-3, regression equations are specified via the **~** operator, the only difference here being that we are referencing the latent variables.

To set the scale of the latent factors, we have standardized the factor (residual) variances by using **std.lv=TRUE** in the **sem** function call.

Within the **summary** function, we requested only the fit measures. We will examine the model estimates only after obtaining a model that we deem to fit the data reasonably well.

Let us now turn to the fit indices for the model:

```
lavaan 0.6-2 ended normally after 53 iterations

  Optimization method                           NLMINB
  Number of free parameters                         73

  Number of observations                           132

  Estimator                                         ML
```

```
  Model Fit Test Statistic                        350.000
  Degrees of freedom                                  202
  P-value (Chi-square)                              0.000

Model test baseline model:

  Minimum Function Test Statistic                1205.231
  Degrees of freedom                                  231
  P-value                                           0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                       0.848
  Tucker-Lewis Index (TLI)                          0.826

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)                 -8012.004
  Loglikelihood unrestricted model (H1)         -7837.004

  Number of free parameters                            73
  Akaike (AIC)                                  16170.009
  Bayesian (BIC)                                16380.453
  Sample-size adjusted Bayesian (BIC)           16149.553

Root Mean Square Error of Approximation:

  RMSEA                                             0.075
  90 Percent Confidence Interval          0.061  0.087
  P-value RMSEA <= 0.05                             0.002

Standardized Root Mean Square Residual:

  SRMR                                              0.101
```

The fit indices indicate that the model does not fit the data well. Rather than examining the parameter estimates, we will next look at the modification indices to get a sense for what might be causing the model to fit poorly, keeping in mind that any model modification must be theoretically justifiable.

```
modindices(fit.1, sort.=TRUE, minimum.value=10)
```

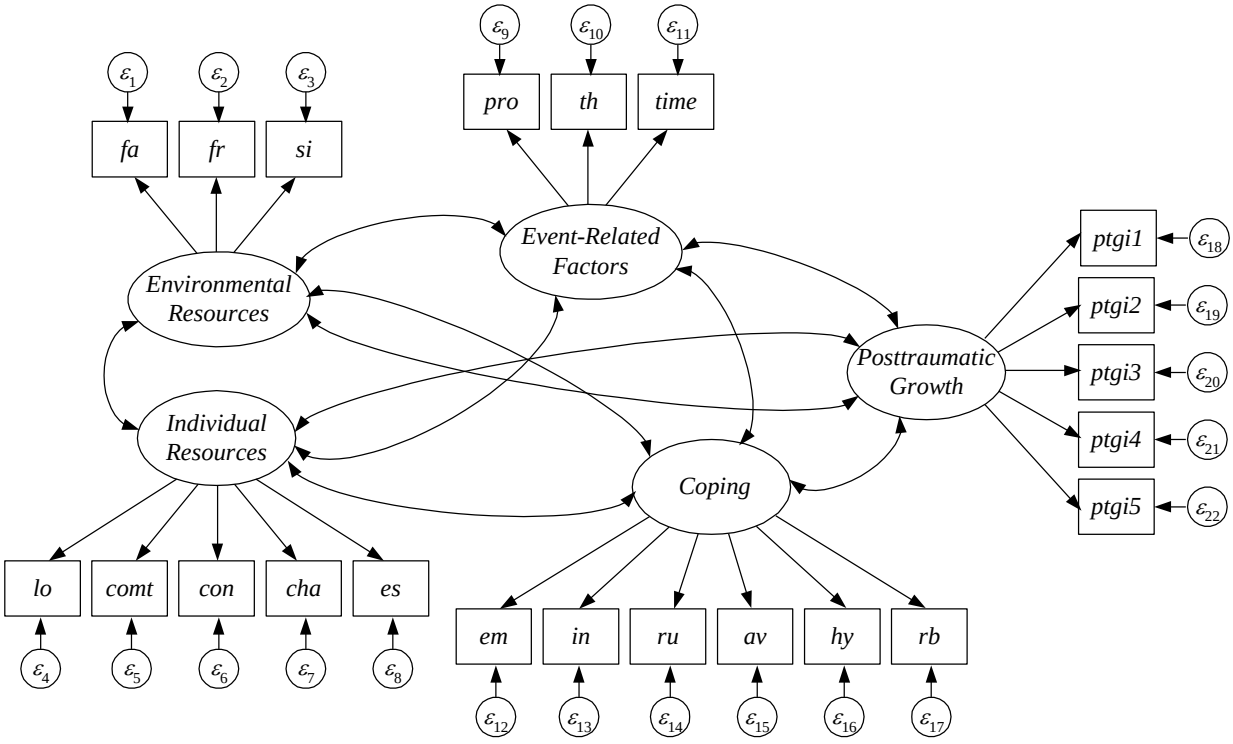
```
     lhs op   rhs     mi     epc sepc.lv sepc.all sepc.nox
368   ru ~~    hy 32.238  34.381  34.381    4.461    4.461
233 comt ~~   cha 27.844   2.684   2.684    0.546    0.546
348   em ~~   ind 21.340 -26.655 -26.655   -0.410   -0.410
183   fa ~~   ind 14.250  -7.574  -7.574   -0.333   -0.333
250  con ~~   cha 12.343  -2.215  -2.215   -0.381   -0.381
119   IR =~ ptgi5 11.877  -0.730  -0.730   -0.241   -0.241
256  con ~~    em 10.695  -8.717  -8.717   -0.304   -0.304
160  PTG =~   cha 10.321   0.590   0.688    0.282    0.282
```

Modification indices suggest that the largest improvement to the model chi-square could be achieved by allowing hypervigilance to correlate with rumination, over and above the correlation implied by the coping factor, allowing challenge to correlate with commitment over and above the individual resources factor, and allowing indirect and emotional coping to correlate above and beyond the correlation implied by the coping factor. These modifications reflect misspecification in the measurement model.

Confirmatory Factor Analysis

When building a structural equation model, a useful strategy to isolate misspecification is to begin by ensuring that the foundation of the overall model, the measurement model, is correctly specified. Once the measurement model has been properly specified, one can consider misfit introduced by the structural model. Thus, we turn next to a CFA with saturated covariances among factors. This strategy will allow us to get measurement right so that measurement misspecification is not confounded with structural misfit.

The CFA model is shown below.



The corresponding script to specify and fit this model is

```
mod.2 <-'#specifying measurement model
         ER =~ fa + fr + si
         IR =~ comt + con + cha + es + lo
         ERF =~ pro + th + time
         CPP =~ em + ind + ru + av + hy + rb
         PTG =~ ptgi1 + ptgi2 + ptgi3 + ptgi4 + ptgi5
        '

fit.2 <- sem(mod.2, sample.cov=mip.cov, sample.mean=mip.mns, sample.nobs=132,
             meanstructure=TRUE, std.lv=TRUE)
summary(fit.2, fit.measures=TRUE, estimates=FALSE)
```

The model fit information is shown below:

```
lavaan 0.6-2 ended normally after 53 iterations

  Optimization method                           NLMINB
  Number of free parameters                         76

  Number of observations                           132

  Estimator                                         ML
  Model Fit Test Statistic                     348.636
  Degrees of freedom                               199
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic             1205.231
  Degrees of freedom                               231
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.846
  Tucker-Lewis Index (TLI)                       0.822

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -8011.322
  Loglikelihood unrestricted model (H1)      -7837.004

  Number of free parameters                         76
  Akaike (AIC)                               16174.645
  Bayesian (BIC)                             16393.738
  Sample-size adjusted Bayesian (BIC)        16153.348
```

```
Root Mean Square Error of Approximation:

  RMSEA                                          0.075
  90 Percent Confidence Interval          0.062  0.088
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.099
```

The model still does not fit the data well, confirming our hypothesis that the measurement model, and not the structural model, is misspecified. Indeed, we can conduct a likelihood ratio test comparing the CFA model with the originally hypothesized SEM because the hypothesized model is a constrained version of the CFA with three structural parameters fixed to zero.

```
lavTestLRT(fit.2, fit.1)
```

```
Chi Square Difference Test

       Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.2 199 16175 16394 348.64
fit.1 202 16170 16380 350.00     1.3639       3      0.714
```

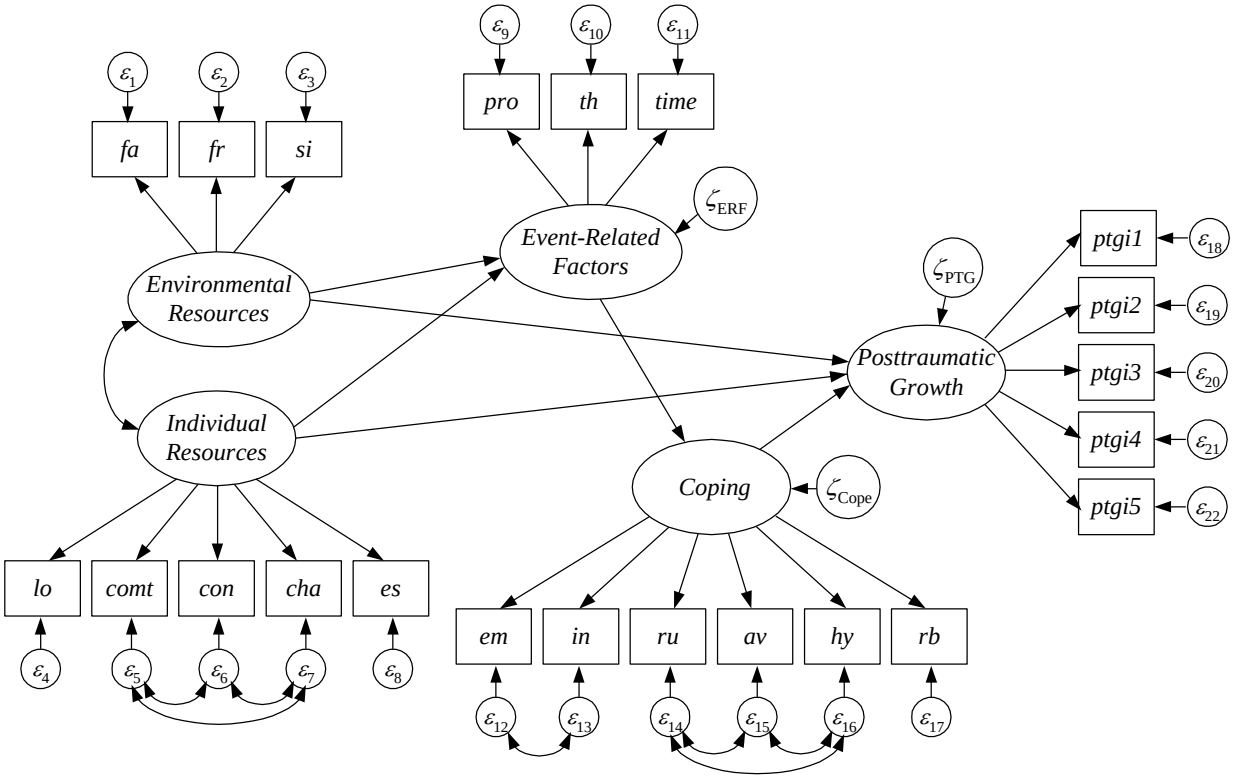
Thus, there is no significant decrement in fit associated with restricting the relationships among the latent variables:

$$\Delta\chi^2(3) = 350.00 - 348.64 = 1.36,\ p = .71$$

Combining this information with the information provided earlier from the modification indices, we can conclude that the measurement model is the primary source of misfit and requires respecification. Theoretically, allowing some residuals to correlate (as suggested by the MIs) makes sense because some factors include multiple subscale scores as indicators. When combined with other items from different scales, we would expect some degree of local dependence for subscales from a common measure. Specifically, the **IR** factor includes three Psychological Hardiness subscale scores as indicators (**comt**, **con**, and **cha**), but also two indicators from independent scales (**lo** and **es**). The coping factor includes two indicators from the Ways of Coping Inventory (**em** and **ind**), three indicators from the Impact of Event Scale (**ru**, **av**, and **hy**), and a religious beliefs score from another scale (**rb**).

Revised Model

We now introduce correlated uniquenesses for **comt**, **con**, and **cha** on the individual resources factor, between **em** and **ind** on the coping factor, and among **ru**, **av**, and **hy** on the coping factor.



These correlated uniquenesses are included in a new model syntax object:

```
mod.3 <-'#specifying measurement model portion
         ER =~ fa + fr + si
         IR =~ comt + con + cha + es + lo
         ERF =~ pro + th + time
         CPP =~ em + ind + ru + av + hy + rb
         PTG =~ ptgi1 + ptgi2 + ptgi3 + ptgi4 + ptgi5

         #specifying structural model portion
         ERF ~ ER + IR
         CPP ~ ERF
         PTG ~ ER + IR + CPP

         #correlating uniquenesses of same-scale items
         comt ~~ con + cha
         con ~~ cha
         em ~~ ind
         ru ~~ av + hy
         av ~~ hy
        '
```

```
fit.3 <- sem(mod.3, sample.cov=mip.cov, sample.mean=mip.mns, sample.nobs=132,
             meanstructure=TRUE, std.lv=TRUE)
summary(fit.3, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)
```

Lavaan does not use separate names for uniquenesses/residuals/disturbances. Instead, uniquenesses or disturbances are referred to by the referent variable. Thus, the line **comt ~~ con + cha** thus includes a covariance between the uniquenesses of **comt** and **con** and between **comt** and **cha**.

The model fit is shown here:

```
lavaan 0.6-2 ended normally after 138 iterations

  Optimization method                           NLMINB
  Number of free parameters                         80

  Number of observations                           132

  Estimator                                         ML
  Model Fit Test Statistic                     270.278
  Degrees of freedom                               195
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function Test Statistic             1205.231
  Degrees of freedom                               231
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.923
  Tucker-Lewis Index (TLI)                       0.908

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -7972.143
  Loglikelihood unrestricted model (H1)      -7837.004

  Number of free parameters                         80
  Akaike (AIC)                               16104.287
  Bayesian (BIC)                             16334.911
  Sample-size adjusted Bayesian (BIC)        16081.869

Root Mean Square Error of Approximation:

  RMSEA                                          0.054
  90 Percent Confidence Interval          0.037  0.069
  P-value RMSEA <= 0.05                          0.325
```

```
Standardized Root Mean Square Residual:

  SRMR                                             0.090
```

We can evaluate the improvement in fit associated with adding these 7 new free parameters to the hypothesized model via the chi-square difference (likelihood ratio) test:

```
lavTestLRT(fit.3, fit.1)
```

```
Chi Square Difference Test

       Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.3 195 16104 16335 270.28
fit.1 202 16170 16380 350.00     79.722       7  1.569e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

These results show that the correlated uniquenesses significantly improved the fit of the model:

$$\Delta\chi^2(7) = 350.00 - 270.28 = 79.72,\ p < .001.$$

Other fit indices suggest that the modified model fits the data reasonably well (though not wonderfully).

Raw and fully standardized parameter estimates are presented below.

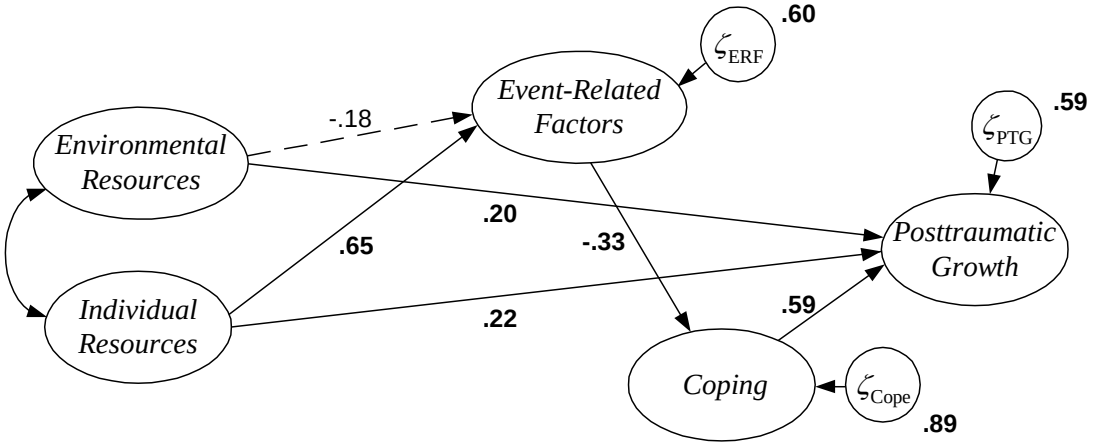
```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  ER =~
    fa                1.317    0.362    3.637    0.000    1.317    0.342
    fr                5.855    0.767    7.634    0.000    5.855    0.968
    si                4.457    0.776    5.740    0.000    4.457    0.598
  IR =~
    comt              1.028    0.290    3.538    0.000    1.028    0.389
    con               2.312    0.325    7.105    0.000    2.312    0.710
    cha               1.141    0.264    4.323    0.000    1.141    0.468
    es                3.007    0.551    5.453    0.000    3.007    0.510
    lo              -11.555    1.675   -6.897    0.000  -11.555   -0.640
  ERF =~
    pro               0.347    0.103    3.365    0.001    0.446    0.606
    th               -0.342    0.114   -3.012    0.003   -0.440   -0.387
    time              1.316    0.743    1.772    0.076    1.693    0.211
  CPP =~
    em                6.060    1.103    5.495    0.000    6.429    0.566
    ind              -3.100    0.668   -4.645    0.000   -3.289   -0.483
    ru                4.173    0.798    5.229    0.000    4.427    0.592
    av                2.958    0.587    5.039    0.000    3.138    0.572
    hy                3.725    0.625    5.962    0.000    3.952    0.668
    rb                0.221    0.088    2.515    0.012    0.235    0.256
```

```
  PTG =~
    ptgi1             6.035    0.633    9.538    0.000    7.837    0.874
    ptgi2             4.578    0.484    9.457    0.000    5.945    0.866
    ptgi3             2.974    0.365    8.146    0.000    3.862    0.743
    ptgi4             2.259    0.261    8.649    0.000    2.933    0.790
    ptgi5             1.879    0.214    8.794    0.000    2.440    0.803

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  ERF ~
    ER               -0.236    0.181   -1.305    0.192   -0.183   -0.183
    IR                0.834    0.293    2.840    0.005    0.648    0.648
  CPP ~
    ERF              -0.276    0.130   -2.120    0.034   -0.334   -0.334
  PTG ~
    ER                0.259    0.118    2.201    0.028    0.200    0.200
    IR                0.284    0.139    2.039    0.041    0.219    0.219
    CPP               0.722    0.168    4.299    0.000    0.590    0.590

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .comt ~~
   .con              -1.601    0.699   -2.292    0.022   -1.601   -0.287
   .cha               2.242    0.601    3.729    0.000    2.242    0.427
 .con ~~
   .cha              -1.923    0.648   -2.967    0.003   -1.923   -0.388
 .em ~~
   .ind             -16.748    6.617   -2.531    0.011  -16.748   -0.300
 .ru ~~
   .av                2.123    3.766    0.564    0.573    2.123    0.078
   .hy               19.248    4.875    3.948    0.000   19.248    0.725
 .av ~~
   .hy               -0.383    2.978   -0.129    0.898   -0.383   -0.019
  ER ~~
    IR                0.244    0.101    2.407    0.016    0.244    0.244

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .fa               24.960    0.336   74.382    0.000   24.960    6.474
   .fr               19.890    0.526   37.791    0.000   19.890    3.289
   .si               18.240    0.649   28.123    0.000   18.240    2.448
   .comt             10.360    0.230   45.087    0.000   10.360    3.924
   .con              10.350    0.284   36.503    0.000   10.350    3.177
   .cha               9.930    0.212   46.744    0.000    9.930    4.069
   .es               30.330    0.513   59.087    0.000   30.330    5.143
   .lo               80.470    1.571   51.217    0.000   80.470    4.458
   .pro               2.670    0.064   41.612    0.000    2.670    3.622
   .th                1.780    0.099   18.008    0.000    1.780    1.567
   .time              3.910    0.698    5.602    0.000    3.910    0.488
   .em               38.870    0.989   39.289    0.000   38.870    3.420
   .ind              24.340    0.592   41.100    0.000   24.340    3.577
```

