Structural Equation Modeling *without* one of the commercial modeling software packages -- or, how you can use free software to impress your committee and/or colleagues.

By both Research and Statistical Support consultants[1]

Structural Equation Modeling (SEM) has become quite popular among the social science set over the last several years. However, unless one was a SAS user and aware of the PROC CALIS function in SAS; conducting SEM typically involved becoming proficient with one of the commercially available, modeling specific, software packages (e.g. EQS, Lisrel, AMOS, Mplus, etc.). That is no longer the case due to the growing popularity and functionality of the R programming language (Muenchen, 2010; Vance, 2009). Multiple libraries are available for conducting SEM (and other modeling techniques) in R. Two of those libraries will be discussed here.

The purpose of this month's article was to provide a demonstration of the basic functions for using R to conduct SEM. It was not our goal to *teach* SEM as that would require a great deal more time and space than is allocated here. However, we do cover some basic tenants of SEM below, such as sample size, number of manifest variables, and their relationship to overidentification, as well as the two stage approach. Suffice to say, SEM is a powerful data analysis tool for researchers and has gained popularity because it offers the ability to model measurement error, it accepts a variety of types of variables, and it can model a variety of different relationships between variables. These last two points can be summed up to say, SEM is extremely flexible in terms of the types of models which can be specified.

The general approach we tend to follow here at RSS when conducting SEM is the two stage approach advocated by Anderson and Gerbing (1988). The two stage approach consists of stage 1, verifying the measurement model and stage 2, testing the structural model. The measurement model is simply a confirmatory factor analysis to ensure you are measuring what you believe you are measuring. The structural model involves testing the theoretical causal relationships between *primarily* latent variables.

The fictional (simulated) data we will be using is available here: SEMData.sav. Once saved to your machine, the data can be imported into R, named exsem, and attached using the following code with slight changes to show the file path to where you saved the data on your machine:

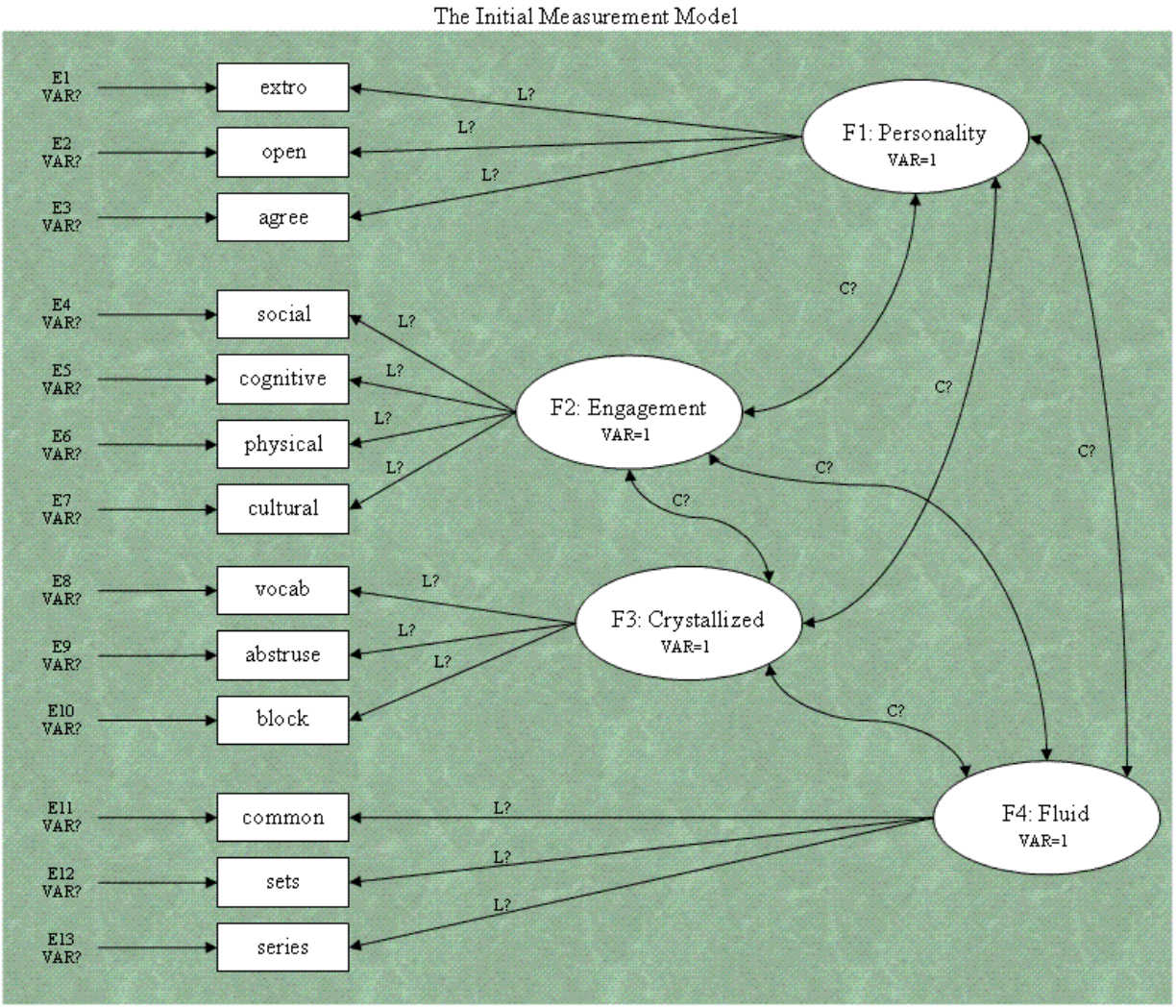
```
library(foreign)
exsem <- read.spss("C:/Documents and Settings/username/Desktop/SEMData.sav",
use.value.labels=TRUE, max.value.labels=Inf, to.data.frame=TRUE)
attach(exsem)
```