```
   .ru                12.260    0.651   18.827    0.000   12.260    1.639
   .av                11.800    0.478   24.698    0.000   11.800    2.150
   .hy                 9.170    0.515   17.804    0.000    9.170    1.550
   .rb                 2.740    0.080   34.348    0.000    2.740    2.990
   .ptgi1             19.840    0.780   25.432    0.000   19.840    2.214
   .ptgi2             10.530    0.597   17.632    0.000   10.530    1.535
   .ptgi3             12.650    0.453   27.956    0.000   12.650    2.433
   .ptgi4              9.800    0.323   30.309    0.000    9.800    2.638
   .ptgi5              6.350    0.264   24.018    0.000    6.350    2.091
    ER                 0.000                               0.000    0.000
    IR                 0.000                               0.000    0.000
   .ERF                0.000                               0.000    0.000
   .CPP                0.000                               0.000    0.000
   .PTG                0.000                               0.000    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .fa                13.130    1.669    7.867    0.000   13.130    0.883
   .fr                 2.287    7.782    0.294    0.769    2.287    0.063
   .si                35.663    6.303    5.658    0.000   35.663    0.642
   .comt               5.913    0.825    7.170    0.000    5.913    0.848
   .con                5.269    1.182    4.457    0.000    5.269    0.496
   .cha                4.654    0.694    6.704    0.000    4.654    0.781
   .es                25.739    3.570    7.209    0.000   25.739    0.740
   .lo               192.339   31.799    6.049    0.000  192.339    0.590
   .pro                0.344    0.083    4.156    0.000    0.344    0.633
   .th                 1.096    0.157    6.966    0.000    1.096    0.850
   .time              61.446    7.847    7.830    0.000   61.446    0.955
   .em                87.866   14.001    6.276    0.000   87.866    0.680
   .ind               35.475    5.157    6.879    0.000   35.475    0.766
   .ru                36.371    6.685    5.441    0.000   36.371    0.650
   .av                20.284    3.598    5.638    0.000   20.284    0.673
   .hy                19.397    4.201    4.618    0.000   19.397    0.554
   .rb                 0.785    0.099    7.895    0.000    0.785    0.934
   .ptgi1             18.908    3.311    5.711    0.000   18.908    0.235
   .ptgi2             11.741    1.995    5.886    0.000   11.741    0.249
   .ptgi3             12.112    1.676    7.227    0.000   12.112    0.448
   .ptgi4              5.196    0.752    6.911    0.000    5.196    0.377
   .ptgi5              3.275    0.482    6.789    0.000    3.275    0.355
    ER                 1.000                               1.000    1.000
    IR                 1.000                               1.000    1.000
   .ERF                1.000                               0.604    0.604
   .CPP                1.000                               0.888    0.888
   .PTG                1.000                               0.593    0.593
```

We focus on the structural parameter estimates in this chapter because interpretation of measurement models has been discussed previously. **IR** has a significant direct effect on **ERF** ($\gamma = .834$; S.E. = .293; *p* = .005) and **PTG** ($\gamma = .284$; S.E. = .139; *p* = .041). **ER** has a significant direct effect on **PTG** ($\gamma = .259$; S.E. = .118; *p* = .28) and **coping** ($\gamma = -.276$; S.E. = .130; *p* = .034). **Coping** is significantly related to **PTG** ($\gamma = .722$; S.E. = .168 *p* < .001).

The standardized structural parameter estimates have been drawn on the path diagram below to enhance interpretation of the model results. The non-significant path from environmental resources to event-related factors is dashed. All other paths are statistically significant and shown with solid lines.



We can also examine the R-square statistics to get a sense of the magnitude of these effects:

```
R-Square:
                  Estimate
    fa               0.117
    fr               0.937
    si               0.358
    comt             0.152
    con              0.504
    cha              0.219
    es               0.260
    lo               0.410
    pro              0.367
    th               0.150
    time             0.045
    em               0.320
    ind              0.234
    ru               0.350
    av               0.327
    hy               0.446
    rb               0.066
    ptgi1            0.765
    ptgi2            0.751
    ptgi3            0.552
    ptgi4            0.623
    ptgi5            0.645
    ERF              0.396
    CPP              0.112
    PTG              0.407
```

Together, these results suggest that environmental and individual resources have a moderate, direct, positive influence on posttraumatic growth, cognitive coping has a strong, direct, positive influence on posttraumatic growth, individual resources strongly predict more event-related factors (shorter time since prognosis, poorer prognosis, and greater threat), and more positive event-related factors predicts moderately less cognitive coping. Individual resources and event related factors have a complex relationship with posttraumatic growth. To better understand these relationships, we must consider direct, indirect, and total effects.

Examining Direct, Indirect and Total Effects

As in Chapter 3, to compute and test indirect and total effects, we need to modify the model syntax object to include parameter labels for the paths involved in the effects so that we can define the effects of interest. Here we will focus on the effects on post-traumatic growth.

```
mod.3b <-'#specifying measurement model portion
          ER =~ fa + fr + si
          IR =~ comt + con + cha + es + lo
          ERF =~ pro + th + time
          CPP =~ em + ind + ru + av + hy + rb
          PTG =~ ptgi1 + ptgi2 + ptgi3 + ptgi4 + ptgi5

          #specifying structural model portion
          ERF ~ a1*ER + a2*IR
          CPP ~ b*ERF
          PTG ~  d1*ER + d2*IR + c*CPP

          #correlating uniquenesses of same-scale items
          comt ~~ con + cha
          con ~~ cha
          em ~~ ind
          ru ~~ av + hy
          av ~~ hy

          #effects of ERF
          dir_ERF := 0
          ind_ERF := b*c
          tot_ERF := 0 + b*c

          #effects of ER
          dir_ER := d1
          ind_ER := a1*b*c
          tot_ER := d1 + a1*b*c

          #effects of IR
          dir_IR := d2
          ind_IR := a2*b*c
          tot_IR := d2 + a2*b*c
         '
fit.3b <- sem(mod.3b, sample.cov=mip.cov, sample.mean=mip.mns,
              sample.nobs=132, meanstructure=TRUE, std.lv=TRUE)
summary(fit.3b, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)
```

We will thus obtain direct, indirect and total effect estimates of **ERF**, **IR**, and **ER** on **PTG**.

Most of the results obtained by fitting this model are identical to those presented previously, so only the new information associated with the effect decomposition is shown below:

```
Defined Parameters:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    dir_ERF           0.000                               0.000    0.000
    ind_ERF          -0.199    0.100   -1.982    0.047   -0.197   -0.197
    tot_ERF          -0.199    0.100   -1.982    0.047   -0.197   -0.197
    dir_ER            0.259    0.118    2.201    0.028    0.200    0.200
    ind_ER            0.047    0.041    1.152    0.249    0.036    0.036
    tot_ER            0.306    0.123    2.494    0.013    0.236    0.236
    dir_IR            0.284    0.139    2.039    0.041    0.219    0.219
    ind_IR           -0.166    0.091   -1.833    0.067   -0.128   -0.128
    tot_IR            0.118    0.132    0.893    0.372    0.091    0.091
```

For each predictor-to-outcome effect, we have directed lavaan to first compute the direct effect, then the indirect effect, and finally the total effect (along with standard errors and significance tests). Standard errors are computed using the delta method (producing what is known as the Sobel test for the indirect effects). Unlike in Chapter 3, we cannot use non-parametric bootstrapping to obtain more accurate standard errors because we do not have access to the raw data.

We will closely examine the effect of **IR** on **PTG**. We start by noting that the total effect of **IR** on **PTG** is nonsignificant (p = .372). Upon closer examination, however, it is apparent that **IR** is related to **PTG** both directly and indirectly. These effects are in opposite directions such that the net, total effect is nearly zero. The direct effect of **IR** on **PTG** is positive and significant (p = .041), whereas the indirect effect is negative and marginally significant (p = .067).

Chapter 6
Multiple Groups Models

Multiple group analysis of Holzinger-Swineford Data

The data for this demonstration were provided by Holzinger & Swineford in their 1939 monograph *A Study in Factor Analysis: The Stability of a Bi-Factor Solution*. The sample includes 301 7th and 8th grade students, between 11-16 years of age, drawn from two schools. The data is in the text file **hs.dat** and the accompanying R-script file is **ch06.R**. The variables in the data set that we will use were described in Chapter 4 except for the grouping variable:

school 0 = Pasteur Elementary (*N* = 156); 1 =Grant-White Elementary (*N* = 145)

Please refer to Chapter 4 for a list and description of the other variables used in this demonstration.

Preliminary Steps

As in previous chapters, we begin by reading in the data and assigning variable names:

```
hs <- read.table("hs.dat", header=FALSE)
names(hs) <- c("school", "female", "age", "month",
               "visperc", "cubes", "lozenges",
               "parcomp", "sencomp", "wordmean",
               "addition", "countdot", "sccaps")
```

Then, to put the variables on more similar scales, we do a simple linear transformation of **addition**, **countdot**, and **sccaps** (see Chapter 4):

```
hs$addition <- hs$addition/4
hs$countdot <- hs$countdot/4
hs$sccaps <- hs$sccaps/4
```

Next, to make our results easier to interpret, we will replace the numeric codes for **school** with the school names, as follows:

```
library(plyr)
hs$school <- as.character(hs$school)
hs$school <- revalue(hs$school, c("0"="Pasteur","1"="Grant White"))
```

Note that this requires installation of the **plyr** package. The **revalue** function of the **plyr** package provides an easy way to replace the numeric school codes with the school names. We first convert school from a numeric variable to a character variable using the **as.character** function, and then call the **revalue** function of **plyr** to conditionally assign the school names.

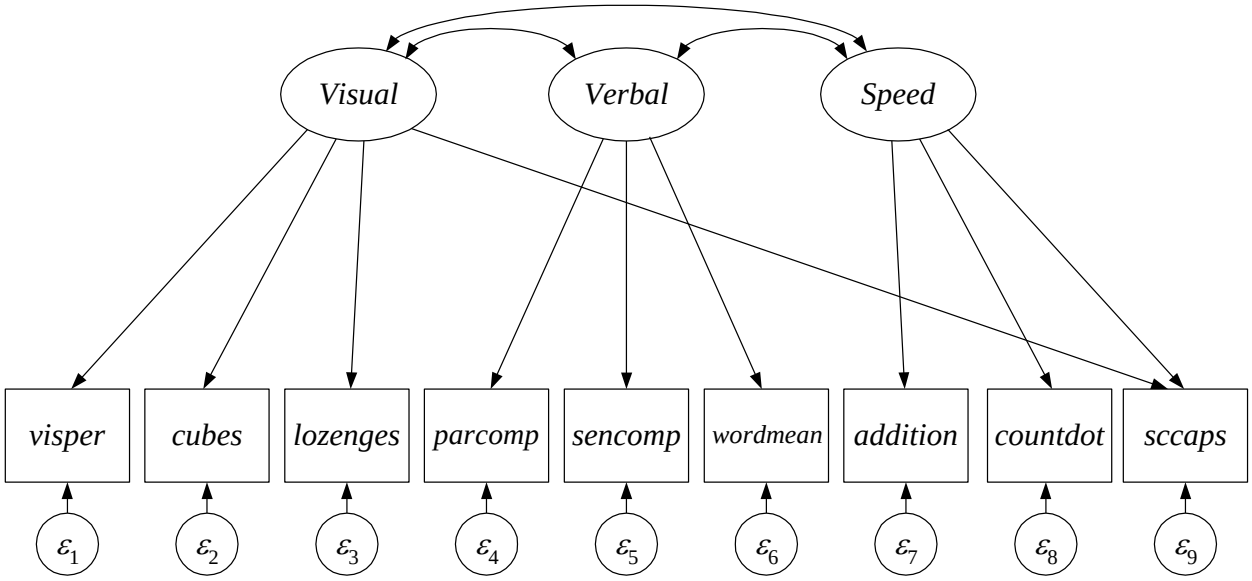
We are now ready to begin fitting the multiple group models with lavaan. To do so, we will load the **lavaan** package. Additionally, invariance testing can be conducted straightforwardly by also loading the **semTools** package (which must first be installed).

```
library(lavaan)
library(semTools)
```

The **semTools** package provides the **measurementInvariance** function, which provides an easy way to conduct likelihood ratio tests between models imposing configural, weak, strong and (optionally) strict invariance.

Evaluating Total Invariance

In Chapter 4, we analyzed the pooled data, ignoring the fact that students were grouped within two schools. We thus implicitly assumed that the same model and set of parameter estimates apply equally to students from the two groups. The final model that we obtained assuming homogeneity across schools is illustrated below:



This model fit adequately well:

$$\chi^2(23) = 52.38,\ p = .0004$$
$$\text{CFI} = .97, \text{TLI} = .95$$
$$\text{RMSEA} = .065, \text{CI}_{90} = (0.042, 0.089)$$
$$\text{SRMR} = .041$$

Because we analyzed pooled data and assessed fit relative to the pooled moments, these fit statistics do not inform us as to whether our assumption of homogeneity across the two schools was reasonable. We can explicitly test whether total invariance (an identical model with identical parameter values) holds across groups by assessing the fit of the model to the group-specific moments. We do this by fitting a multiple groups CFA, constraining all parameter estimates to equality.

A model imposing total invariance can be fit in lavaan by first defining a basic CFA model syntax object as follows:

```
mg.1 <- '#measurement model
         visual =~ visperc + cubes + lozenges + sccaps
         verbal =~ parcomp + sencomp + wordmean
         speed =~ addition + countdot + sccaps
        '
```

The model syntax is identical to that described in Chapter 4.

Next, we will use some new options when we fit the model, as shown here:

```
fit.total <- cfa(mg.1, data=hs, meanstructure=TRUE, std.lv=TRUE,
                 group="school",
                 group.equal=c("loadings","intercepts","residuals",
                               "lv.variances","lv.covariances","means"))
summary(fit.total, fit.measures=TRUE)
```

We have indicated that we wish to conduct a multiple group analysis by using the **group** argument of the fitting function (here **cfa**, but this is also available with the **sem** function). Additionally, we have specified the **group.equal** argument, which is used to impose equality constraints across groups on entire sets of parameter estimates. Here, we have indicated that the factor loadings, intercepts of the indicators, residual variances of the indicators, factor variances, factor covariances and factor means, respectively, should all be held equal. The factor variances are also implicitly held equal over groups by requesting that the factors be standardized with **std.lv=TRUE**, so specifying that they should be equal using **group.equal** too is somewhat redundant. Indeed, upon fitting this model, lavaan outputs the warning:

```
Warning message:
In lav_partable_flat(FLAT, blocks = "group", meanstructure = meanstructure,  :
  lavaan WARNING: std.lv = TRUE forces all variances to be unity in all groups,
despite group.equal = "loadings"
```

In this case, however, this is precisely what we want: We wish to scale the factors to have variances of one in all groups.

Now we can view model fit statistics to determine how well this highly restrictive model fits the data. Note that this is the exact model that was estimated at the end of Chapter 4, except that information about school attended is now included in the likelihood function. Therefore, parameter estimates for this model will be identical to those obtained in Chapter 4 (and are thus not shown here), but model fit will be different, as reported below.

```
lavaan 0.6-2 ended normally after 49 iterations

  Optimization method                           NLMINB
  Number of free parameters                         62
  Number of equality constraints                    31

  Number of observations per group
  Pasteur                                          156
  Grant White                                      145
```

```
  Estimator                                      ML
  Model Fit Test Statistic                  194.023
  Degrees of freedom                             77
  P-value (Chi-square)                        0.000

Chi-square for each group:

  Pasteur                                   103.182
  Grant White                                90.840

Model test baseline model:

  Minimum Function Test Statistic           957.769
  Degrees of freedom                             72
  P-value                                     0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                 0.868
  Tucker-Lewis Index (TLI)                    0.876

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)           -8309.780
  Loglikelihood unrestricted model (H1)   -8212.768

  Number of free parameters                      31
  Akaike (AIC)                            16681.560
  Bayesian (BIC)                          16796.480
  Sample-size adjusted Bayesian (BIC)     16698.166

Root Mean Square Error of Approximation:

  RMSEA                                       0.100
  90 Percent Confidence Interval          0.083  0.118
  P-value RMSEA <= 0.05                       0.000

Standardized Root Mean Square Residual:

  SRMR                                        0.111
```

Including information about school attendance has resulting in a substantial decrement in model fit. Now that school is included as a grouping variable, we can see that the fit of the model is inadequate. (Note that the SRMR here deviates slightly from the SRMR reported from Mplus that was given in the lecture notes but all other fit measures are the same. Evidently the two programs compute SRMR somewhat differently in the multiple group setting).

Evaluating the Level of Invariance

When evaluating a latent variable model with multiple groups, it is important to determine the level of measurement invariance that is present. One way to do this is to start with the least restrictive model (configural invariance) and move to more restrictive models (weak and then strong invariance) as allowed by the data. The **semTools** package provides a function, **measurentInvariance**, which nicely automates this process.

Before implementing this function, however, we need to think about how we wish to scale the latent factors to identify a multiple group factor analysis model. The configural invariance model imposes the least restrictions so we can consider it to establish minimum conditions for identification. Recall that, under configural invariance, the form of the model is equivalent across groups but all of the parameter estimates are allowed to differ except for those that must be set equal simply to identify the model. There are several ways to set the scale of the factors for identification purposes. One could, for instance, standardize the factors in both groups. Another option would be to set the intercept and factor loading for one indicator per factor to 0 and 1 in both groups, respectively. Our preference is instead to standardize the factors in a reference group (here Grant-White) and equate the factor loadings and intercepts for one indicator per factor in each other group (here Pasteur) to the reference group values. This parameterization provides useful interpretations when moving from configural invariance to weak and strong invariance models. (All of these options result in the same model fit, and the chi-square is a simple summation of the chi-squares that would be obtained by fitting the model separately in each group).

We can define this minimally identified model with the model syntax option shown here:

```
mg.2 <- '#loading of first indicator per factor set equal over groups
          visual =~ NA*visperc + c(a,a)*visperc + cubes + lozenges + sccaps
          verbal =~ NA*parcomp + c(b,b)*parcomp + sencomp + wordmean
          speed =~ NA*addition + c(c,c)*addition + countdot + sccaps

          #intercept of first indicator per factor set equal over groups
          visperc ~ c(d,d)*1
          parcomp ~ c(e,e)*1
          addition ~ c(f,f)*1

          #standardizing factor variances in Grant White only
          visual ~~ c(NA,1)*visual
          verbal ~~ c(NA,1)*verbal
          speed ~~ c(NA,1)*speed

          #standardizing factor means in Grant White only
          visual ~ c(NA,0)*1
          verbal ~ c(NA,0)*1
          speed ~ c(NA,0)*1
         '
```

Some aspects of this syntax are new, in part because scaling shortcuts (like **std.lv**) aren't as useful in the multiple group context.

First, notice that, in defining each factor, the first indicator is specified twice, first with the **NA** modifier and next with the **c(label,label)** modifier (e.g., **visual =~ NA*visperc + c(a,a)*visperc**). The **NA** modifier overrides the lavaan default to set the fix the first loading for each factor to one. These loadings will instead be estimated. The **c(label,label)** modifier gives the labels for the factor loading in group 1 (Pasteur) and group 2 (Grant-White), respectively. Since the same label is used for each group's loading, the estimated value will be constrained to be equal across groups. When multiple modifiers are specified for the same coefficient, the modifiers accumulate as lavaan parses the syntax (i.e., they both apply).

Second, we similarly specified that the intercepts of the first item per factor should be held equal, again using the **c(label,label)** modifier (e.g., **visperc ~ c(d,d)*1**).

Third, we standardized the factors in Grant-White only, but estimated factor variances and means in Pasteur. For the factor variances, this is accomplished by using **c(NA,1)** as the modifier (freely estimated for the first group, set to one for the second). In similar fashion, we use **c(NA,0)** as the modifier for the factor means (freely estimated for the first group, set to zero for the second).

By default, all other parameters (intercepts, loadings, residual variances, factor covariances) will differ between groups.

Now that we have specified a minimally identified model syntax object, we can run the **measurementInvariance** function to obtain fit information, likelihood ratio tests, and parameter estimates for the configural, weak, and strong invariance models simultaneously. This is all accomplished with the single line:

```
mi <- measurementInvariance(model=mg.2,data=hs,group="school")
```

(There is also an option to also include strict invariance but, as we discussed in lecture, this is not always useful so we don't illustrate this here.)

We immediately obtain the following output:

```
Measurement invariance models:

Model 1 : fit.configural
Model 2 : fit.loadings
Model 3 : fit.intercepts
Model 4 : fit.means

Chi Square Difference Test

               Df   AIC   BIC   Chisq Chisq diff Df diff Pr(>Chisq)
fit.configural 46 16631 16861  81.547
fit.loadings   53 16625 16829  89.266      7.720       7      0.358
fit.intercepts 59 16654 16836 130.828     41.562       6  2.244e-07 ***
fit.means      62 16688 16859 170.526     39.698       3  1.235e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Fit measures:

                 cfi rmsea cfi.delta rmsea.delta
fit.configural 0.960 0.072        NA          NA
fit.loadings   0.959 0.067     0.001       0.004
fit.intercepts 0.919 0.090     0.040       0.023
fit.means      0.877 0.108     0.041       0.018
```

Four models were fit to the data, of which the first three are of interest as they correspond to configural, weak, and strong invariance models. The last model retains strong invariance but sets the factor means to be equal. While that may be a restriction of interest, rejecting it has no bearing on the level of measurement invariance so we ignore it for now.

For each model, we obtain the chi-square, AIC, BIC, CFI, and RMSEA. We also obtain chi-square difference tests and differences in CFI and RMSEA for each model relative to the prior model in the sequence. Comparing across the models, we can see that the configural invariance model fits reasonably well and that imposing weak invariance does not significantly degrade model fit ($\Delta\chi^2(7) = 7.72, p = .36$). However, imposing strong invariance results in a noticeable deterioration of model fit, as reflected both in a significant chi-square difference test relative to the weak invariance model ($\Delta\chi^2(6) = 41.56, p < .0001$) and poorer CFI and RMSEA values.

To see more detailed information on these models, we can use the **summary** function, referencing the specific fit object of interest, as shown here (results not shown):

```
summary(mi$fit.configural)
summary(mi$fit.loadings)
summary(mi$fit.intercepts)
```

Having established weak invariance, we can safely compare factor variances and covariances across groups. However, if only weak invariance holds then we cannot also compare factor means. Although the strong invariance model was rejected, we may still be able to satisfy partial strong invariance if only a small number of items require different intercepts across the two groups. We will use modification indices to identify these items.

Identifying and Allowing for Partial Strong Invariance

In prior chapters, we used the **modIndices** function to obtain modification indices. To obtain modification indices for equality constraints, however, we must use the **lavTestScore** function (score test is another name for modification index), as shown:

```
lavTestScore(mi$fit.intercepts)
```

Running this syntax produces the results shown here:

```
univariate score tests:

     lhs op   rhs     X2 df p.value
1   .p1. == .p38.  0.828  1   0.363
2   .p2. == .p39.  0.272  1   0.602
3   .p3. == .p40.  0.223  1   0.637
4   .p4. == .p41.  1.846  1   0.174
5   .p5. == .p42.  1.050  1   0.306
6   .p6. == .p43.  6.250  1   0.012
7   .p7. == .p44.  2.393  1   0.122
8   .p8. == .p45.  2.947  1   0.086
9   .p9. == .p46.  1.993  1   0.158
10 .p10. == .p47.  0.149  1   0.700
11 .p11. == .p48.  3.117  1   0.077
12 .p12. == .p49.  2.140  1   0.144
13 .p13. == .p50. 16.399  1   0.000
14 .p32. == .p69.  6.197  1   0.013
15 .p33. == .p70. 20.623  1   0.000
16 .p34. == .p71.  1.530  1   0.216
```

We can see that there are two large MIs, on lines 13 and 15 of the output. These involve equality constraints for parameter 13 (**.p13.**) with 50 (**.p50.**) and parameter 33 (**.p33.**) with 70 (**.p70.**). We can clarify which parameters these are by running the following commands:

```
param<-parTable(mi$fit.intercepts)
param[c(13,50,33,70),]
```

The first line puts all the parameters from the strong invariance model into the object **param**, and then the second command prints out only the 13th, 50th, 33rd, and 70th rows of **param**, in that order, corresponding to the parameters associated with the largest MIs.

We obtain the following results:

```
   id      lhs op rhs user block group free ustart exo label plabel  start    est    se
13 13 addition ~1        1     1     1   13     NA   0     f  .p13. 25.486 23.295 0.472
50 50 addition ~1        1     2     2   50     NA   0     f  .p50. 22.545 23.295 0.472
33 33 lozenges ~1        0     1     1   33     NA   0 .p33.  .p33. 19.897 17.323 0.636
70 70 lozenges ~1        0     2     2   64     NA   0 .p33.  .p70. 15.966 17.323 0.636
```

Thus we see that the MI of 16.399 for **.p13. == .p50.** is for the equality constraint on the intercept of **addition**. Likewise, the MI of 20.623 for **.p33. == .p70.** is for the equality constraint on the intercept of **lozenges**. These two items appear to be largely responsible for the lack of fit of the strong invariance model.

To fit the model with partial invariance, we first define a new model syntax object, as follows:

```
mg.3 <- '#loading of first indicator per factor set equal over groups
          visual =~ NA*visperc + cubes + lozenges + sccaps
          verbal =~ NA*parcomp + sencomp + wordmean
          speed =~ NA*addition + countdot + sccaps

          #standardizing factor variances in Grant White only
          visual ~~ c(NA,1)*visual
          verbal ~~ c(NA,1)*verbal
          speed ~~ c(NA,1)*speed

          #standardizing factor means in Grant White only
          visual ~ c(NA,0)*1
          verbal ~ c(NA,0)*1
          speed ~ c(NA,0)*1
         '
```

It is very similar to the prior model syntax object, **mg.2**, except that we have removed the equality constraints on the intercepts and loadings of the first indicator per factor. This is necessary here because we need to allow the intercepts of **addition** to differ and this is the first item on the **speed** factor. The model will instead be identified through equality constraints imposed via the **group.equal** argument of the fitting function, as shown here:

```
partstr <- cfa(mg.3, data=hs, meanstructure=TRUE, group="school",
               group.equal=c("loadings","intercepts"),
               group.partial=c("lozenges ~1","addition ~1"))
summary(partstr, fit.measures=TRUE, standardized=TRUE)
```

We have seen the **group.equal** argument before (in the total invariance model), which we use here to set all loadings and intercepts equal over groups. Alone, this would impose full strong invariance but we can also implement the **group.partial** argument to remove specific parameters from these equality constraints, allowing for partial invariance. Specifically, here we designate **group.partial=c("lozenges ~1","addition ~1")** to allow the intercepts for just **lozenges** and **addition** to differ over groups.

Model fit is shown below:

```
lavaan 0.6-2 ended normally after 87 iterations

  Optimization method                           NLMINB
  Number of free parameters                         68
  Number of equality constraints                    17

  Number of observations per group
  Pasteur                                          156
  Grant White                                      145

  Estimator                                         ML
  Model Fit Test Statistic                      95.064
  Degrees of freedom                                57
  P-value (Chi-square)                           0.001
```

```
Chi-square for each group:

  Pasteur                                    59.744
  Grant White                                35.320

Model test baseline model:

  Minimum Function Test Statistic           957.769
  Degrees of freedom                             72
  P-value                                     0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                 0.957
  Tucker-Lewis Index (TLI)                    0.946

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)           -8260.301
  Loglikelihood unrestricted model (H1)   -8212.768

  Number of free parameters                      51
  Akaike (AIC)                            16622.601
  Bayesian (BIC)                          16811.664
  Sample-size adjusted Bayesian (BIC)     16649.921

Root Mean Square Error of Approximation:

  RMSEA                                       0.067
  90 Percent Confidence Interval          0.042  0.090
  P-value RMSEA <= 0.05                       0.123

Standardized Root Mean Square Residual:

  SRMR                                        0.057
```

The fit for this model is adequate. Notice also that lavaan has divided the overall model chi-square into the portion associated with each school (which sum to the total).

We can also compare the fit of the partial strong invariance model to the less restricted weak invariance model:

```
lavTestLRT(partstr, mi$fit.loadings)
```

```
Chi Square Difference Test

                Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
mi$fit.loadings 53 16625 16829 89.266
partstr         57 16623 16812 95.064     5.7981       4     0.2147
```

We see that the fit for the partial strong invariance model is not significantly worse than the fit for the weak invariance model ($\Delta\chi^2(4) = 5.80, p = .21$).

Additionally, we can evaluate the improvement in fit of the partial strong invariance model relative to the more restricted full strong invariance model:

```
lavTestLRT(mi$fit.intercepts, partstr)
```

```
Chi Square Difference Test

                  Df   AIC   BIC   Chisq Chisq diff Df diff Pr(>Chisq)
partstr           57 16623 16812  95.064
mi$fit.intercepts 59 16654 16836 130.828     35.764       2  1.714e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Thus the partial strong invariance model also fits the data significantly better than the full strong invariance model ($\Delta\chi^2(2) = 35.76, p < .0001$).

Overall, partial strong invariance is supported. With partial strong invariance, we are able to confidently interpret (co)variance parameters for latent variables. We can also make cautious inferences related to factor means, but the interpretation of factor mean differences is tenuous given that these differences are accurately estimated only if we restricted the intercepts of the truly invariant indicators to equality.

Raw and standardized parameter estimates for the partial strong invariance model are shown here:

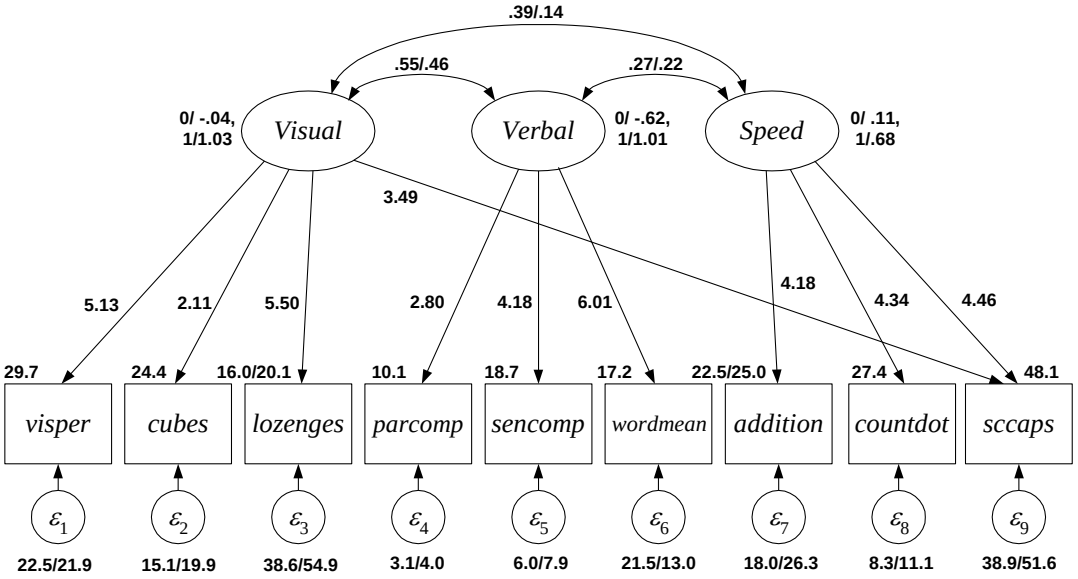
```
Group 1 [Pasteur]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    visperc (.p1.)    5.132    0.539    9.515    0.000    5.206    0.744
    cubes   (.p2.)    2.110    0.322    6.552    0.000    2.140    0.432
    lozengs (.p3.)    5.499    0.627    8.772    0.000    5.577    0.602
    sccaps  (.p4.)    3.487    0.600    5.815    0.000    3.537    0.391
  verbal =~
    parcomp (.p5.)    2.797    0.210   13.320    0.000    2.814    0.814
    sencomp (.p6.)    4.183    0.309   13.520    0.000    4.208    0.831
    wordmen (.p7.)    6.078    0.472   12.866    0.000    6.115    0.862
```

```
  speed =~
    additin (.p8.)    4.180    0.472    8.862    0.000    3.448    0.558
    countdt (.p9.)    4.345    0.427   10.169    0.000    3.584    0.732
    sccaps  (.10.)    4.464    0.666    6.703    0.000    3.682    0.407

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    verbal            0.464    0.124    3.739    0.000    0.454    0.454
    speed             0.136    0.103    1.319    0.187    0.162    0.162
  verbal ~~
    speed             0.220    0.093    2.362    0.018    0.265    0.265

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    visual           -0.037    0.147   -0.253    0.801   -0.037   -0.037
    verbal           -0.618    0.131   -4.733    0.000   -0.614   -0.614
    speed             0.109    0.131    0.837    0.403    0.133    0.133
   .visperc (.29.)   29.715    0.560   53.063    0.000   29.715    4.246
   .cubes   (.30.)   24.447    0.309   79.162    0.000   24.447    4.936
   .lozengs          20.101    0.896   22.429    0.000   20.101    2.168
   .sccaps  (.32.)   48.149    0.700   68.812    0.000   48.149    5.320
   .parcomp (.33.)   10.063    0.264   38.181    0.000   10.063    2.912
   .sencomp (.34.)   18.722    0.390   48.052    0.000   18.722    3.696
   .wordmen (.35.)   17.238    0.587   29.353    0.000   17.238    2.429
   .additin          25.029    0.626   39.961    0.000   25.029    4.051
   .countdt (.37.)   27.397    0.428   63.966    0.000   27.397    5.596

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    visual            1.029    0.247    4.173    0.000    1.000    1.000
    verbal            1.012    0.190    5.335    0.000    1.000    1.000
    speed             0.680    0.166    4.105    0.000    1.000    1.000
   .visperc          21.879    4.583    4.774    0.000   21.879    0.447
   .cubes            19.945    2.455    8.123    0.000   19.945    0.813
   .lozenges         54.860    7.878    6.964    0.000   54.860    0.638
   .sccaps           51.604    7.096    7.272    0.000   51.604    0.630
   .parcomp           4.023    0.625    6.431    0.000    4.023    0.337
   .sencomp           7.946    1.306    6.084    0.000    7.946    0.310
   .wordmean         12.977    2.460    5.276    0.000   12.977    0.258
   .addition         26.285    3.765    6.981    0.000   26.285    0.689
   .countdot         11.124    2.665    4.174    0.000   11.124    0.464


Group 2 [Grant White]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    visperc (.p1.)    5.132    0.539    9.515    0.000    5.132    0.734
    cubes   (.p2.)    2.110    0.322    6.552    0.000    2.110    0.477
```

```
    lozengs (.p3.)    5.499    0.627    8.772    0.000    5.499    0.663
    sccaps  (.p4.)    3.487    0.600    5.815    0.000    3.487    0.382
  verbal =~
    parcomp (.p5.)    2.797    0.210   13.320    0.000    2.797    0.847
    sencomp (.p6.)    4.183    0.309   13.520    0.000    4.183    0.862
    wordmen (.p7.)    6.078    0.472   12.866    0.000    6.078    0.795
  speed =~
    additin (.p8.)    4.180    0.472    8.862    0.000    4.180    0.702
    countdt (.p9.)    4.345    0.427   10.169    0.000    4.345    0.833
    sccaps  (.10.)    4.464    0.666    6.703    0.000    4.464    0.489

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    verbal            0.545    0.082    6.669    0.000    0.545    0.545
    speed             0.393    0.103    3.816    0.000    0.393    0.393
  verbal ~~
    speed             0.268    0.095    2.834    0.005    0.268    0.268

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    visual            0.000                              0.000    0.000
    verbal            0.000                              0.000    0.000
    speed             0.000                              0.000    0.000
   .visperc (.29.)   29.715    0.560   53.063    0.000   29.715    4.252
   .cubes   (.30.)   24.447    0.309   79.162    0.000   24.447    5.526
   .lozengs          15.966    0.689   23.175    0.000   15.966    1.925
   .sccaps  (.32.)   48.149    0.700   68.812    0.000   48.149    5.280
   .parcomp (.33.)   10.063    0.264   38.181    0.000   10.063    3.047
   .sencomp (.34.)   18.722    0.390   48.052    0.000   18.722    3.860
   .wordmen (.35.)   17.238    0.587   29.353    0.000   17.238    2.256
   .additin          22.545    0.495   45.576    0.000   22.545    3.785
   .countdt (.37.)   27.397    0.428   63.966    0.000   27.397    5.254

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    visual            1.000                              1.000    1.000
    verbal            1.000                              1.000    1.000
    speed             1.000                              1.000    1.000
   .visperc          22.501    4.198    5.360    0.000   22.501    0.461
   .cubes            15.122    1.947    7.767    0.000   15.122    0.773
   .lozenges         38.583    6.016    6.413    0.000   38.583    0.561
   .sccaps           38.868    5.875    6.616    0.000   38.868    0.467
   .parcomp           3.080    0.557    5.534    0.000    3.080    0.282
   .sencomp           6.027    1.174    5.135    0.000    6.027    0.256
   .wordmean         21.463    3.266    6.572    0.000   21.463    0.367
   .addition         18.011    3.038    5.928    0.000   18.011    0.508
   .countdot          8.311    2.519    3.299    0.001    8.311    0.306
```

Parameter estimates are listed separately by school. Note that in Chapter 6 of the lecture notes we also present the confidence intervals around the latent factor means. We do not show this

here, but these are easy to obtain. To do this, just include **ci=TRUE** in the summary function, i.e., **summary(partstr, fit.measures=TRUE, ci=TRUE)**. For ease of interpretation, the raw parameter estimates for each school are overlaid on the diagram below:



Interpretations for the factor covariances are discussed first because these are the estimates in which we are most confident. The variances of the **Visual** and **Verbal** factors are nearly identical, but the variance of the **Speed** factor is unexpectedly smaller in the Pasteur group. However, we can tell that this difference is not quite significant because the factor mean and variance estimates for Pasteur are estimated relative to those in the Grant-White group. If a 95% confidence interval constructed for each school contains the fixed value from Grant-White, then the groups are not significantly different.

As expected, the covariance of the **Verbal** with **Visual** is lower in Pasteur than Grant-White, but the covariances with **Speed** are harder to interpret given the difference in the variance for **Speed** across the two groups. In this case, we can instead consider the correlations among the factors (reported with the standardized estimates). The **Verbal** factor is less highly correlated with **Visual** in Pasteur, but is equally correlated with **Speed** across schools. **Speed** is less correlated with **Visual** in Pasteur.

Cautiously moving to group differences in factor means, we see that the mean for the **Verbal** factor is significantly lower in the Pasteur group, but the means of the other two factors are nearly identical. This difference is consistent with our original hypothesis that the performance of Pasteur students on verbal tasks would be worse than that of Grant-White students.

Finally, we can examine group differences in measurement. Holding the **Visual** factor constant, the expected value for **lozenges** is 4.1 units higher in the Pasteur group than the Grant White group. Holding the **Speed** factor constant, the expected value for **addition** is 2.5 unites higher in the Pasteur group than the Grant White group. Thus, students in Pasteur find these tests easier than students of equal ability (with respect to **Visual** and **Speed**) from Grant White.

Chapter 7
SEM with Categorical Indicators

Structural Equation Modeling of Schizophrenia Symptoms

The data for this demonstration are from a manuscript written by Mueser, Curran, and McHugo (1997). The sample includes 437 patients with schizophrenia who were administered the Brief Psychiatric Rating Scale, which consists of 18 item. Items were initially coded on a 7-point scale, but were dichotomized due to extreme sparseness in higher categories. Items 17 and 18 were eliminated because exploratory analysis showed that they do not factor well with the other items. The data file for this example is **`bprs.dat`** and the accompanying R-script file is **`ch07.R`**. The binary variables (0=symptom absent, 1=symptom present) are:

`Item01` somatic concern

`Item02` anxiety

`Item03` emotional withdrawal

`Item04` conceptual disorganization

`Item05` guilt feelings

`Item06` tension

`Item07` mannerisms/posturing

`Item08` grandiosity

`Item09` depressed mood

`Item10` hostility

`Item11` suspiciousness

`Item12` hallucinatory behavior

`Item13` motor retardation

`Item14` uncooperativeness

`Item15` unusual thought content

`Item16` blunted affect

In lecture we described two general approaches to estimating SEMs with discrete dependent variables: a limited information approach called diagonally-weighted least squares (DWLS) and a full information approach called maximum likelihood (ML). There are pros and cons to each, although when both are available, ML is superior from a statistical standpoint because it is asymptotically unbiased and maximally efficient. However, ML does not provide omnibus tests of model fit (e.g., model chi-square or RMSEA) nor does it provide modification indices. In lecture we demonstrated both DWLS and ML for the BPRS data obtained from Mplus. However, although lavaan currently implements a version of ML for categorical endogenous variables, this is still in the early stages of development. We attempted to use this ML estimator for the full four-factor model in this demonstration and were not able to obtain a converged solution, although we were able to obtain estimates for a single factor at a time. Because of this, here we focus solely on the DWLS estimator.

Preliminary Steps

We begin by reading in the data and assigning variable names, as follows:

```
bprs <- read.table("bprs.dat", header=FALSE)
names <- c(paste("item", 1:9, sep = "0"),
           paste("item", 10:18, sep = ""))
names(bprs) <- names
```

Note that because the variable names all begin with the prefix "item" and then are numbered consecutively from 01 to 18, we took a syntax shortcut here with the **paste** function to generate the labels **item01-item18**.

Because the data for this example is a bit different from the others we've considered so far, it's useful to examine the data before proceeding to model specification and fitting. First, we simply view the first 20 rows of data for the first 8 items to get a sense of the data structure:

```
bprs[1:20,1:8]
```

```
   item01 item02 item03 item04 item05 item06 item07 item08
1       1      1      0      0      0      1      0      0
2       1      1      0      1      1      1      1      0
3       0      0      0      1      1      0      0      1
4       0      1      0      0      1      0      0      0
5       0      1      0      0      0      1      0      0
6       1      1      0      1      1      1      1      0
7       1      0      1      1      0      0      0      1
8       1      1      1      1      1      1      0      0
9       1      1      1      1      0      1      1      1
10      1      1      1      1      1      1      0      1
11      0      1      0      0      1      0      0      0
12      0      1      0      0      1      0      1      0
13      1      1      1      0      0      1      1      0
14      1      1      0      0      0      1      0      0
15      0      1      0      0      0      0      0      0
16      1      1      1      1      1      1      0      0
17      1      0      1      1      1      0      0      0
18      0      0      0      0      1      0      0      0
19      1      1      0      1      1      1      1      0
20      0      1      0      0      1      0      0      0
```

We can clearly see that the data file consists entirely of 0's and 1's. Next, we can use the **colMeans** function to see how frequently each item is endorsed (focusing on the 16 items in our model). The mean of each binary item reflects the proportion of cases endorsing that item. Because these are psychiatric symptoms, lower endorsement rates reflect more "severe" items.