The data contains 750 observations on 13 variables: Extroversion (extro), Openness to Experience (open), Agreeableness (agree), Social Engagement (social), Cognitive Engagement (cognitive), Physical Engagement (physical), Cultural Engagement (cultural), Vocabulary (vocab), Abstruse Analogies (abstruse), Block Design (block), Common Analogies (common), Letter Sets (sets), and Letter Series (series). The data is used here to represent a fictional

longitudinal model of cognitive functioning in later life. Please note the liberal use of the word fictional in the preceding statements. This data was simulated for the use of demonstrating functions in R; it should not be relied upon as actual research data, it was simulated.

Stage 1: Verifying the Measurement Model.

Below you'll find a diagram which represents our measurement model. Notice we are unconcerned with the relationships between latent factors; those paths are free to vary (estimating the correlations or covariances) without specifying hypothetical causal relationships between them. Also notice, the variances of the factors are fixed at 1. Remember, the latent factors are unobserved and therefore we do not have an idea of their variance, nor their scale. Another thing to notice is that each observed score (i.e. manifest variable) is caused by a latent factor which we believe we are measuring indirectly, *and* measurement error. These error terms are shown in the diagram with arrows pointing toward the manifest variables as classical test theory suggests (i.e. observed score = true score + measurement error). This bears mentioning because, as stated above, it is one of the reasons SEM is so popular; it allows us to model measurement error.



One of the key requirements of SEM is overidentification. A model is said to be overidentified if it contains more unique inputs (sometimes called informations) than the number of parameters being estimated. In our example, we have 13 manifest variables. We can apply the following formula to calculate the number of unique inputs:

(1) $$\text{number of unique inputs} = (p\,(\,p + 1\,)\,) / 2$$

where p = the number of manifest variables. Given this formula and our 13 manifest variables; we calculate 91 unique inputs or informations which is greater than the number of parameters we are estimating. Looking at the diagram, we see 6 covariances (C?), 13 loadings (L?), and 13 error variances (VAR?). Adding these up, we get 32 parameters to be estimated. You'll notice that for our measurement model, we have specified the variance of the latent factors to be 1 (VAR=1). This is done to allow estimation of all the factor loadings. Remember too that SEM requires large sample sizes. Several general rules have been put forth as lowest reasonable sample size estimates; at least 200 cases at a minimum, at least 5 cases per manifest or measured variable, at least 400 cases, at least 25 cases per measured variable, 5 observations or cases per parameter to be estimated, 10 observations or cases per parameter to be estimated...etc. The bottom line is this; SEM is powerful when done with adequately large samples -- the larger the better. Another issue related to sample size, is the recommendation of having at least 3 manifest variables for each latent factor; with the suggestion of having 4 or more manifest variables for each latent factor (Anderson & Gerbing, 1988). Having 4 allows you the flexibility of deleting one if you find it does not contribute meaningfully to a latent factor or the model in general (e.g. it loads on more than one factor to a meaningful extent). Another consideration is that of remaining realistic when setting out to study particular phenomena with SEM in mind as the analysis. It is often easy to develop some very complex models containing a great number of manifest variables. However, complex models containing more than 20 manifest variables can lead to confusion in interpretation and a lack of fit, as well as convergence difficulty. Bentler and Chou (1987) recommend a limit of 20 manifest variables.

Through out the rest of this article, we will be using two libraries to run the SEM on our example data. The sem library contributed by John Fox and the lavaan library contributed by Yves Rosseel. The sem library has existed for a few years now and offers “functions for fitting general linear structural equation models (with observed and unobserved variables) by the method of maximum likelihood using the RAM approach, and for fitting structural equations in observed-variable models by two-stage least squares” (Fox, 2010). The lavaan library is a relatively new package (May, 2010) which was created to make it easier for new R users to conduct latent variable modeling (e.g. confirmatory factor analysis, SEM, & latent growth curve models). The primary benefit of the lavaan library is the intuitive way in which models are specified. As you will see in the code below, the RAM approach used in the sem library tends to necessitate many more lines of code than is necessary to conduct the same model in lavaan. As an example of lavaan’s economy of code, consider the examples given in the lavaan users manual (Rosseel, 2010) for each type of linear specification.

```
# Regression or path equations
  y  ~ F1 + F2 + x1 + x2
  F1 ~ F2 + F3
  F2 ~ F3 + x1 + x2
```

```
# Latent variable definitions
  F1 =~ y1 + y2 + y3
  F2 =~ y4 + y5 + y6
  F3 =~ y7 + y8 + y9 + y10
# Variances and covariances
  y1 ~~ y1
  y1 ~~ y2
  F1 ~~ F2
# Intercepts
  y1 ~ 1
  F1 ~ 1
```

Again, you will see below how economical the lavaan library is when compared with the sem library; but, either can be used and the sem library offers a few more diagnostic functions than the lavaan library.

Using the sem library

Using the sem library to verify the measurement model on our example sem data (exsem), we first need to load the library, then create an object of the covariance matrix.

```
library(sem)
cov.sem <- cov(exsem)
```

Next, we need to specify the measurement model. Here, you will see the familiar RAM style specification which requires many lines.

```
measurement.model.1 <- specify.model()
personality -> extro, load11, NA
personality -> open, load12, NA
personality -> agree, load13, NA
engagement -> social, load14, NA
engagement -> cognitive, load21, NA
engagement -> physical, load22, NA
engagement -> cultural, load23, NA
crystallized -> vocab, load31, NA
crystallized -> abstruse, load32, NA
crystallized -> block, load33, NA
fluid -> common, load41, NA
fluid -> sets, load42, NA
fluid -> series, load43, NA
extro <-> extro, evar1, NA
open <-> open, evar2, NA
agree <-> agree, evar3, NA
social <-> social, evar4, NA
cognitive <-> cognitive, evar5, NA
physical <-> physical, evar6, NA
cultural <-> cultural, evar7, NA
vocab <-> vocab, evar8, NA
abstruse <-> abstruse, evar9, NA
block <-> block, evar10, NA
common <-> common, evar11, NA
sets <-> sets, evar12, NA
```

```
series <-> series, evar13, NA
personality <-> engagement, cov1, NA
personality <-> crystallized, cov2, NA
personality <-> fluid, cov3, NA
engagement <-> crystallized, cov4, NA
engagement <-> fluid, cov5, NA
crystallized <-> fluid, cov6, NA
personality <-> personality, NA, 1
engagement <-> engagement, NA, 1
crystallized <-> crystallized, NA, 1
fluid <-> fluid, NA, 1
```