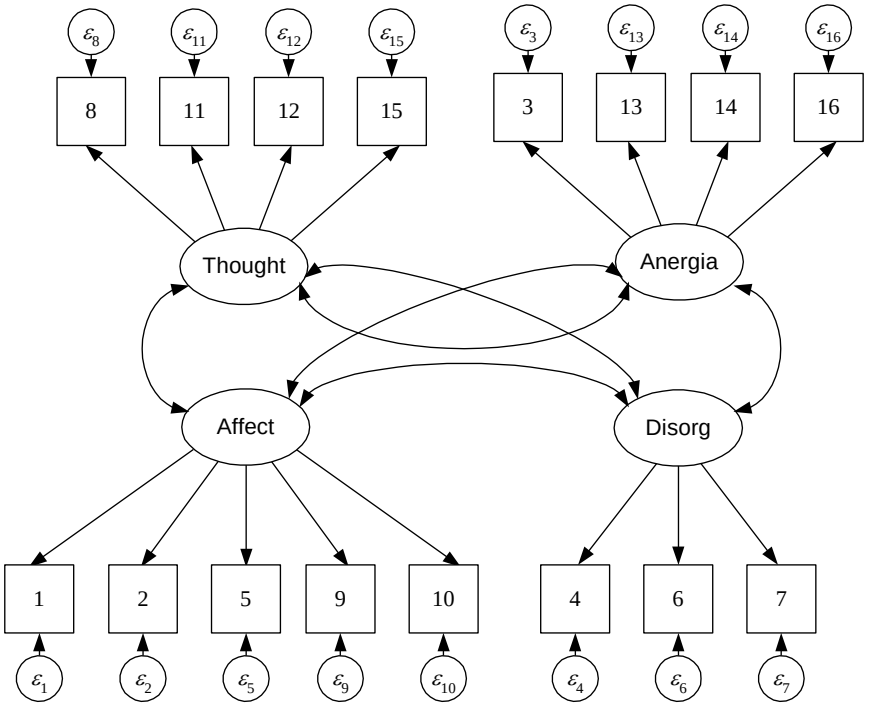
```
colMeans(bprs[1:16])
```

```
   item01    item02    item03    item04    item05    item06    item07    item08
0.5623679 0.7484144 0.5940803 0.5665962 0.3805497 0.5877378 0.3023256 0.3319239
   item09    item10    item11    item12    item13    item14    item15    item16
0.5539112 0.4566596 0.6701903 0.5243129 0.4355180 0.3234672 0.7484144 0.8054968
```

If we were to use our usual normal-theory maximum likelihood estimator for these data, we would be violating a number of cardinal assumptions (i.e., linearity, normality, homoscedasticity). Instead, we will abandon the linear SEM we have considered thus far and move to a wholly new statistical procedure that is explicitly designed for discretely scaled dependent variables.

Confirmatory Factor Analysis using Diagonally Weighted Least Squares Estimation

We begin with the following hypothesized confirmatory factor model:



Using the underlying latent variable conceptualization, the model follows the following form:

$$\mathbf{y}_i^* = \boldsymbol{\nu} + \boldsymbol{\Lambda}\boldsymbol{\eta}_i + \boldsymbol{\varepsilon}_i$$
$$\boldsymbol{\eta}_i = \boldsymbol{\alpha} + \boldsymbol{\zeta}_i$$

where

$$VAR(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta}$$
$$VAR(\boldsymbol{\zeta}_i) = \boldsymbol{\Psi}$$

For each item j, $y_{ij} = 1 \text{ if } y_{ij}^* > \tau_j$.

For identification, we assume that each y^* is standard normal with a mean of 0 and variance of 1. Thus, we only require an estimate of the correlation matrix of the y^*, $\mathbf{S}^*$. This is estimated in two steps: first, item thresholds are estimated using univariate response frequencies; second, polychoric correlations are estimated using bivariate response frequencies. $\mathbf{S}^*$ is constructed

using pairwise polychoric correlation estimates. Recall that to calculate the polychorics, there should be no empty cells in the 2-way bivariate frequency tables, so it is a good idea to check these before proceeding with this approach.

If we assume that all item intercepts are zero, an assumption that is implicit in assuming that the y^* are standard normal, then it is possible to estimate item thresholds. Likewise, we implicitly assume the variance of each y^* is one, so residual variances in Θ are not actually estimated – they are taken as remainders from 1.

As usual, we must identify the scale of the structural latent variables by setting their means and variances to zero and one or by fixing an item on each factor to have an intercept of zero and a factor loading of one. We choose to do the former so that covariances among latent variables are interpretable as correlations and the estimated factor loadings are in a standardized metric.

The lavaan script for specifying and fitting this model is given below:

```
library(lavaan)

cat.1 <- '#measurement model
          Thought =~ item08 + item11 + item12 + item15
          Anergia =~ item03 + item13 + item14 + item16
          Affect =~ item01 + item02 + item05 + item09 + item10
          Disorg =~ item04 + item06 + item07
         '
fit.1 <- cfa(cat.1, data=bprs, std.lv=TRUE, ordered=names[1:16])
summary(fit.1, fit.measures=TRUE, estimates=FALSE)
modindices(fit.1, sort.=TRUE, minimum.value=10)
```

Virtually all of this we have seen before. We have specified the basic measurement model structure in the **cat.1** model syntax object in exactly the same way as was illustrated for continuous indicators in Chapter 4. Similarly, we've requested fit information and modification indices just as in prior examples. The principal difference is that we use the **ordered** argument in the fitting function to tell lavaan that the items are ordered-categorical (in this case, binary). This automatically triggers a switch from ML to DWLS estimation.

It is also worth remembering that neither model fit statistics nor modification indices would be available with full maximum likelihood estimation, so obtaining this information is a key advantage to using DWLS (although at the cost of less efficient estimates for the model parameters).

Model fit is shown below:

```
lavaan 0.6-2 ended normally after 17 iterations

  Optimization method                           NLMINB
  Number of free parameters                         38

  Number of observations                           473

  Estimator                                       DWLS      Robust
  Model Fit Test Statistic                     305.991     331.217
  Degrees of freedom                            98          98
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  0.978
  Shift parameter                                           18.476
    for simple second-order correction (Mplus variant)

Model test baseline model:

  Minimum Function Test Statistic             1986.794    1449.631
  Degrees of freedom                           120         120
  P-value                                        0.000       0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.889       0.825
  Tucker-Lewis Index (TLI)                       0.864       0.785

  Robust Comparative Fit Index (CFI)                            NA
  Robust Tucker-Lewis Index (TLI)                               NA

Root Mean Square Error of Approximation:

  RMSEA                                          0.067       0.071
  90 Percent Confidence Interval          0.059  0.076       0.063  0.079
  P-value RMSEA <= 0.05                          0.001       0.000

  Robust RMSEA                                                  NA
  90 Percent Confidence Interval                                NA     NA

Standardized Root Mean Square Residual:

  SRMR                                           0.107       0.107
```

By default, when using DWLS, lavaan reports both naïve and robust fit statistics. We will interpret the robust fit statistics (from the column “Robust”)

The absolute fit of the model, which we are able to assess using this estimator (but not with ML), is poor for the hypothesized model. (There are a few trivial discrepancies from the corresponding values reported in the lecture notes which were obtained with Mplus, likely due to minor variations in how the robust statistics are calculated).

Given the poor fit, we will view modification indices to see how the model might be improved rather than interpret the current parameter estimates.

```
        lhs op    rhs     mi    epc sepc.lv sepc.all sepc.nox
135  Disorg =~ item13 61.706 -0.460  -0.460   -0.460   -0.460
209  item13 ~~ item16 52.517  1.006   1.006    2.663    2.663
96  Thought =~ item13 50.125 -0.345  -0.345   -0.345   -0.345
124  Affect =~ item13 33.296 -0.301  -0.301   -0.301   -0.301
205  item03 ~~ item04 31.230  0.393   0.393    0.754    0.754
134  Disorg =~ item03 30.432  0.324   0.324    0.324    0.324
216  item13 ~~ item06 30.380 -0.400  -0.400   -0.702   -0.702
116 Anergia =~ item04 29.310  0.394   0.394    0.394    0.394
262  item06 ~~ item07 24.913  0.426   0.426    0.635    0.635
198  item03 ~~ item14 23.859  0.451   0.451    0.732    0.732
147  item08 ~~ item13 22.242 -0.336  -0.336   -0.509   -0.509
260  item04 ~~ item06 21.397 -0.597  -0.597   -1.113   -1.113
95  Thought =~ item03 20.107  0.222   0.222    0.222    0.222
117 Anergia =~ item06 20.046 -0.311  -0.311   -0.311   -0.311
123  Affect =~ item03 18.968  0.230   0.230    0.230    0.230
199  item03 ~~ item16 15.479 -0.615  -0.615   -1.664   -1.664
104 Thought =~ item04 14.749  0.536   0.536    0.536    0.536
155  item08 ~~ item04 14.746  0.274   0.274    0.440    0.440
218  item14 ~~ item16 14.282 -0.487  -0.487   -1.161   -1.161
128  Affect =~ item06 14.194  0.490   0.490    0.490    0.490
106 Thought =~ item07 13.721 -0.376  -0.376   -0.376   -0.376
224  item14 ~~ item04 11.962  0.258   0.258    0.436    0.436
120  Affect =~ item11 11.237  0.400   0.400    0.400    0.400
97  Thought =~ item14 10.223  0.153   0.153    0.153    0.153
```

Many of the highest modification indices involve **item13**. Rather than blindly following the suggestion of the modification indices, we consider why **item13** might want to load on all of the factors in the model and have residual correlations with lots of other items. The content of **item13** concerns motor retardation, a potential side-effect of the psychotropic medications prescribed for schizophrenia. It is therefore not a "clean" measure of latent **anergia**, the construct that we intended to measure with this item. For instance, motor retardation could be a consequence of pharmacological treatment instigated to reduce thought disorder, resulting in a high cross-loading on the thought disorder factor. We will thus remove **item13** from subsequent analyses.

Revised Model with DWLS Estimation

We do not present the equations or path diagram for the revised model because they are virtually identical to the last model, except that **item13** has been omitted. Similarly, the new model syntax object simply omits this item. The script for specifying and fitting the model is shown next. This time, we requested the parameter estimates and R-square measures for the model.

```
cat.2 <- '#measurement model
          Thought =~ item08 + item11 + item12 + item15
          Anergia =~ item03 + item14 + item16
          Affect =~ item01 + item02 + item05 + item09 + item10
          Disorg =~ item04 + item06 + item07
         '
fit.2 <- cfa(cat.2, data=bprs, std.lv=TRUE, ordered=names[1:16])
summary(fit.2, fit.measures=TRUE, rsquare=TRUE)
modindices(fit.2, sort.=TRUE, minimum.value=10)
```

Model fit for the revised model is presented below:

```
lavaan 0.6-2 ended normally after 18 iterations

  Optimization method                           NLMINB
  Number of free parameters                         36

  Number of observations                           473

  Estimator                                       DWLS      Robust
  Model Fit Test Statistic                     156.657     184.102
  Degrees of freedom                                84          84
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  0.916
  Shift parameter                                           13.160
    for simple second-order correction (Mplus variant)

Model test baseline model:

  Minimum Function Test Statistic             1714.722    1275.104
  Degrees of freedom                               105         105
  P-value                                        0.000       0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.955       0.914
  Tucker-Lewis Index (TLI)                       0.944       0.893

Root Mean Square Error of Approximation:

  RMSEA                                          0.043       0.050
  90 Percent Confidence Interval          0.032  0.053       0.040  0.060
  P-value RMSEA <= 0.05                          0.870       0.468

  Robust RMSEA                                                  NA
  90 Percent Confidence Interval                                NA      NA

Standardized Root Mean Square Residual:

  SRMR                                           0.083       0.083
```

Model fit is acceptable, and much improved from the original model.

Modification indices (shown below) do not suggest any theoretically defensible additional changes to our model (e.g., **item04**, conceptual disorganization, does not theoretically belong on an **anergia** factor, and we also don't want to go correlating residuals willy nilly):

```
        lhs op    rhs     mi    epc sepc.lv sepc.all sepc.nox
239  item06 ~~ item07 28.240  0.425   0.425    0.601    0.601
237  item04 ~~ item06 21.782 -0.605  -0.605   -1.160   -1.160
110 Anergia =~ item04 21.353  0.454   0.454    0.454    0.454
121  Affect =~ item06 15.817  0.420   0.420    0.420    0.420
111 Anergia =~ item06 15.103 -0.336  -0.336   -0.336   -0.336
146  item08 ~~ item04 13.379  0.262   0.262    0.451    0.451
114  Affect =~ item11 11.183  0.388   0.388    0.388    0.388
100 Thought =~ item07 10.671 -0.320  -0.320   -0.320   -0.320
```

Thus, we will retain the current model and examine the parameter estimates provided by the DWLS estimator with robust standard errors:

```
Parameter Estimates:

  Information                                 Expected
  Information saturated (h1) model        Unstructured
  Standard Errors                           Robust.sem

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  Thought =~
    item08            0.477    0.062    7.651    0.000
    item11            0.822    0.043   19.318    0.000
    item12            0.706    0.044   15.982    0.000
    item15            0.915    0.041   22.196    0.000
  Anergia =~
    item03            0.918    0.106    8.675    0.000
    item14            0.635    0.085    7.448    0.000
    item16            0.445    0.086    5.172    0.000
  Affect =~
    item01            0.473    0.068    6.974    0.000
    item02            0.693    0.069    9.979    0.000
    item05            0.554    0.063    8.734    0.000
    item09            0.519    0.064    8.152    0.000
    item10            0.576    0.068    8.437    0.000
  Disorg =~
    item04            0.750    0.076    9.922    0.000
    item06            0.615    0.069    8.980    0.000
    item07            0.440    0.071    6.227    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  Thought ~~
    Anergia           0.182    0.074    2.452    0.014
    Affect            0.590    0.065    9.137    0.000
    Disorg            0.518    0.075    6.935    0.000
```

```
  Anergia ~~
    Affect           -0.012    0.085   -0.137    0.891
    Disorg            0.366    0.085    4.332    0.000
  Affect ~~
    Disorg            0.488    0.082    5.942    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .item08            0.000
   .item11            0.000
   .item12            0.000
   .item15            0.000
   .item03            0.000
   .item14            0.000
   .item16            0.000
   .item01            0.000
   .item02            0.000
   .item05            0.000
   .item09            0.000
   .item10            0.000
   .item04            0.000
   .item06            0.000
   .item07            0.000
    Thought           0.000
    Anergia           0.000
    Affect            0.000
    Disorg            0.000

Thresholds:
                   Estimate  Std.Err  z-value  P(>|z|)
    item08|t1         0.435    0.060    7.278    0.000
    item11|t1        -0.440    0.060   -7.369    0.000
    item12|t1        -0.061    0.058   -1.056    0.291
    item15|t1        -0.670    0.063  -10.688    0.000
    item03|t1        -0.238    0.058   -4.084    0.000
    item14|t1         0.458    0.060    7.641    0.000
    item16|t1        -0.861    0.066  -13.016    0.000
    item01|t1        -0.157    0.058   -2.709    0.007
    item02|t1        -0.670    0.063  -10.688    0.000
    item05|t1         0.304    0.059    5.182    0.000
    item09|t1        -0.136    0.058   -2.342    0.019
    item10|t1         0.109    0.058    1.883    0.060
    item04|t1        -0.168    0.058   -2.892    0.004
    item06|t1        -0.222    0.058   -3.809    0.000
    item07|t1         0.518    0.061    8.545    0.000
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .item08            0.772
   .item11            0.324
   .item12            0.501
   .item15            0.162
   .item03            0.157
   .item14            0.597
   .item16            0.802
   .item01            0.776
   .item02            0.519
   .item05            0.693
   .item09            0.730
   .item10            0.668
   .item04            0.438
   .item06            0.621
   .item07            0.806
    Thought           1.000
    Anergia           1.000
    Affect            1.000
    Disorg            1.000

<snip>

R-Square:
                   Estimate
    item08            0.228
    item11            0.676
    item12            0.499
    item15            0.838
    item03            0.843
    item14            0.403
    item16            0.198
    item01            0.224
    item02            0.481
    item05            0.307
    item09            0.270
    item10            0.332
    item04            0.562
    item06            0.379
    item07            0.194
```

Output for a categorical model is similar to the output that we have seen previously for linear models, with two differences. First, because DWLS estimation assumes that the latent distribution underlying each item is standard normal, a threshold has been estimated for each dichotomous item. The threshold estimates how 'difficult' it is for the item to be endorsed. Second, for the same reason, the residual variances aren't estimated, but are remainders from one, and thus have no standard errors or tests.

We also obtain the communalities for the model (in section labeled **R-square**) As with a linear factor model, the variance explained in each item is the sum of squared factor loadings for that item times each factor's variance. Factor variances are all one in this model and each item only loads on a single factor, so the communalities are simply equal to the squared factor loadings.

The obtained estimates indicate that each item has a reasonably-sized factor loading and none of the thresholds are too high or too low (around ± 3, indicating very low marginal endorsement rates in one or the other category of the binary item). The model does a fairly good job at explaining the variance in the fifteen measured variables.

Factor correlations indicate that **thought** is moderately correlated with **affect** ($r = .590$; S.E. = .065) and **disorg** ($r = .518$; S.E. = .075), **disorg** is moderately correlated with **anergia** ($r = .366$; S.E. = .085) and **affect** ($r = .488$; S.E. = .082), **anergia** and **thought** are mildly correlated ($r = .182$; S.E. = .074), and **affect** and **anergia** are not significantly related ($r = -.012$; S.E. = .085).

Plotting Item Characteristic Curves

To aid interpretation of the factor-to-item relationships (i.e., the factor loadings and the thresholds), it is useful to look at plots of the item characteristic curves (ICCs) or tracelines. In these graphs, an individual's latent variable score is on the x-axis and the conditional probability of item endorsement given the latent variable score is on the y-axis. Items with steeper slopes have higher factor loadings and are more discriminating than items with flatter slopes. Items that are more to the left of the screen have lower thresholds, and are therefore easier to endorse.

There are many ways to make plots within R, including a variety of packages. Here we provide a relatively simple script for generating the ICCs. Because our focus is on fitting SEM models and not on making graphs, we do not walk through the script in great detail, touching only on the most important aspects.

As a first step, we need to extract the threshold and loading estimates from the fit object generated by lavaan for our model. We do that here:

```
est <- parameterEstimates(fit.2)
threshold <- est[est$op=="|",c("lhs","est")]
loading <- est[est$op=="=~",c("rhs","est")]
names(threshold) <- c("item","est")
names(loading) <- c("item","est")
```

This produces two matrices, one called **threshold** that consists of a column of labels for the items and a column of threshold estimates and another called **loading** that likewise includes the item labels in the first column but has a second column consisting of factor loading estimates.

Next we set up the plot. Here we demonstrate this for the items loading on the thought disorder factor.

```
plot(c(-4,4),c(0,1),type="n",xlab="Eta",ylab="Endorsement Probability")
items <- c("item08","item11","item12","item15")
lines <- c(1:length(items))
colors <- rainbow(length(items))
```

The **plot** function call here simply sets the frame for the plot, indicating that the x-axis will go from -4 to 4 and be labeled "Eta" and that the y-axis will go from 0 to 1 and be labeled "Endorsement probability". We then put the item labels for the items loading on thought disorder into the **items** vector and created the vectors **lines** and **colors** to enable us to vary the line types and colors of the ICCs for the four items.

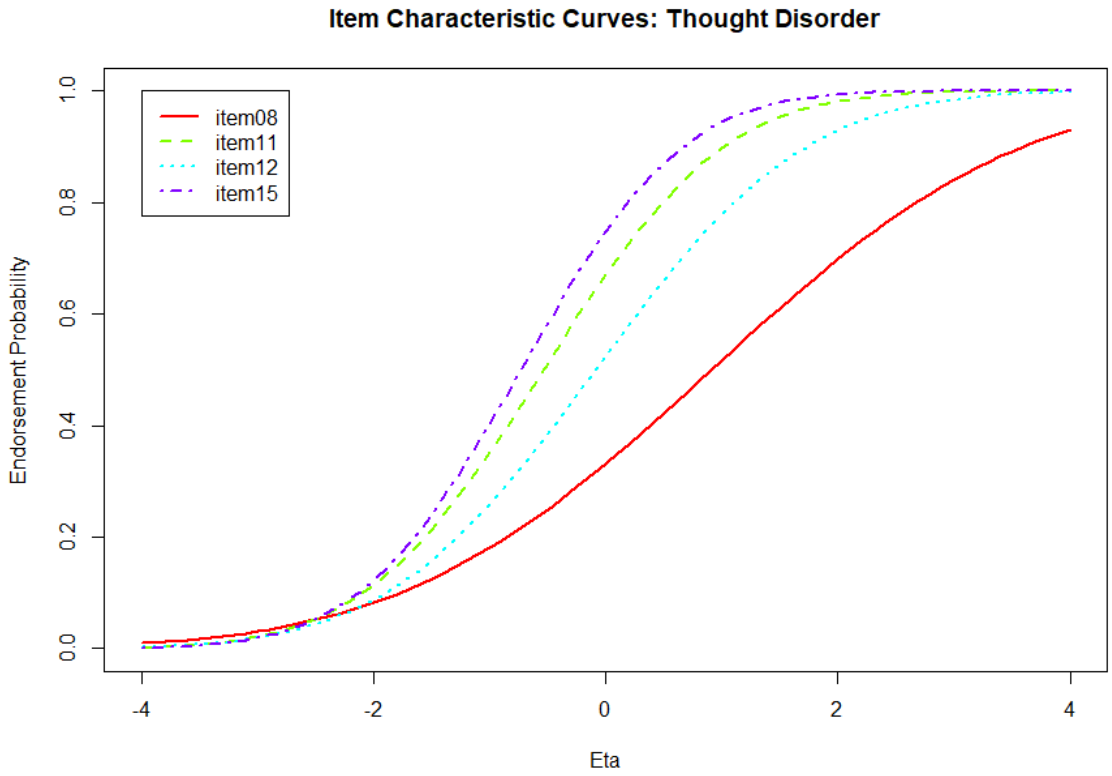
The actual plotting of the ICCs is done with the following script:

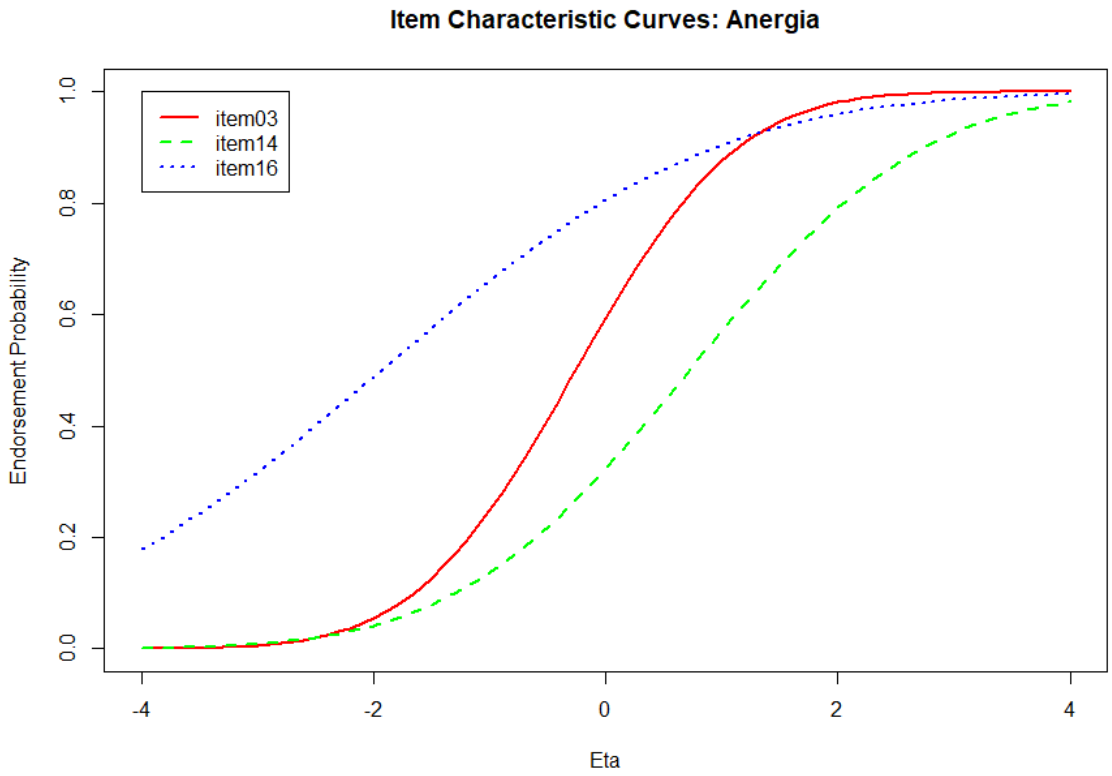
```
for (i in 1:4){
  eta <- seq(-4, 4, by=.1)
  int <- -1*threshold[i,"est"]
  slp <- loading[i,"est"]
  prob <- pnorm(int + slp*eta)
  dat <- data.frame(cbind(eta, prob))

  lines(dat$eta, dat$prob, type="l", lwd=2, lty=lines[i],
        col=colors[i])
}
```

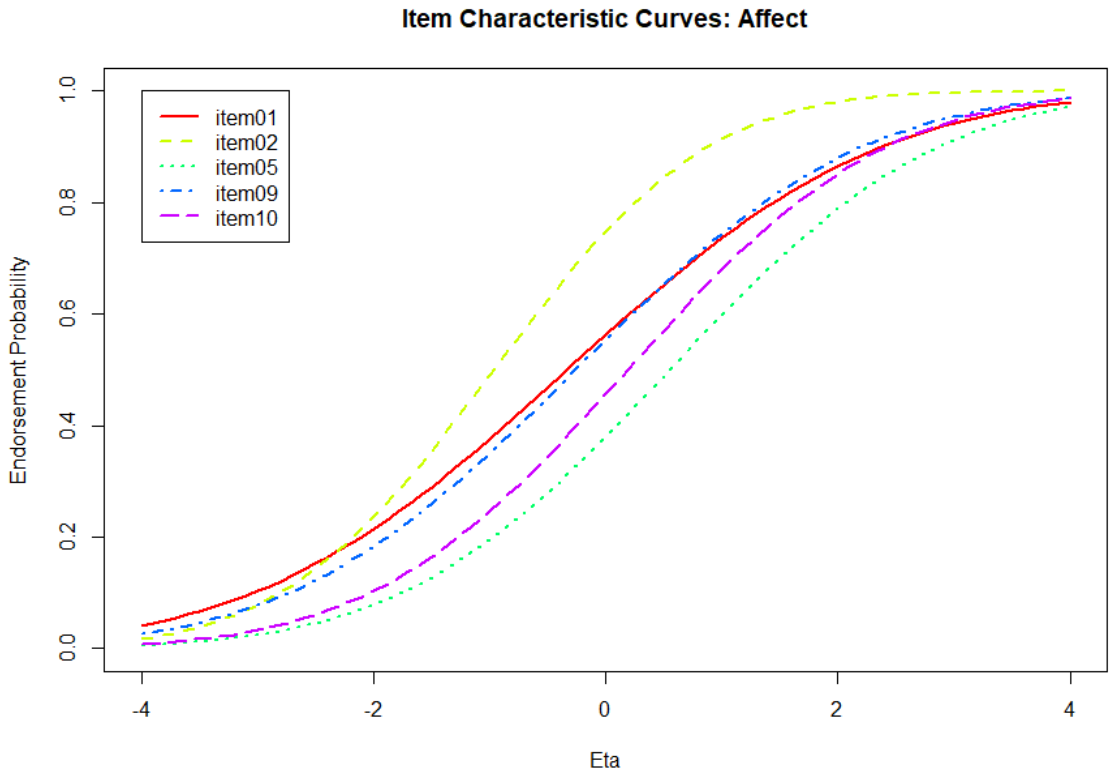
For each item, we choose factor values from -4 to 4 in increments of .1, pick off the threshold and loading of the item, and compute the implied probability of endorsement of the item given these item parameters. The values of the factor and corresponding endorsement probabilities are put into a data frame called **dat**, and then the **lines** function is used to add the ICC for each item to the plot with different line types and colors across items.

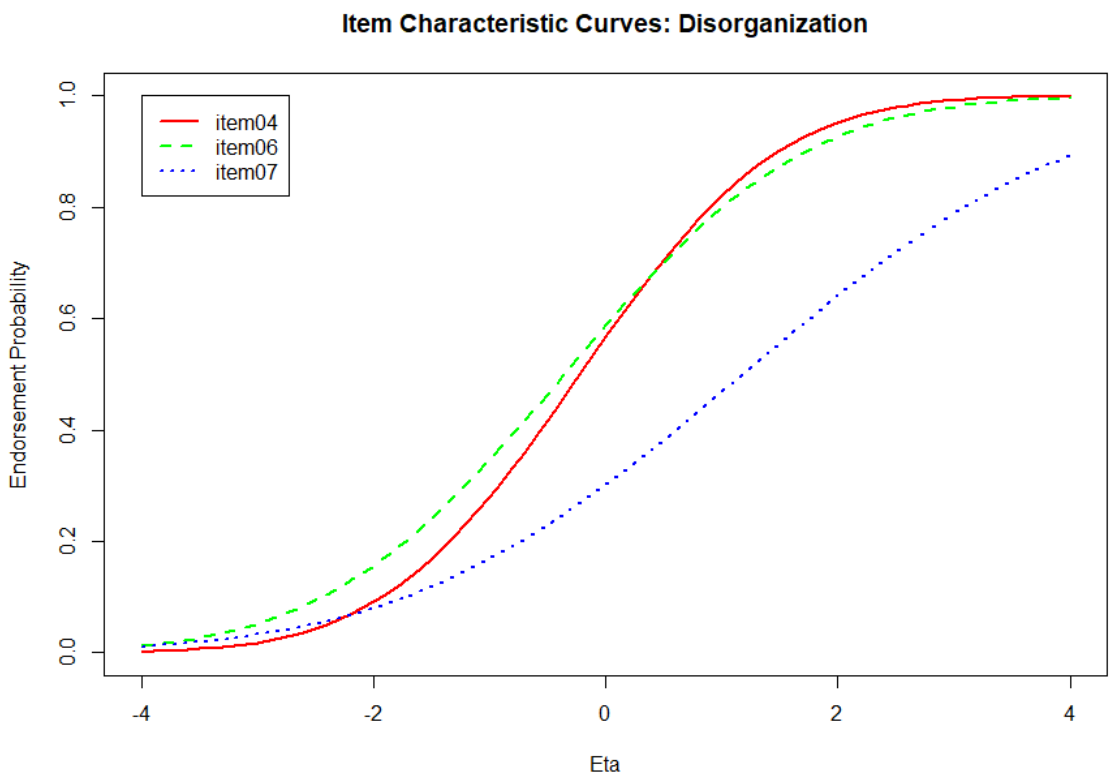
Last, we can give our plot a title and legend as follows:

```
title("Item Characteristic Curves: Thought Disorder")
legend(-4, 1, items, col=colors, lty=lines, lwd=2)
```









All of the ICCs look reasonable although there is clearly variation in the quality of the items (with lower quality items having flatter slopes).

For the thought disorder factor there is a great deal of variation in the severity and discrimination of the items. **Item08** "grandiosity" is both the least discriminating item (i.e., least strongly related to the factor, with the flattest ICC and lowest factor loading) and the symptom that is most severe (requires a very high level of thought disorder to have a high endorsement probability).

Moving to the **affect** factor we see these items are relatively consistent in their discrimination but vary somewhat in severity. **Item02,** "anxiety", has the lowest threshold and is thus the least severe symptom of **affect**. In contrast, **item05** "guilt feelings" has the highest threshold and thus its presence indicates more severe affect disorder.

Chapter 8
Latent Growth Curve Analysis

Latent Growth Curve Analysis of Antisocial Behavior

The data for this demonstration are from the National Longitudinal Study of Youth. The sample includes 405 children who were aged 6-8 at the first assessment (in 1986). Children were assessed bi-yearly (until 1992), resulting in up to four measurement occasions per child. Data are in the file **anti.dat** and the script is in **ch08.R**. The variables in the data set that we will use are described below:

id	participant identification; this will not be used in the analysis
gen	0 = female; 1 = male
homecog	level of cognitive support in the home (mean-centered); higher values denote more cognitive support in the home
anti6-anti14	repeated assessments of mother reports of child antisocial behavior (higher values indicate more antisocial behavior); post-script denotes child age at time of assessment

We are interested in characterizing the functional form of developmental change in aggressive behavior for children aged 6-14. We would also like to understand how our time invariant covariates (gender and cognitive support in the home) contribute to initial levels of, and change in, aggressive behavior.

Preliminary Steps

As a starting point, we will read in the data and assign variable names:

```
classes <- rep("numeric",12)
anti <- read.table("anti.dat", header=FALSE, colClasses=classes,
na.strings=".")
y <- paste("anti", 6:14, sep = "")
names(anti) <- c(c("id","gen","homecog"),y)
```

The **classes** vector created in the top line is referred to in the **read.table** function with the **colClasses** argument to define the data type for all variables as numeric. Without these steps, our variables would be assigned the factors data type instead. We also indicate that missing data is coded as **.** using the **na.strings** argument of **read.table**. The last two lines assign the variable names. We defined the vector **y** using the **paste** function to include the names of the repeated measures, i.e., **anti6**, **anti7**, **anti8**, …, **anti14**. Then we used **names** function to assign all the variable names.

We are now ready to examine some descriptive statistics. For simple univariate descriptive statistics, we like to us the **describe** function of the **psych** package (which must be installed).

```
library(psych)
describe(anti[c("gen","homecog",y)])
```

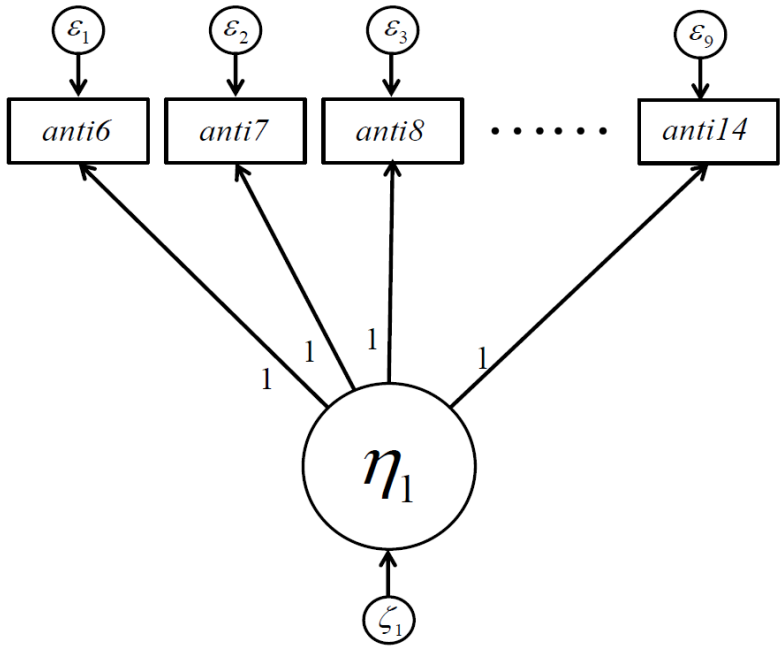
The univariate statistics provided by the **describe** function are shown here:

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
gen	1	405	0.50	0.50	1.00	0.50	0.00	0.00	1.00	1	0.00	-2.00	0.02
homecog	2	405	0.00	2.58	0.11	0.13	2.97	-7.89	5.11	13	-0.45	-0.14	0.13
anti6	3	122	1.57	1.67	1.00	1.34	1.48	0.00	9.00	9	1.33	2.32	0.15
anti7	4	168	1.55	1.55	1.00	1.35	1.48	0.00	7.00	7	1.05	0.82	0.12
anti8	5	146	1.97	1.80	1.00	1.75	1.48	0.00	7.00	7	0.92	0.01	0.15
anti9	6	192	1.89	1.95	1.00	1.58	1.48	0.00	9.00	9	1.23	1.18	0.14
anti10	7	151	2.14	2.14	2.00	1.81	1.48	0.00	10.00	10	1.24	1.22	0.17
anti11	8	174	1.79	1.95	1.00	1.50	1.48	0.00	10.00	10	1.54	3.01	0.15
anti12	9	135	1.84	1.78	2.00	1.62	2.97	0.00	7.00	7	0.75	-0.33	0.15
anti13	10	173	2.24	2.27	2.00	1.93	2.97	0.00	10.00	10	1.09	0.74	0.17
anti14	11	101	1.96	2.09	1.00	1.67	1.48	0.00	10.00	10	1.13	1.11	0.21

We see that there are 405 cases, but there are varying numbers assessed at each specific age (e.g., 122 of the 405 cases provided data on antisocial at age 6, 168 at age 7, and so on). There appears to be a slight upward trend in the means over time.

Intercept-Only Model

As we have seen in preceding chapters, it is often useful to start with simple models and then to increase in model complexity. Adhering to this strategy, we begin with an intercept-only model. This model evaluates whether individual differences exist with respect to mean levels of mother-reported aggression, aggregating over age of assessment. That is, we exclude a time trend from the initial model. The path diagram for this model is:



The measurement model takes the following form:

$$\begin{bmatrix} anti6_i \\ anti7_i \\ anti8_i \\ anti9_i \\ anti10_i \\ anti11_i \\ anti12_i \\ anti13_i \\ anti14_i \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \eta_{1i} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{bmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \ldots, \theta_{99})$$

and the latent variable model is simply

$$\eta_{1i} = \alpha_1 + \zeta_{1i}$$

where $E(\eta_{1i}) = \alpha_1$ and $VAR(\eta_{1i}) = VAR(\zeta_{1i}) = \psi$.

The repeated measures have the following model-implied mean and covariance structures:

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\alpha}$$

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \mathbf{1}\psi\mathbf{1}' + \boldsymbol{\Theta}$$

This model implies that the repeated measures of antisocial behavior exhibit no systematic change over time, and that variation among repeated measurements of antisocial behavior are solely a function of individual baseline propensities to engage in antisocial behavior and measurement error (or time-specific true score variance). Variance in the repeated measures that is not explained by the latent intercept (η_1) is considered to be unsystematic error ($\varepsilon_1 \ldots \varepsilon_9$).

The script for specifying and fitting this model in lavaan is shown here:

```
library(lavaan)

int <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
             1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
             1*anti14
       '
fit.int <- growth(int, data=anti, missing="ML")
summary(fit.int, fit.measures=TRUE, estimates=FALSE)
```

In the model syntax object, we simply list out the repeated measures as indicator variables. To define `I` as a latent intercept factor, we assigned each repeated measure a factor loading of one.

To fit the model, we use the **growth** function, which invokes a number of defaults useful for growth modeling such as including mean structure, setting indicator intercepts to zero, and estimating means of latent growth factors. Thus, we need not explicitly define these aspects of the model in the model syntax object. Note that we also indicated **missing = "ML"** so that lavaan would use the individual likelihood version of ML to accommodate partially missing data.

Model fit is shown below:

```
lavaan 0.6-2 ended normally after 40 iterations

  Optimization method                           NLMINB
  Number of free parameters                         11

  Number of observations                           405
  Number of missing patterns                        49

  Estimator                                         ML
  Model Fit Test Statistic                     102.290
  Degrees of freedom                                43
  P-value (Chi-square)                           0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                       NA
  Tucker-Lewis Index (TLI)                          NA

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2661.782
  Loglikelihood unrestricted model (H1)      -2610.637

  Number of free parameters                         11
  Akaike (AIC)                                5345.564
  Bayesian (BIC)                              5389.607
  Sample-size adjusted Bayesian (BIC)         5354.702

Root Mean Square Error of Approximation:

  RMSEA                                          0.058
  90 Percent Confidence Interval          0.044  0.073
  P-value RMSEA <= 0.05                          0.163

Standardized Root Mean Square Residual:

  SRMR                                           0.191
```

The model fit statistics reported by lavaan do not match those presented in lecture, which were generated by Mplus. For instance, lavaan reports a chi-square of 102.29 with 43 degrees of freedom. In contrast, Mplus reports a chi-square of 101.71 with 39 degrees of freedom. Additionally, lavaan did not compute CFI or TLI, whereas Mplus did. The reason for these differences has to do with how the two programs account for missing data.

We can observe the extent of missing data using the **lavInspect** function of lavaan:

```
coverage <- lavInspect(fit.int, what = "coverage")
coverage*405
```

The coverage matrix generated by the **what = "coverage"** argument to this function indicates the fraction of data available for each element of the covariance matrix. We multiplied this by 405, our total sample size, to observe the number of cases available with which to estimate each covariance:

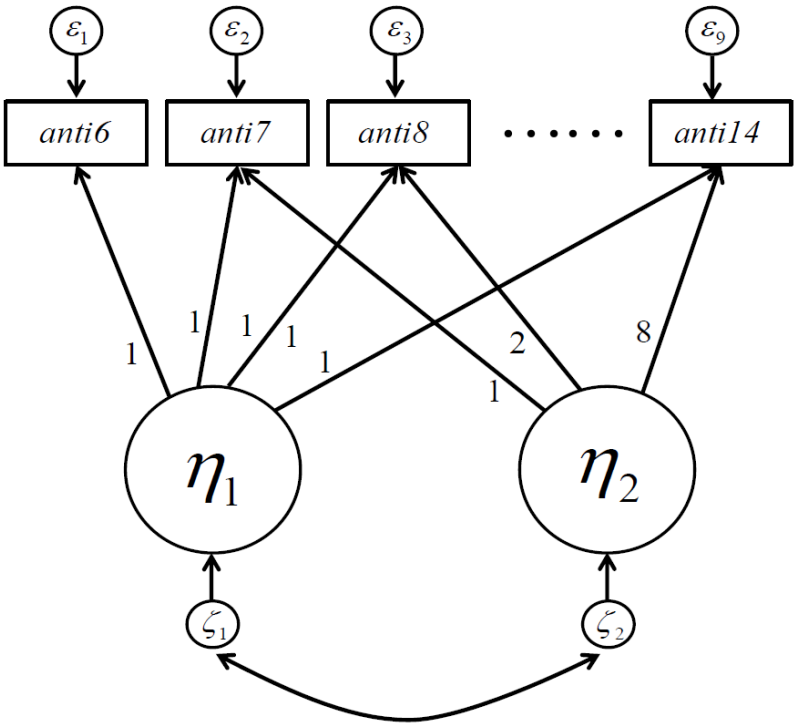
```
       anti6 anti7 anti8 anti9 anti10 anti11 anti12 anti13 anti14
anti6  122
anti7    0   168
anti8   31     0   146
anti9   86   106     0   192
anti10  12    49    98     3   151
anti11  80    82    28   139     1   174
anti12  28    45    81    16    99    14   135
anti13  61    96    22   138    16   145    12   173
anti14   0    35    66     3    85     4    82    11   101
```

There are four elements for which zero cases provide data. These "zero cells" are due to the accelerated longitudinal design of the study. For instance, no children were observed at both 6 and 7 years of age given the biennial assessment schedule, thus the covariance between **anti6** and **anti7** has zero coverage in the data. Mplus automatically detects these zero cells and removes them from the degrees of freedom for the model chi-square test. Lavaan does not do so. This is why the degrees of freedom for the chi-square test were lower by four in Mplus relative to lavaan. Additionally, the precise values of the model chi-squares differ slightly owing to differences in convergence criteria for the saturated model. These zero-coverage covariances are technically not identified in the saturated model and thus take on arbitrary (and irrelevant) values in the estimation process. Lavaan and Mplus will proceed with fitting the saturated model in any event whereas most other software programs will not (and thus will not provide any tests of absolute fit).