Next, we can run the sem and assign the measurement model to an sem object (here called m.model.1). Then, we can get a summary of that object with 95% confidence intervals, as well as extracting the standardized coefficients from that sem object.

```
m.model.1 <- sem(measurement.model.1, cov.sem, 750, maxiter = 10000)
summary(m.model.1, conf.level=0.95)
standardized.coefficients(m.model.1)
```

We can also use that sem object to get a summary of the residuals and plot a histogram of those residuals.

```
summary(residuals(m.model.1))
hist(residuals(m.model.1))
```

Using the lavaan library

First, we need to detach the sem library[2], then we can load the lavaan library (this is necessary because both have an 'sem' function).

```
detach("package:sem")
library(lavaan)
```

Next, we need to specify the measurement model, but we do not need to use the RAM format and instead use a more linear-equation-like format.

```
measurement.model.2 <- '
Personality  =~ extro + open + agree
Engagement   =~ social + cognitive + physical + cultural
Crystallized =~ vocab + abstruse + block
Fluid        =~ common + sets + series
'
```

Next, we can fit the measurement model (i.e. confirmatory factor analysis) while assigning it to an object (here called m.model.2). The 'std.lv = TRUE' command sets the variance of all the latent variables to 1. The second line below provides a summary of the model. The 'fit.measures = TRUE' command provides additional fit indices; by default only the chi-square is given. The 'standardize = TRUE' command provides standardized coefficients/loadings.

```
m.model.2 <- cfa(measurement.model.2, data = exsem, std.lv = TRUE)
```

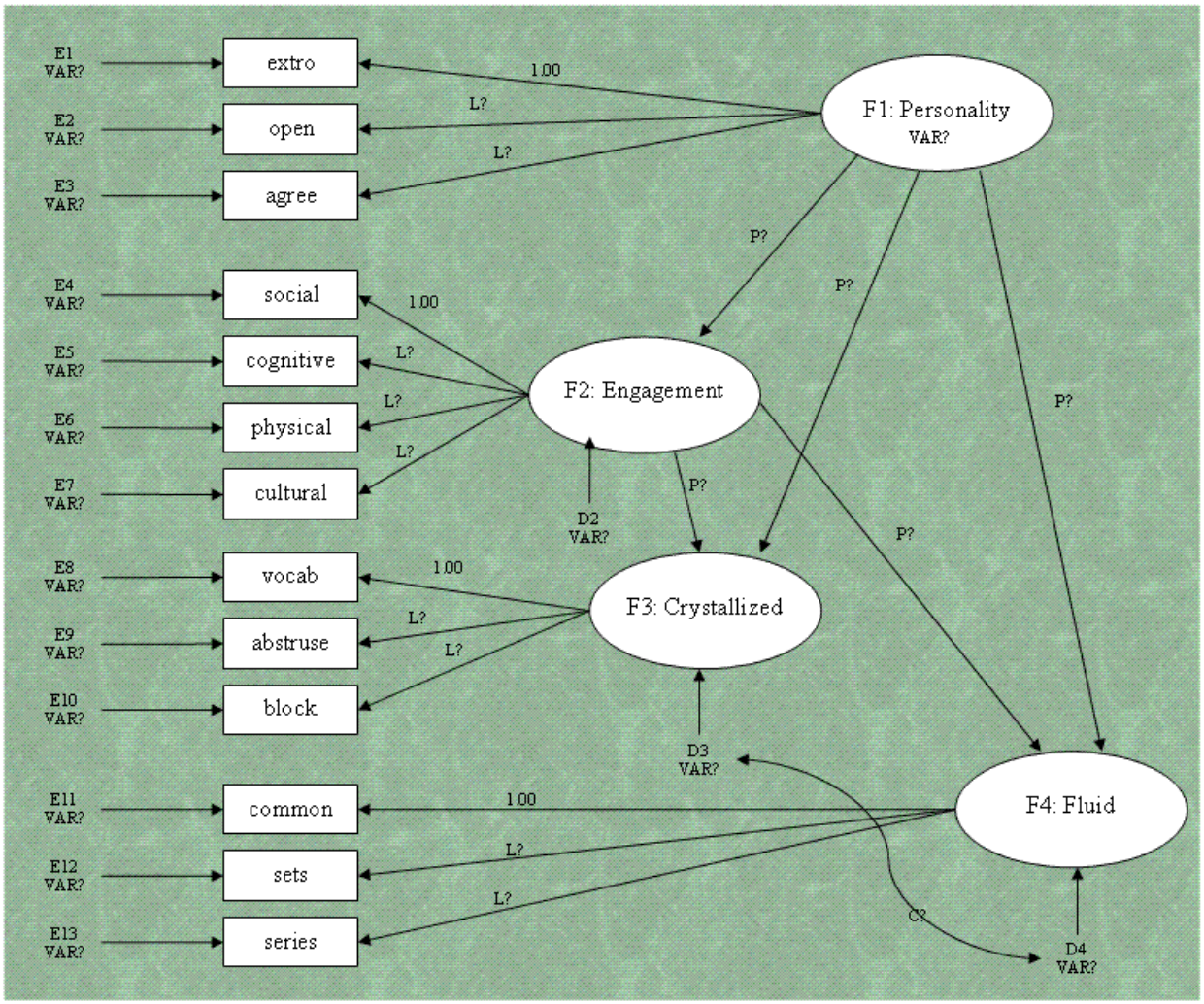
```
summary(m.model.2, fit.measures = TRUE, standardize = TRUE)
```

Notice the coefficients are all the same when comparing the sem library output to the lavaan library output.

Stage 2: Testing the Structural Model.

Below you'll find a diagram which represents our structural model. Notice we are now concerned with the theoretical causal relationships between latent factors, those paths will be estimated. Also notice, the variances of the factors are no longer fixed at 1.00. This time, we must fix one of loadings from each of the factors to 1 in order to set the scale of the factors and estimate their variances (and disturbance terms). Often the first or the largest loading gets fixed to 1.00. Disturbance terms (D2, D3, D4) are new here and represent the error associated with each factor which can be thought of as any causal influence for that factor which was not included in the model. Also notice there is no disturbance term specified for our one exogenous factor (Personality); exogenous meaning, all of its causal influences lie *outside* the specified model. Disturbance terms are only specified for endogenous factors.

The Structural Model



The structural model has the same number of unique informations and the same number of parameters as did the measurement model and is therefore, overidentified. That may not always be the case so it is often a good idea to create a diagram (or even simply a drawing) to identify the informations and parameters to be estimated and ensure overidentification.

Using the sem library

Using the sem library to test the structural model is very similar to what was done with the measurement model. The RAM specification has only a few changes to incorporate the estimation of the factor variances and factor paths. But first, we need to detach the lavaan library and attach or load the sem library.

```
detach("package:lavaan")
library(sem)
```