In this case, regardless of whether we account for the zero-cells or not, we reach the same conclusion regarding model fit. All measures of fit suggest that the intercept-only model does not adequately describe our data. The sample means provide a clue for why this is the case; there appears to be a linear upward trend in the means over time. Thus, we will move to a more complex model with a latent linear slope in addition to the latent intercept.

Intercept and Slope Model

We will be able to determine whether including a latent linear slope significantly improves model fit relative to the intercept-only model because these two models are nested. We want to fit the following linear growth model:



The model takes the following form:

$$\begin{bmatrix} anti6_i \\ anti7_i \\ anti8_i \\ anti9_i \\ anti10_i \\ anti11_i \\ anti12_i \\ anti13_i \\ anti14_i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \end{bmatrix} \begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{bmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \ldots, \theta_{99})$$

and the latent variable model is now

$$\begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} \zeta_{1i} \\ \zeta_{2i} \end{bmatrix}$$

where

$$E(\boldsymbol{\eta}_i) = \boldsymbol{\alpha},\ COV(\boldsymbol{\eta}_i) = COV(\boldsymbol{\zeta}_i) = \boldsymbol{\Psi}, \text{ and } \boldsymbol{\Psi} = \begin{bmatrix} \psi_{11} & \\ \psi_{21} & \psi_{22} \end{bmatrix}$$

The repeated measures have the following model implied mean and covariance structures:

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}\boldsymbol{\alpha}$$

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}\boldsymbol{\Psi}\boldsymbol{\Lambda}' + \boldsymbol{\Theta}$$

Now the repeated measures have an expected value that consists of the model-implied mean of antisocial behavior at age 6 (α_1) plus the expected change in antisocial behavior corresponding to the linear increase in age (α_2). The covariance matrix of the repeated measures reflect the variances and covariance of the intercept and slope factors, as transmitted through the factor loading matrix. Additional variance is added at each time point by the independent errors of the repeated measures.

An interesting implication of the linear growth model is that the variance contribution from the linear slope increases as a quadratic function of time because the variance of the slope factor is multiplied by the square of the factor loading in the calculation of total variance of the repeated measures.

This model is specified and fit as follows:

```
lin <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
            1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
            1*anti14
        S =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
            4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
            8*anti14
         '
fit.lin <- growth(lin, data=anti, missing="ML")
summary(fit.lin, fit.measures=TRUE, rsquare=TRUE)

lavTestLRT(fit.lin, fit.int)
```

We have added a new latent variable, **S**, and specified that each of the repeated measures load on this latent variable with linearly incrementing fixed factor loadings of 0, 1, 2, etc. This defines **S** to be the slope factor of a linear growth curve model. `I` retains its interpretation as the intercept of the model, now representing initial levels of antisocial behavior (at age 6).

In addition to requesting fit statistics, parameter estimates, and R-squared statistics, we also requested a likelihood ratio test to examine the improvement in fit of the linear model relative to the intercept-only model.

Model fit statistics for the linear growth model are shown here:

```
lavaan 0.6-2 ended normally after 53 iterations

  Optimization method                           NLMINB
  Number of free parameters                         14

  Number of observations                           405
  Number of missing patterns                        49

  Estimator                                         ML
  Model Fit Test Statistic                      53.171
  Degrees of freedom                                40
  P-value (Chi-square)                           0.079

User model versus baseline model:

  Comparative Fit Index (CFI)                       NA
  Tucker-Lewis Index (TLI)                          NA

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2637.223
  Loglikelihood unrestricted model (H1)      -2610.637

  Number of free parameters                         14
  Akaike (AIC)                                5302.445
  Bayesian (BIC)                              5358.500
  Sample-size adjusted Bayesian (BIC)         5314.076

Root Mean Square Error of Approximation:

  RMSEA                                          0.029
  90 Percent Confidence Interval          0.000  0.047
  P-value RMSEA <= 0.05                          0.972

Standardized Root Mean Square Residual:

  SRMR                                           0.123
```

The chi-square and RMSEA suggest this model fits well. The likelihood ratio test comparing this model with the intercept-only model is also highly significant ($\chi^2(3) = 49.1, p < .001$), indicating that the linear slope substantially improves the model fit:

```
Chi Square Difference Test

        Df    AIC    BIC   Chisq Chisq diff Df diff Pr(>Chisq)
fit.lin 40 5302.4 5358.5  53.171
fit.int 43 5345.6 5389.6 102.290     49.119       3  1.231e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We thus now consider the parameter estimates for this model, shown here:

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  I =~
    anti6             1.000
    anti7             1.000
    anti8             1.000
    anti9             1.000
    anti10            1.000
    anti11            1.000
    anti12            1.000
    anti13            1.000
    anti14            1.000
  S =~
    anti6             0.000
    anti7             1.000
    anti8             2.000
    anti9             3.000
    anti10            4.000
    anti11            5.000
    anti12            6.000
    anti13            7.000
    anti14            8.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~~
    S                 0.050    0.052    0.962    0.336

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             0.000
   .anti7             0.000
   .anti8             0.000
   .anti9             0.000
   .anti10            0.000
   .anti11            0.000
   .anti12            0.000
   .anti13            0.000
   .anti14            0.000
    I                 1.621    0.086   18.870    0.000
    S                 0.074    0.018    4.128    0.000
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             1.881    0.375    5.021    0.000
   .anti7             1.345    0.264    5.087    0.000
   .anti8             1.686    0.290    5.818    0.000
   .anti9             1.867    0.251    7.430    0.000
   .anti10            2.181    0.323    6.748    0.000
   .anti11            1.664    0.249    6.691    0.000
   .anti12            1.458    0.274    5.329    0.000
   .anti13            1.799    0.337    5.335    0.000
   .anti14            1.726    0.414    4.174    0.000
    I                 1.093    0.292    3.743    0.000
    S                 0.027    0.012    2.167    0.030

R-Square:
                   Estimate
    anti6             0.367
    anti7             0.476
    anti8             0.454
    anti9             0.467
    anti10            0.468
    anti11            0.576
    anti12            0.645
    anti13            0.633
    anti14            0.676
```

Notice that the factor loadings and observed variable intercepts are all fixed and hence no standard errors, z-statistics, or p-values are reported for these values.

The means for the intercept and linear slope components of growth are both significant. On average, mothers report antisocial behavior of 1.62 at age 6 and these scores increase at a rate of $.07$-units per year.

The variances of the intercept and linear slope components of growth are also both significant. There is significant individual variability in the initial levels of antisocial behavior and in the rates of increase in antisocial behavior over time.

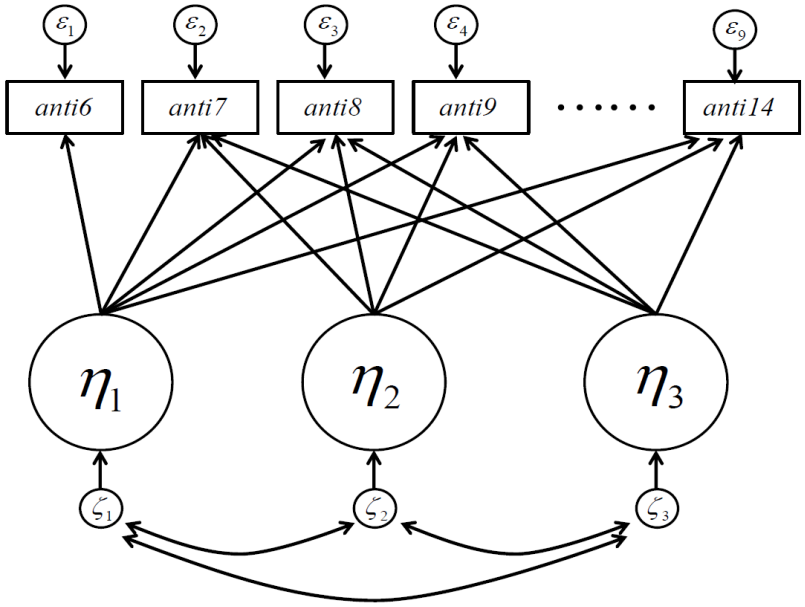
There is a non-significant covariance between the intercept and the linear slope components of growth. On average, individual variability in starting point is not related to individual variability in rate of change over time.

The R-squared statistics show that the growth factors account for an increasing percentage of the variance in the repeated measures as age advances.

Although this model appears quite reasonable, we have not yet considered functional forms of growth other than a linear trajectory—we will do this next.

Quadratic Growth Trajectory Model

We now extend the model to include a quadratic growth factor, as shown in the diagram:



The model takes the form:

$$\begin{bmatrix} anti6_i \\ anti7_i \\ anti8_i \\ anti9_i \\ anti10_i \\ anti11_i \\ anti12_i \\ anti13_i \\ anti14_i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \end{bmatrix} \begin{bmatrix} \eta_{1i} \\ \eta_{2i} \\ \eta_{3i} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{bmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \ldots, \theta_{99})$$

and the latent variable model is now

$$\begin{bmatrix} \eta_{1i} \\ \eta_{2i} \\ \eta_{3i} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} + \begin{bmatrix} \zeta_{1i} \\ \zeta_{2i} \\ \zeta_{3i} \end{bmatrix}$$

where

$$COV\left(\zeta_i\right) = \boldsymbol{\Psi} = \begin{bmatrix} \psi_{11} & & \\ \psi_{21} & \psi_{22} & \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix}$$

The repeated measures again have the following model implied mean and covariance structures:

$$\boldsymbol{\mu}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}\boldsymbol{\alpha}$$

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}\boldsymbol{\Psi}\boldsymbol{\Lambda}' + \boldsymbol{\Theta}$$

The script for specifying and estimating this model is:

```
quad <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
              1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
              1*anti14
         L =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
              4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
              8*anti14
         Q =~ 0*anti6 + 1*anti7 + 4*anti8 + 9*anti9 +
              16*anti10 + 25*anti11 + 36*anti12 + 49*anti13 +
              64*anti14
        '
fit.quad <- growth(quad, data=anti, missing="ML")
summary(fit.quad, fit.measures=TRUE, rsquare=TRUE)

lavTestLRT(fit.quad, fit.lin)
```

Now we have specified a third factor called **Q** to represent quadratic growth by fixing the factor loadings to the squares of the linear factor loadings. We also requested a likelihood ratio test of the quadratic model relative to the linear model.

Model fit for the quadratic model is given here:

```
lavaan 0.6-2 ended normally after 86 iterations

  Optimization method                           NLMINB
  Number of free parameters                         18

  Number of observations                           405
  Number of missing patterns                        49

  Estimator                                         ML
  Model Fit Test Statistic                      42.558
  Degrees of freedom                                36
  P-value (Chi-square)                           0.210

User model versus baseline model:

  Comparative Fit Index (CFI)                       NA
  Tucker-Lewis Index (TLI)                          NA

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2631.916
  Loglikelihood unrestricted model (H1)      -2610.637

  Number of free parameters                         18
  Akaike (AIC)                                5299.832
  Bayesian (BIC)                              5371.902
  Sample-size adjusted Bayesian (BIC)         5314.786

Root Mean Square Error of Approximation:

  RMSEA                                          0.021
  90 Percent Confidence Interval          0.000  0.043
  P-value RMSEA <= 0.05                          0.989

Standardized Root Mean Square Residual:

  SRMR                                           0.130
```

This model fits the data significantly better than the linear growth model with an LRT of $\Delta\chi^2(4) = 10.6, p = .03$; however, the improvement is modest.

```
Chi Square Difference Test

         Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.quad 36 5299.8 5371.9 42.558
fit.lin  40 5302.4 5358.5 53.171     10.613       4    0.03127 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Parameter estimates are shown below:

```
 Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  I =~
    anti6             1.000
    anti7             1.000
    anti8             1.000
    anti9             1.000
    anti10            1.000
    anti11            1.000
    anti12            1.000
    anti13            1.000
    anti14            1.000
  L =~
    anti6             0.000
    anti7             1.000
    anti8             2.000
    anti9             3.000
    anti10            4.000
    anti11            5.000
    anti12            6.000
    anti13            7.000
    anti14            8.000
  Q =~
    anti6             0.000
    anti7             1.000
    anti8             4.000
    anti9             9.000
    anti10           16.000
    anti11           25.000
    anti12           36.000
    anti13           49.000
    anti14           64.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~~
    L                -0.492    0.355   -1.384    0.166
    Q                 0.049    0.036    1.351    0.177
  L ~~
    Q                -0.036    0.021   -1.710    0.087

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             0.000
   .anti7             0.000
   .anti8             0.000
   .anti9             0.000
   .anti10            0.000
   .anti11            0.000
   .anti12            0.000
```

```
   .anti13            0.000
   .anti14            0.000
    I                 1.520    0.107   14.171    0.000
    L                 0.155    0.061    2.540    0.011
    Q                -0.011    0.007   -1.433    0.152

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             0.594    0.792    0.750    0.453
   .anti7             0.900    0.368    2.446    0.014
   .anti8             1.603    0.283    5.662    0.000
   .anti9             1.735    0.263    6.605    0.000
   .anti10            1.916    0.336    5.709    0.000
   .anti11            1.499    0.262    5.713    0.000
   .anti12            1.310    0.273    4.801    0.000
   .anti13            2.158    0.399    5.411    0.000
   .anti14            2.000    0.722    2.770    0.006
    I                 2.123    0.793    2.677    0.007
    L                 0.399    0.190    2.097    0.036
    Q                 0.003    0.003    1.317    0.188

R-Square:
                   Estimate
    anti6             0.781
    anti7             0.635
    anti8             0.502
    anti9             0.530
    anti10            0.550
    anti11            0.638
    anti12            0.678
    anti13            0.556
    anti14            0.563
```

The mean ($\hat{\alpha}_3 = -.011$) and variance ($\psi_{33} = .003$) of the quadratic factor were both non-significant even though the likelihood ratio test was significant. Given the modest LRT and the non-significant mean and variance of the quadratic factor, we do not regard the quadratic model as providing a meaningful improvement on the linear model.

Freed Loading ("Fully Latent") Growth Model: Additional Demonstration

Because the linear LCM fit the data well for the repeated measures of antisocial behavior, we did not present a freed-loading model in lecture. However, here we demonstrate the application of this approach in case you would like to apply this to your own data. The fully latent LCM is an alternative way to capture potentially nonlinear change over time in which we retain the basic form of the linear growth model but we free some of the loadings associated with the slope factor. The slope factor is then perhaps better referred to as a "shape" factor, as estimated loadings that differ from the observed time scores will imply nonlinear trajectories.

Recall that there are two primary parameterizations of this model, which we refer to in the course notes as options one and two, respectively. In option one, the first two loadings are fixed to zero and one to set the scale of the shape factor and the remaining loadings are estimated to capture potentially nonlinear change over time. In diagram form, this model is precisely the same as the two-factor linear model except that we estimate a subset of loadings on the slope factor.

The matrix expression of the model is identical to the linear model, with the sole exception that the factor loadings for the shape factor are estimated for ages eight through fourteen:

$$\begin{bmatrix} anti6_i \\ anti7_i \\ anti8_i \\ anti9_i \\ anti10_i \\ anti11_i \\ anti12_i \\ anti13_i \\ anti14_i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & \lambda_3 \\ 1 & \lambda_4 \\ 1 & \lambda_5 \\ 1 & \lambda_6 \\ 1 & \lambda_7 \\ 1 & \lambda_8 \\ 1 & \lambda_9 \end{bmatrix} \begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{bmatrix}$$

All else is as defined for the linear model.

The corresponding lavaan script is:

```
free.1 <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
                1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
                1*anti14
           S =~ 0*anti6 + 1*anti7 + anti8 + anti9 +
                anti10 + anti11 + anti12 + anti13 +
                anti14
          '
fit.free.1 <- growth(free.1, data=anti, missing="ML")
summary(fit.free.1, fit.measures=TRUE, rsquare=TRUE)

lavTestLRT(fit.free.1, fit.lin)
```

Notice that the factor loadings for **anti8** through **anti14** on the factor **S** are no longer constrained to fixed values and hence will instead be estimated. Additionally, we requested a likelihood ratio test to compare the fit of this model to the linear growth model.

Resulting model fit is reported here:

```
  Estimator                                         ML
  Model Fit Test Statistic                      27.651
  Degrees of freedom                            33
  P-value (Chi-square)                          0.731

User model versus baseline model:

  Comparative Fit Index (CFI)                   NA
  Tucker-Lewis Index (TLI)                      NA

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2624.462
  Loglikelihood unrestricted model (H1)      -2610.637

  Number of free parameters                     21
  Akaike (AIC)                               5290.925
  Bayesian (BIC)                             5375.006
  Sample-size adjusted Bayesian (BIC)        5308.371

Root Mean Square Error of Approximation:

  RMSEA                                         0.000
  90 Percent Confidence Interval          0.000  0.028
  P-value RMSEA <= 0.05                         1.000

Standardized Root Mean Square Residual:

  SRMR                                          0.103
```

This is a very good fitting model. Indeed, the model may fit too well. Indications that we may be over-fitting the data include a model chi-square less than the degrees of freedom (the expected value for a chi-square) and a RMSEA of zero. Comparing this model to the linear growth model, we find significantly better fit for the freed loadings model ($\chi^2(7) = 25.52, p < .001$):

```
Chi Square Difference Test

           Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.free.1 33 5290.9 5375.0 27.651
fit.lin    40 5302.4 5358.5 53.171     25.521       7  0.0006132 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

It’s woth noting, however, that the BIC, which favors parsimony, is actually better for the linear model (5358.5 versus 5375.0).

Parameter estimates are shown below:

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  I =~
    anti6             1.000
    anti7             1.000
    anti8             1.000
    anti9             1.000
    anti10            1.000
    anti11            1.000
    anti12            1.000
    anti13            1.000
    anti14            1.000
  S =~
    anti6             0.000
    anti7             1.000
    anti8             6.478   11.023    0.588    0.557
    anti9             7.010   11.719    0.598    0.550
    anti10            9.231   16.021    0.576    0.564
    anti11            6.703   11.147    0.601    0.548
    anti12            7.017   11.907    0.589    0.556
    anti13           13.858   24.615    0.563    0.573
    anti14            7.688   12.986    0.592    0.554

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~~
    S                 0.022    0.047    0.472    0.637

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             0.000
   .anti7             0.000
   .anti8             0.000
   .anti9             0.000
   .anti10            0.000
   .anti11            0.000
   .anti12            0.000
   .anti13            0.000
   .anti14            0.000
    I                 1.524    0.112   13.563    0.000
    S                 0.058    0.108    0.537    0.591
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             1.815    0.389    4.670    0.000
   .anti7             1.272    0.326    3.904    0.000
   .anti8             1.491    0.260    5.731    0.000
   .anti9             1.757    0.262    6.696    0.000
   .anti10            1.916    0.360    5.322    0.000
   .anti11            1.878    0.255    7.380    0.000
   .anti12            1.617    0.277    5.832    0.000
   .anti13            1.109    0.536    2.070    0.038
   .anti14            2.169    0.395    5.493    0.000
    I                 1.024    0.335    3.051    0.002
    S                 0.013    0.048    0.272    0.786

R-Square:
                   Estimate
    anti6             0.361
    anti7             0.459
    anti8             0.555
    anti9             0.529
    anti10            0.571
    anti11            0.504
    anti12            0.550
    anti13            0.789
    anti14            0.496
```

The parameter estimates are also somewhat suggestive of over-fit. Notice that the factor loadings jump around quite a bit and that both the mean and variance of the slope (or shape) factor are non-significant. Each factor loading can be interpreted as indicating a multiple for the amount of change occurring between that time and the first time point relative to the change observed across the first two time points (e.g., the amount of change between age 6 and age 8 is 6.5 times the amount of change between age 6 and age 7).

The alternative parameterization of the freed loading model, or option two, scales the slope factor in terms of the first and last time points. The diagram is again the same as that for the linear model except that we freely estimate the interior loadings on the shape factor:

$$\begin{bmatrix} anti6_i \\ anti7_i \\ anti8_i \\ anti9_i \\ anti10_i \\ anti11_i \\ anti12_i \\ anti13_i \\ anti14_i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & \lambda_2 \\ 1 & \lambda_3 \\ 1 & \lambda_4 \\ 1 & \lambda_5 \\ 1 & \lambda_6 \\ 1 & \lambda_7 \\ 1 & \lambda_8 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{bmatrix}$$

The script for fitting this model is shown here:

```
free.2 <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
                1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
                1*anti14
           S =~ 0*anti6 + anti7 + anti8 + anti9 +
                anti10 + anti11 + anti12 + anti13 +
                1*anti14
          '
fit.free.2 <- growth(free.2, data=anti, missing="ML")
summary(fit.free.2)
```

Note that the only change to the model syntax object is the placement of the fixed factor loadings for the shape factor, **s**. Additionally, because model fit will be identical to the first parameterization of this model, we did not request this information again in the **summary** function. Nor did we request R-squared statistics or conduct the LRT, as these would also all be identical.

In contrast, parameter estimates will differ given the change in how the shape factor is scaled, as seen here:

```
 Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  I =~
    anti6             1.000
    anti7             1.000
    anti8             1.000
    anti9             1.000
    anti10            1.000
    anti11            1.000
    anti12            1.000
    anti13            1.000
    anti14            1.000
  S =~
    anti6             0.000
    anti7             0.130    0.220    0.592    0.554
    anti8             0.843    0.262    3.211    0.001
    anti9             0.912    0.266    3.433    0.001
    anti10            1.201    0.341    3.525    0.000
    anti11            0.872    0.261    3.336    0.001
    anti12            0.913    0.287    3.182    0.001
    anti13            1.803    0.521    3.463    0.001
    anti14            1.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~~
    S                 0.170    0.287    0.594    0.552
```

```
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             0.000
   .anti7             0.000
   .anti8             0.000
   .anti9             0.000
   .anti10            0.000
   .anti11            0.000
   .anti12            0.000
   .anti13            0.000
   .anti14            0.000
    I                 1.524    0.112   13.563    0.000
    S                 0.447    0.158    2.823    0.005

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6            1.815    0.389    4.670    0.000
   .anti7            1.272    0.326    3.904    0.000
   .anti8            1.491    0.260    5.731    0.000
   .anti9            1.757    0.262    6.696    0.000
   .anti10           1.916    0.360    5.322    0.000
   .anti11           1.878    0.255    7.380    0.000
   .anti12           1.617    0.277    5.832    0.000
   .anti13           1.109    0.536    2.070    0.038
   .anti14           2.169    0.395    5.493    0.000
    I                1.024    0.335    3.051    0.002
    S                0.773    0.514    1.504    0.133
```

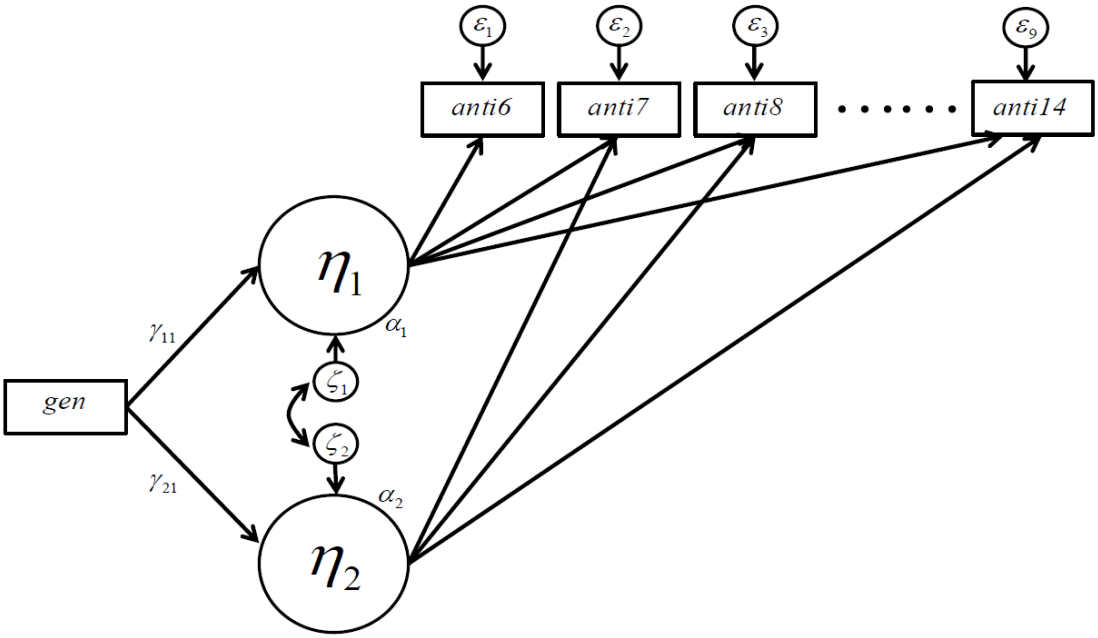
Each estimated factor loading now represents the proportion of total change that has occurred at each intermediate time point. For instance, the loading for age 8 indicates that 84.3% of the change observed between age 6 and age 14 has occurred by age 8. This is a particularly appealing interpretation if the trajectory is monotonically increasing or decreasing. In the present case, however, some factor loadings exceed one, leading to some awkward interpretations. For instance, the change between age 6 and age 13 is 180% of the change from age 6 to age 14. Again, the bouncing factor loadings are suggestions of problems with the model. For the most part, the observed means are fluctuating around a relatively stable level and freeing the factor loadings to capture these fluctuations seems to be over-fitting the data.

We thus move forward with the more parsimonious linear growth model.

Linear Growth Model Conditional on Gender

It is good practice to determine the functional form of growth before introducing exogenous predictors of growth. Now that we have established a linear growth model, we want to try to explain the growth with covariates.

First, we will condition the latent intercept and linear slope factors on gender, a binary time invariant covariate. The path diagram for this model is:



We return to the measurement model for linear growth:

$$\begin{bmatrix} anti6_i \\ anti7_i \\ anti8_i \\ anti9_i \\ anti10_i \\ anti11_i \\ anti12_i \\ anti13_i \\ anti14_i \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \\ 1 & 6 \\ 1 & 7 \\ 1 & 8 \end{bmatrix} \begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1i} \\ \varepsilon_{2i} \\ \varepsilon_{3i} \\ \varepsilon_{4i} \\ \varepsilon_{5i} \\ \varepsilon_{6i} \\ \varepsilon_{7i} \\ \varepsilon_{8i} \\ \varepsilon_{9i} \end{bmatrix}$$

where

$$COV(\boldsymbol{\varepsilon}_i) = \boldsymbol{\Theta} = DIAG(\theta_{11}, \theta_{22}, \ldots, \theta_{99})$$

and the latent variable model is now

$$\begin{bmatrix} \eta_{1i} \\ \eta_{2i} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} \gamma_{11} \\ \gamma_{21} \end{bmatrix} [gen_i] + \begin{bmatrix} \zeta_{1i} \\ \zeta_{2i} \end{bmatrix}$$

where

$$COV\left(\zeta_i\right) = \boldsymbol{\Psi} = \begin{bmatrix} \psi_{11} & \\ \psi_{21} & \psi_{22} \end{bmatrix}$$

The repeated measures have the following model implied mean and covariance structures:

$$\boldsymbol{\mu}_y(\boldsymbol{\theta}) = \boldsymbol{\Lambda}(\boldsymbol{\alpha} + \boldsymbol{\Gamma}\boldsymbol{\mu}_x)$$

$$\boldsymbol{\Sigma}_{yy}(\boldsymbol{\theta}) = \boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{\Sigma}_{xx}\boldsymbol{\Gamma}' + \boldsymbol{\Psi})\boldsymbol{\Lambda}' + \boldsymbol{\Theta}$$

The script for this model is:

```
gen <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
             1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
             1*anti14
        S =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
             4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
             8*anti14
        I ~ gen
        S ~ gen
       '
fit.gen <- growth(gen, data=anti, missing="ML")
summary(fit.gen, fit.measures=TRUE, rsquare=TRUE)
```

All of this is identical to the unconditional linear model from before with the exception that two lines have been added, **`I ~ gen`** and **`S ~ gen`**, to indicate that the growth factors are to be regressed on gender.

Results are given here:

```
  Estimator                                         ML
  Model Fit Test Statistic                      58.069
  Degrees of freedom                                47
  P-value (Chi-square)                           0.129

User model versus baseline model:

  Comparative Fit Index (CFI)                       NA
  Tucker-Lewis Index (TLI)                          NA

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2620.972
  Loglikelihood unrestricted model (H1)      -2591.937

  Number of free parameters                         16
  Akaike (AIC)                                5273.943
  Bayesian (BIC)                              5338.006
  Sample-size adjusted Bayesian (BIC)         5287.236
```

```
Root Mean Square Error of Approximation:

  RMSEA                                          0.024
  90 Percent Confidence Interval          0.000  0.043
  P-value RMSEA <= 0.05                          0.992

Standardized Root Mean Square Residual:

  SRMR                                           0.114

Parameter Estimates:

  Information                                 Observed
  Observed information based on                Hessian
  Standard Errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  I =~
    anti6             1.000
    anti7             1.000
    anti8             1.000
    anti9             1.000
    anti10            1.000
    anti11            1.000
    anti12            1.000
    anti13            1.000
    anti14            1.000
  S =~
    anti6             0.000
    anti7             1.000
    anti8             2.000
    anti9             3.000
    anti10            4.000
    anti11            5.000
    anti12            6.000
    anti13            7.000
    anti14            8.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~
    gen               0.789    0.166    4.756    0.000
  S ~
    gen               0.014    0.035    0.391    0.696

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
 .I ~~
   .S                 0.054    0.051    1.045    0.296
```

```
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             0.000
   .anti7             0.000
   .anti8             0.000
   .anti9             0.000
   .anti10            0.000
   .anti11            0.000
   .anti12            0.000
   .anti13            0.000
   .anti14            0.000
   .I                 1.224    0.119   10.316    0.000
   .S                 0.066    0.026    2.584    0.010

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             1.900    0.368    5.164    0.000
   .anti7             1.376    0.262    5.255    0.000
   .anti8             1.706    0.287    5.939    0.000
   .anti9             1.859    0.248    7.498    0.000
   .anti10            2.182    0.321    6.787    0.000
   .anti11            1.637    0.245    6.684    0.000
   .anti12            1.467    0.274    5.350    0.000
   .anti13            1.837    0.338    5.430    0.000
   .anti14            1.740    0.413    4.213    0.000
   .I                 0.910    0.283    3.214    0.001
   .S                 0.025    0.012    2.067    0.039

R-Square:
                   Estimate
    anti6             0.359
    anti7             0.467
    anti8             0.449
    anti9             0.467
    anti10            0.468
    anti11            0.579
    anti12            0.643
    anti13            0.627
    anti14            0.672
    I                 0.146
    S                 0.002
```

There was significant prediction of the intercept as a function of gender ($\hat{\gamma}_{11} = .789,\ p < .001$), but not of slope ($\hat{\gamma}_{21} = .014,\ p = .696$). On average, boys reported levels of antisocial behavior $.79$-units higher than did girls. On average, girls and boys did not differ in the rates of change in antisocial behavior over time. We have explained 14.6% of the variance in trajectory intercepts, but virtually no variance in trajectory slopes.

It is possible to probe the effect of **gen** by examining the simple intercept and slope for males and females separately. This can be accomplished in a variety of ways. Here we demonstrate how to obtain these tests and plots directly within R.

To compute and test simple intercepts and slopes we can define new parameters to represent these values within the model syntax object (similar to our use of this feature of lavaan in Chapter 3):

```
gen <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
            1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
            1*anti14
        S =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
            4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
            8*anti14
        I ~ a1*1 + g11*gen
        S ~ a2*1 + g21*gen

        #simple intercept and slope for girls
        int_girl := a1
        slp_girl := a2

        #simple intercept and slope for boys
        int_boy := a1 + g11
        slp_boy := a2 + g21
       '
fit.gen <- growth(gen, data=anti, missing="ML")
summary(fit.gen)
```

We have assigned labels **a1**, **g11**, **a2**, and **g21** to the intercepts and slopes of the latent variable regression equations. We then can compute the simple intercepts and slopes for girls and boys as shown by using the define operator, **:=**, to specify that these are linear combinations of the labeled parameters. In the equations for calculating the simple intercpets and slopes **g11** and **g21** are, respectively, multiplied by **gen**; however, for girls **gen** is equal to zero so these parameters simply drop out. That is, **a1** and **a2** already represent the simple intercept and slope for girls. For boys, **gen** is equal to one, so **g11** and **g21** are simply added in to each equation.

All of the output is identical to what we have seen before except for the following new bit:

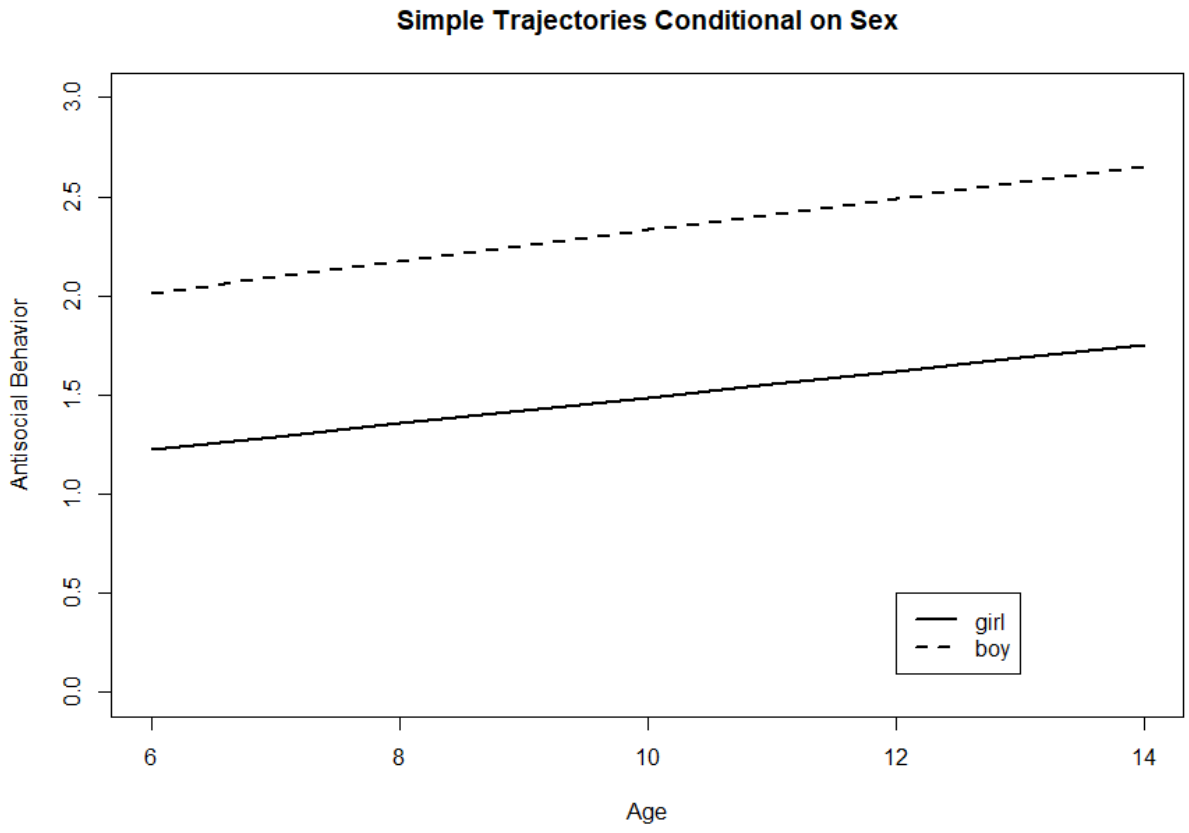
```
Defined Parameters:
                   Estimate  Std.Err  z-value  P(>|z|)
    int_girl          1.224    0.119   10.316    0.000
    slp_girl          0.066    0.026    2.584    0.010
    int_boy           2.013    0.117   17.219    0.000
    slp_boy           0.080    0.025    3.256    0.001
```

Given the lack of significance of the effect of gen on trajectory slopes, we would not interpret the difference in simple slopes between boys and girls, as this does not exceed what one might expect by chance alone at this sample size.

We can also take these values and generate a plot, similarly to how we generated ICC plots in Chapter 7 based on model estimates:

```
plot(c(6,14),c(0,3),type="n",xlab="Age",ylab="Antisocial Behavior")

age <- seq(6, 14, by=1)
pred.girl <- 1.224 + .066*(age-6)
pred.boy <- 2.013 + .080*(age-6)
dat <- data.frame(cbind(age,pred.girl,pred.boy))

lines(dat$age, dat$pred.girl, type="l", lwd=2, lty=1)
lines(dat$age, dat$pred.boy, type="l", lwd=2, lty=2)

title("Simple Trajectories Conditional on Sex")
legend(12, .5, c("girl","boy"), lty=c(1,2), lwd=2)
```

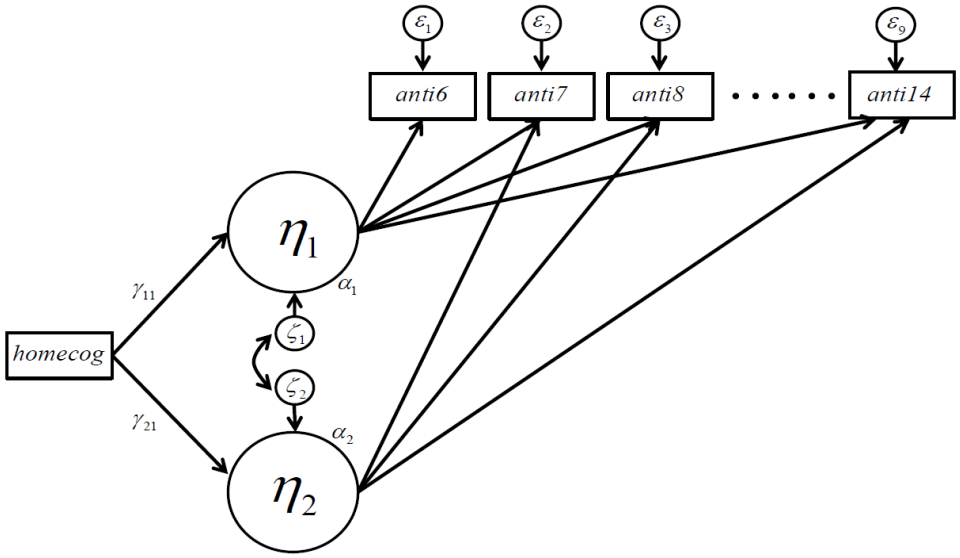
We first created the frame for the plot with the **plot** function. Then, we generated age scores of 6, 7, 8, …, 14. Next, we computed predicted values for girls and boys from the simple intercept and slope estimates, taking care to subtract six so that the first value of age produces a factor loading of zero for the slope. We put all of these into a data frame called **dat** and then added lines to the plot for girls and boys. Last, we added a title and legend to the plot.



From the plot, we see the significant gender difference in intercept for antisocial behavior, but not in rate of change over time.

Linear Growth Model Conditional on Cognitive Stimulation in the Home

Next, we will test whether cognitive stimulation in the home influences baseline ratings of antisocial behavior for children aged 6, and whether cognitive stimulation in the home influences rate of linear growth of aggressive behavior for children aged 6-14. Typically, both **gen** and **homecog** would be included in the same conditional growth model. However, we model them separately for demonstration purposes. The path diagram for the model is:



The measurement model is the same as before but the latent variable model is now:

$$\begin{bmatrix}\eta_{1i}\\ \eta_{2i}\end{bmatrix}=\begin{bmatrix}\alpha_1\\ \alpha_2\end{bmatrix}+\begin{bmatrix}\gamma_{11}\\ \gamma_{21}\end{bmatrix}\left[homecog_i\right]+\begin{bmatrix}\zeta_{1i}\\ \zeta_{2i}\end{bmatrix}$$

where

$$COV\left(\zeta_i\right)=\boldsymbol{\Psi}=\begin{bmatrix}\psi_{11}\\ \psi_{21} & \psi_{22}\end{bmatrix}$$

These equations imply the following mean and covariance structures for the repeated measures:

$$\boldsymbol{\mu}_y(\boldsymbol{\theta})=\boldsymbol{\Lambda}(\boldsymbol{\alpha}+\boldsymbol{\Gamma}\boldsymbol{\mu}_x)$$

$$\boldsymbol{\Sigma}_{yy}(\boldsymbol{\theta})=\boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{\Sigma}_{xx}\boldsymbol{\Gamma}'+\boldsymbol{\Psi})\boldsymbol{\Lambda}'+\boldsymbol{\Theta}$$

This model syntax object is specified almost identically to the prior conditional growth model, simply replacing **gen** with **homecog** as the regressor for the two latent factors:

```
home <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
              1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
              1*anti14
         S =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
              4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
              8*anti14
         I ~ homecog
         S ~ homecog
        '
fit.home <- growth(home, data=anti, missing="ML")
summary(fit.home, fit.measures=TRUE, rsquare=TRUE) [int slope];
```

Model fit is shown below:

```
  Estimator                                         ML
  Model Fit Test Statistic                      68.329
  Degrees of freedom                                47
  P-value (Chi-square)                           0.023

User model versus baseline model:

  Comparative Fit Index (CFI)                       NA
  Tucker-Lewis Index (TLI)                          NA

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -2626.666
  Loglikelihood unrestricted model (H1)      -2592.501

  Number of free parameters                         16
  Akaike (AIC)                                5285.331
  Bayesian (BIC)                              5349.394
  Sample-size adjusted Bayesian (BIC)         5298.624

Root Mean Square Error of Approximation:

  RMSEA                                          0.033
  90 Percent Confidence Interval          0.013  0.050
  P-value RMSEA <= 0.05                          0.951

Standardized Root Mean Square Residual:

  SRMR                                           0.116
```

The model fits the data adequately well, although not as well as it fit when the growth factors were conditional on **gen**. These models are not formally nested, so it is not possible to conduct a chi-square difference test between the two conditional models.

Parameter estimates are shown below:

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  I =~
    anti6             1.000
    anti7             1.000
    anti8             1.000
    anti9             1.000
    anti10            1.000
    anti11            1.000
    anti12            1.000
    anti13            1.000
    anti14            1.000
  S =~
    anti6             0.000
    anti7             1.000
    anti8             2.000
    anti9             3.000
    anti10            4.000
    anti11            5.000
    anti12            6.000
    anti13            7.000
    anti14            8.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~
    homecog          -0.078    0.033   -2.358    0.018
  S ~
    homecog          -0.015    0.007   -2.204    0.028

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
 .I ~~
   .S                 0.044    0.051    0.846    0.398

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6            0.000
   .anti7            0.000
   .anti8            0.000
   .anti9            0.000
   .anti10           0.000
   .anti11           0.000
   .anti12           0.000
   .anti13           0.000
   .anti14           0.000
   .I                1.622    0.085   19.041    0.000
   .S                0.075    0.018    4.189    0.000
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .anti6             1.876    0.371    5.061    0.000
   .anti7             1.351    0.263    5.126    0.000
   .anti8             1.646    0.286    5.761    0.000
   .anti9             1.835    0.247    7.436    0.000
   .anti10            2.233    0.328    6.810    0.000
   .anti11            1.746    0.256    6.825    0.000
   .anti12            1.453    0.270    5.380    0.000
   .anti13            1.728    0.329    5.244    0.000
   .anti14            1.716    0.407    4.219    0.000
   .I                 1.053    0.289    3.648    0.000
   .S                 0.025    0.012    2.097    0.036

R-Square:
                   Estimate
    anti6             0.368
    anti7             0.475
    anti8             0.461
    anti9             0.472
    anti10            0.464
    anti11            0.566
    anti12            0.648
    anti13            0.644
    anti14            0.678
    I                 0.037
    S                 0.058
```

There was significant prediction of the intercept as a function of **homecog** ($\hat{\gamma}_{11} = -.078, p = .018$), as well as the slope ($\hat{\gamma}_{21} = -.015, p = .028$). On average, children reported initial levels of antisocial behavior $.08$-units lower per one-unit increase in cognitive stimulation. On average, children reported trajectories with slopes $.02$-units less per one-unit increase in stimulation. The predictor **homecog** explains 3.7% of the variance in trajectory intercepts and 5.8% of the variance in trajectory slopes.

We must further probe these effects to understand the precise nature of these relations better. To do so, we will calculate and plot simple intercepts and slopes at plus and minus one standard deviation ($sd = 2.58$) from the mean of **homecog** (where the mean is equal to zero because it was mean-centered).

To compute and test the simple intercepts and slopes as a function of **homecog** we will again simply modify the model syntax object, labeling the relevant parameters and using the define operator to specify the quantities of interest:

```
home <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
              1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
              1*anti14
         S =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
              4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
              8*anti14
         I ~ a1*1 + g11*homecog
         S ~ a2*1 + g21*homecog

         #simple intercept and slope for low homecog
         int_lo := a1 - 2.58*g11
         slp_lo := a2 - 2.58*g21

         #simple intercept and slope for medium homecog
         int_md := a1
         slp_md := a2

         #simple intercept and slope for high homecog
         int_hi := a1 + 2.58*g11
         slp_hi := a2 + 2.58*g21
        '
fit.home <- growth(home, data=anti, missing="ML")
summary(fit.home)
```

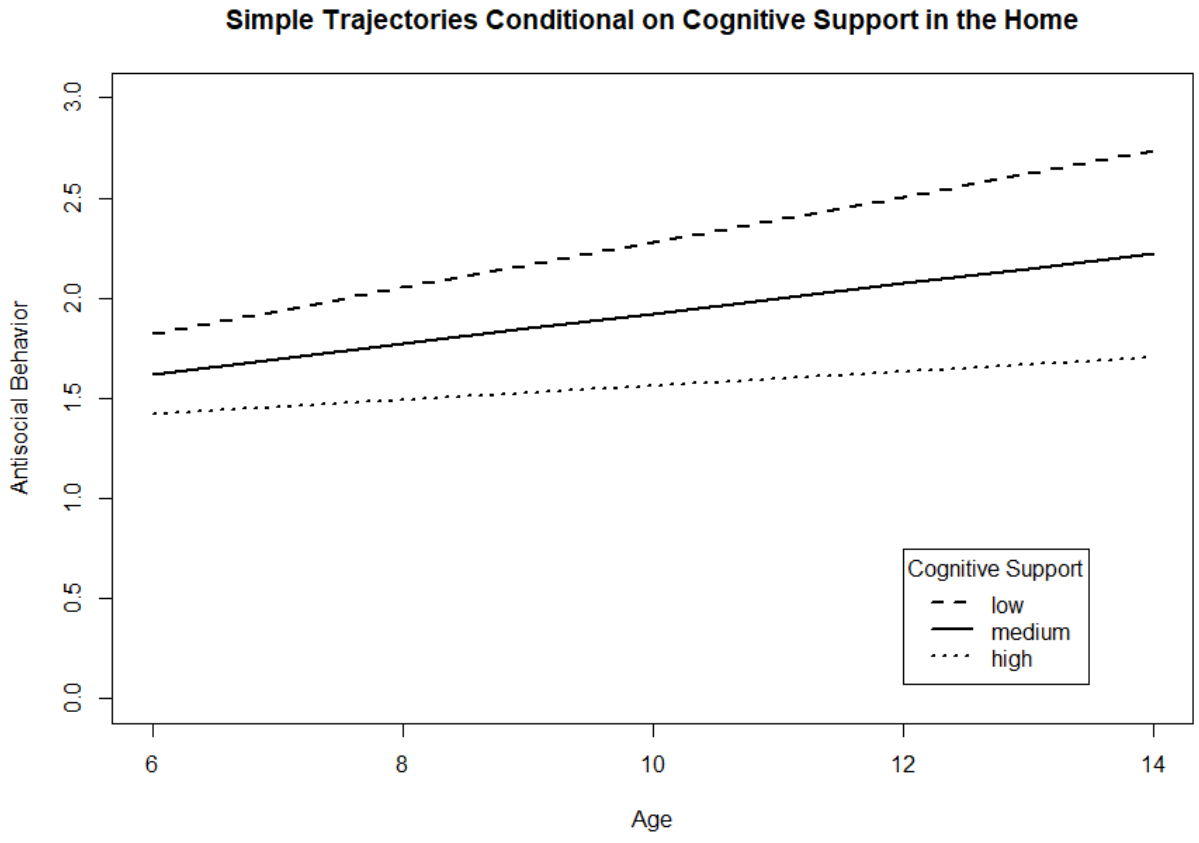
The new output is shown here:

```
Defined Parameters:
                   Estimate  Std.Err  z-value  P(>|z|)
    int_lo            1.823    0.121   15.020    0.000
    slp_lo            0.114    0.026    4.460    0.000
    int_md            1.622    0.085   19.041    0.000
    slp_md            0.075    0.018    4.189    0.000
    int_hi            1.421    0.120   11.863    0.000
    slp_hi            0.035    0.025    1.410    0.158
```

Notice that the simple slope is significantly positive at low and medium levels of home cognitive support, **slp_lo** and **slp_md**, respectively, but is not significantly different from zero when home cognitive support is high, or **slp_hi**.

We can also plot the simple trajectories based on these intercepts and slopes much as we did previously for gender.

```
plot(c(6,14),c(0,3),type="n",xlab="Age",ylab="Antisocial Behavior")

age <- seq(6, 14, by=1)
pred.lo <- 1.823 + .114*(age-6)
pred.md <- 1.622 + .075*(age-6)
pred.hi <- 1.421 + .035*(age-6)
dat <- data.frame(cbind(age,pred.lo,pred.md,pred.hi))

lines(dat$age, dat$pred.lo, type="l", lwd=2, lty=2)
lines(dat$age, dat$pred.md, type="l", lwd=2, lty=1)
lines(dat$age, dat$pred.hi, type="l", lwd=2, lty=3)

title("Simple Trajectories Conditional on Cognitive Support in the Home")
legend(12, .75, c("low","medium", "high"), title="Cognitive Support",
       lty=c(2,1,3), lwd=2)
```



The plot clearly conveys that the slope is relatively flat at high **homecog** but becomes steeper as **homecog** decreases to medium and low levels.

These models could be expanded to include the simultaneous effects of gender and cognitive stimulation, or even the interaction between the two, but we do not pursue this further here.

Appendix A: Generating Path Diagrams in R

This appendix presents two fully worked examples drawn from class demonstrating the creation of path diagrams from lavaan fit objects using the **`semPlot`** package developed by Sacha Epskamp. There are likely other packages that could be used for the same purpose.

Some software programs also offer the ability to draw a path diagram and infer the model specification syntax from the diagram. For instance, the free stand-alone program Ωnyx can produce lavaan script based on a user-specified diagram (see http://onyx.brandmaier.de).

Example 1: Holzinger-Swineford CFA (Chapter 4)

The data for this demonstration were provided by Holzinger & Swineford in their 1939 monograph *A Study in Factor Analysis: The Stability of a Bi-Factor Solution*. The sample includes 301 7th and 8th grade students, between 11-16 years of age, drawn from two schools. The data is in the text file **hs.dat** and the script file is **app_ex1.R**. The variables in the data set that we will use are

visperc visual perception test in which participants select the next image in a series

cubes visual perception test in which participants must mentally rotate a cube

lozenges visual perception test involving mental "flipping" of a parallelogram ("lozenge")

parcomp paragraph comprehension test

sencomp sentence completion task in which participants select most appropriate word to put at the end of a sentence

wordmean verbal ability test in which participants must select a word most similar in meaning to a word used in a sentence.

addition participants have 2 minutes to complete as many 2-number addition problems as they can

countdot participants have 4 minutes to count the number of dots in each of a series of dot pictures

sccaps participants have 3 minutes to indicate whether capital letters are composed entirely of straight lines or include curved lines.

Other variables in the data not included in the models fit here are **school**, **female**, **age**, and **month**. We can read in the data with the following:

```
hs <- read.table("hs.dat", header=FALSE)
names(hs) <- c("school", "female", "age", "month",
               "visperc", "cubes", "lozenges",
               "parcomp", "sencomp", "wordmean",
               "addition", "countdot", "sccaps")
hs$addition <- hs$addition/4
hs$countdot <- hs$countdot/4
hs$sccaps <- hs$sccaps/4
```

As described in Chapter 4, we rescaled the last three variables to give them a similar range to the other indicators.

We will also load the two packages we need (install **semPlot** first if you have not already):

```
library(lavaan)
library(semPlot)
```

The initial model we fit to this data posited three factors, **visual** (with indicators **visperc**, **cubes** and **lozenges**), **verbal** (with indicators **parcomp**, **sencomp**, and **wordmean**), and **speed** (with indicators **addition**, **countdot**, and **sccaps**). We scaled the latent variables by

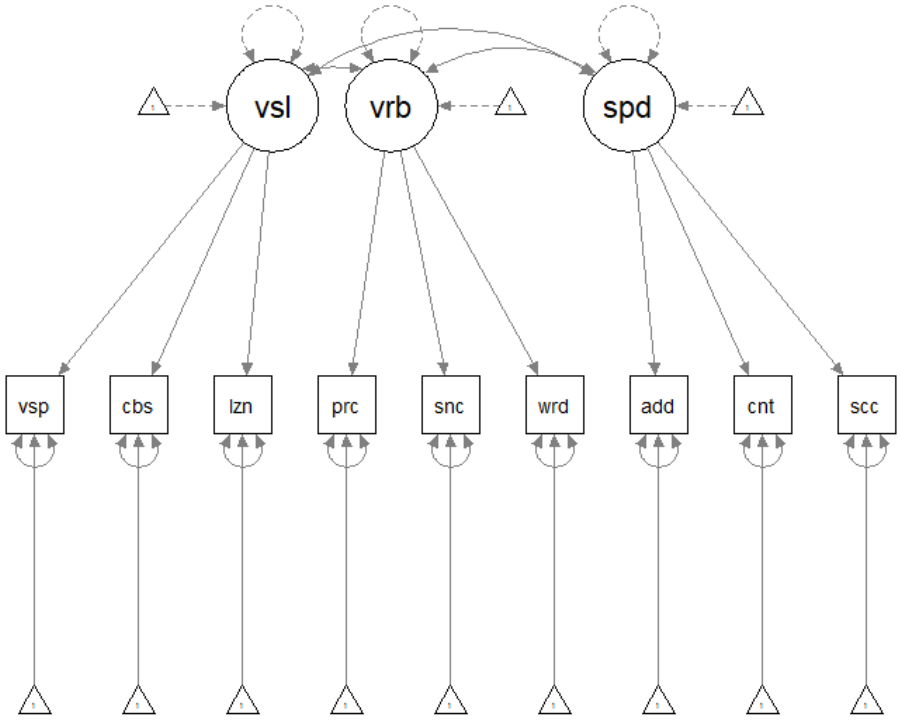
setting their means to zero and variances to one, permitting all factor loadings and intercepts to be freely estimated. The corresponding script for specifying and fitting this model in lavaan is

```
cfa.1a <-'#factor loadings
          visual =~ visperc + cubes + lozenges
          verbal =~ parcomp + sencomp + wordmean
          speed =~ addition + countdot + sccaps
         '
fit.1a <- cfa(cfa.1a, data=hs, meanstructure=TRUE, std.lv=TRUE)
```

We can now use the fit object **`fit.1a`** to generate a path diagram.

We first show the default diagram generated by the **`semPaths`** function:

```
semPaths(fit.1a)
```



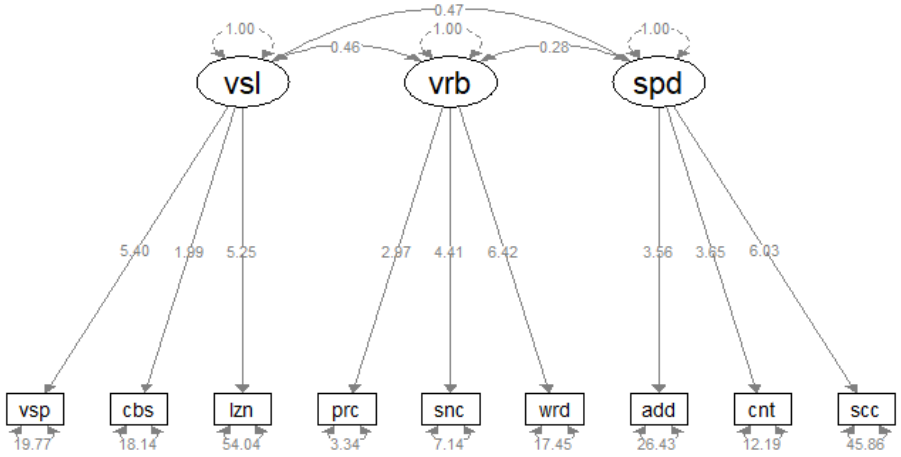
Clearly, some aspects of this plot can be improved upon. The triangles used to convey intercepts create a great deal of clutter and the covariances and variances among the latent variables are overlapping. Further, we might like to display the model estimates on the diagram for ease of interpretation.

Fortunately, there are many options for modifying the layout and information provided in the plot. Fiddling with these a bit, we invoked the following options:

```
semPaths(fit.1a, whatLabels="est", intercepts="FALSE", curve=1.75)
```

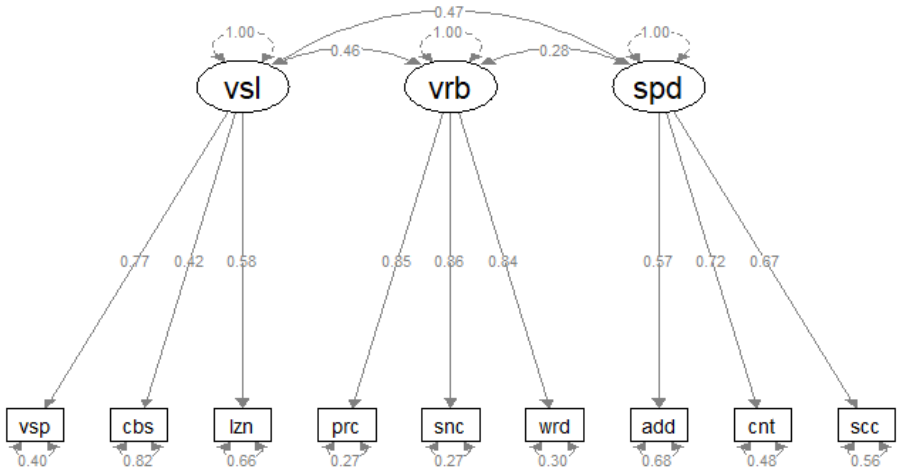
With these commands we will display the estimates (unstandardized) on the paths, remove the triangles for the intercepts, and increase the curvature of the double-headed arrows for the covariances so that they no longer overlap with the variances.

The resulting plot is shown here:



We can also switch the path labels to standardized estimates using the following options:

```
semPaths(fit.1a, whatLabels="std", intercepts="FALSE", curve=1.75)
```



Because the factors were already standardized, the displayed values for the latent variable variances and covariances are unchanged, but we now see standardized factor loadings and residuals (equal to one minus the communality for the indicator).

There are many other ways to modify the plot, as described in the documentation for the package.

Example 2: Conditional Latent Curve Model (Chapter 8)

The data for this demonstration are from the National Longitudinal Study of Youth. The sample includes 405 children who were aged 6-8 at the first assessment (in 1986). Children were assessed bi-yearly (until 1992), resulting in up to four measurement occasions per child. Data are in the file **anti.dat** and the script can be found in **app_ex2.R**. The variables in the data set that we will use are described below:

id	participant identification; this will not be used in the analysis
gen	0 = female; 1 = male
homecog	level of cognitive support in the home (mean-centered); higher values denote more cognitive support in the home
anti6-anti14	repeated assessments of mother reports of child antisocial behavior (higher values indicate more antisocial behavior); post-script denotes child age at time of assessment

We are interested in characterizing the functional form of developmental change in aggressive behavior for children aged 6-14. We would also like to understand how our time invariant covariates (gender and cognitive support in the home) contribute to initial levels of, and change in, aggressive behavior.

To read in the data, we use the following script:

```
classes <- rep("numeric",12)
anti <- read.table("anti.dat",header=FALSE,colClasses=classes,na.strings=".")
y <- paste("anti", 6:14, sep = "")
names(anti) <- c(c("id","gen","homecog"),y)
```

We then load the two packages we need:

```
library(lavaan)
library(semPlot)
```

For this example, we will fit a linear growth model including both **gen** and **homecog** as predictors (in Chapter 8, we considered these predictors separately, but here we include them jointly).

The model is specified and fit as follows:

```
lcm <- 'I =~ 1*anti6 + 1*anti7 + 1*anti8 + 1*anti9 +
             1*anti10 + 1*anti11 + 1*anti12 + 1*anti13 +
             1*anti14
        S =~ 0*anti6 + 1*anti7 + 2*anti8 + 3*anti9 +
             4*anti10 + 5*anti11 + 6*anti12 + 7*anti13 +
             8*anti14
        I ~ gen + homecog
        S ~ gen + homecog
       '
fit.lcm <- growth(lcm, data=anti, missing="ML")
summary(fit.lcm)
```

The results of primary interest, regarding the prediction of the growth factors, are shown here:

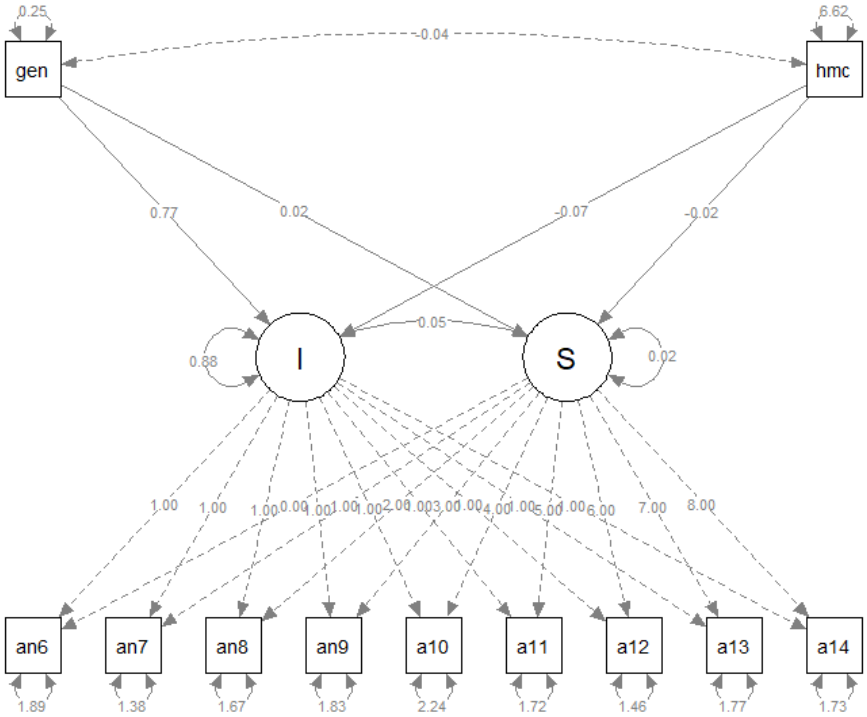
```
Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  I ~
    gen               0.768    0.165    4.664    0.000
    homecog          -0.073    0.032   -2.263    0.024
  S ~
    gen               0.015    0.035    0.443    0.658
    homecog          -0.015    0.007   -2.225    0.026
```

As in analyses which considered these predictors separately…

- Boys have higher intercepts, indicating higher levels of antisocial behavior at age six.
- There is no sex difference in slopes, such that the greater antisocial behavior of boys is maintained throughout the age range (through age 14).
- Kids from homes with higher cognitive support have both lower intercepts, i.e., lower antisocial behavior at age 6, and less steep slopes, i.e., increase in antisocial behavior less over time.

We can diagram these results as follows:

```
semPaths(fit.lcm, whatLabels="est", intercepts="FALSE")
```



Paths that are not estimated as part of the model fitting process are dashed (this includes fixed paths as well as variance/covariance parameters for exogenous predictors which are not explicitly estimated).

Again, this plot could be modified in a variety of ways that we do not pursue here.