Next, we can specify the structural model.

```
structural.model.1 <- specify.model()
personality -> extro, NA, 1
personality -> open, load12, NA
personality -> agree, load13, NA
engagement -> social, NA, 1
engagement -> cognitive, load22, NA
engagement -> physical, load23, NA
engagement -> cultural, load24, NA
crystallized -> vocab, NA, 1
crystallized -> abstruse, load32, NA
crystallized -> block, load33, NA
fluid -> common, NA, 1
fluid -> sets, load42, NA
fluid -> series, load43, NA
extro <-> extro, evar1, NA
open <-> open, evar2, NA
agree <-> agree, evar3, NA
social <-> social, evar4, NA
cognitive <-> cognitive, evar5, NA
physical <-> physical, evar6, NA
cultural <-> cultural, evar7, NA
vocab <-> vocab, evar8, NA
abstruse <-> abstruse, evar9, NA
block <-> block, evar10, NA
common <-> common, evar11, NA
sets <-> sets, evar12, NA
series <-> series, evar13, NA
personality -> engagement, path1, NA
personality -> crystallized, path2, NA
personality -> fluid, path3, NA
engagement -> crystallized, path4, NA
engagement -> fluid, path5, NA
crystallized <-> fluid, cov1, NA
personality <-> personality, fvar1, NA
engagement <-> engagement, dist1, NA
crystallized <-> crystallized, dist2, NA
```

```
fluid <-> fluid, dist3, NA
```

Next, we can test the structural model by passing the covariance matrix from the beginning, and the structural model from just above, to the sem function and assigning it to an sem object (here named s.model.1). Then, we can get a summary of that object with 95% confidence intervals, as well as extracting the standardized coefficients from that sem object as was done previously with the measurement model.

```
s.model.1 <- sem(structural.model.1, cov.sem, 750, maxiter = 10000)
summary(s.model.1, conf.level=0.95)
standardized.coefficients(s.model.1)
```

Then, we can also evaluate the residuals as was done previously with the measurement model.

```
summary(residuals(s.model.1))
hist(residuals(s.model.1))
```

Using the lavaan library

First, we need to detach the sem library, then we can load the lavaan library.

```
detach("package:sem")
library(lavaan)
```

Next, we need to specify the measurement model. Keep in mind that by default, the lavaan library cfa and sem functions constrain the first loading to 1.00.

```
structural.model.2 <- '
Personality  =~ extro + open + agree
Engagement   =~ social + cognitive + physical + cultural
Crystallized =~ vocab + abstruse + block
Fluid        =~ common + sets + series
Engagement   ~ Personality
Crystallized ~ Engagement + Personality
Fluid        ~ Engagement + Personality
Crystallized ~~ Fluid
'
```

Next, we can fit the structural model. Since we are testing the structural model and because of the default constraints, we do not need the extra command (std.lv = TRUE) which constrains all latent variable variances to 1.00 – because, by default the first loading of each latent variable is automatically constrained to 1.00 with the lavaan library functions.

```
s.model.2 <- sem(structural.model.2, data = exsem)
summary(s.model.2, fit.measures = TRUE, standardize = TRUE)
```

Notice the coefficients are all the same when comparing the sem library output to the lavaan library output; in fact, they are virtually identical with only rounding differences present.

Until next time, *I'll let you be in my dreams if I can be in yours*.

Footnotes

[1] Neither of us really cared who would be first author, so the solution was to make it ambiguous. For those who can not stand ambiguity, Jon did much of the writing and Rich did the technical work of embedding and refining the code so that it could be passed from your browser to a server running R, to produce output back in your browser.

[2] Pasting script from MS-Word into R (either the console or an open script window) will generally result in the script working properly. However, the quotation marks here do not work properly and therefore; you should simply type the commands in R which use quotation marks while working through these examples. The only commands which are affected are the detach commands because they are the only commands in this article which rely on quotation marks.

References & Resources

Anderson, J. C., & Gerbing, D. W. (1988). Structural equation modeling in practice: A review and recommended two-step approach. *Psychological Bulletin, 103*, 411 – 423. DOI: 10.1037/0033-2909.103.3.411

Bentler, P. M., & Chou, C. (1987). Practical issues in structural modeling. *Sociological Methods & Research, 16*, 78 – 117. DOI: 10.1177/0049124187016001004

Fox, J. (2010). sem library documentation manual. Retrieved on August 31, 2010 from http://cran.r-project.org/web/packages/sem/index.html

Rosseel, Y. (2010). lavaan: An R package for structural equation modeling and more Version 0.3-1 (BETA). Retrieved on August 31, 2010 from http://cran.r-project.org/web/packages/lavaan/index.html

Muenchen, R. A. (2010). The popularity of data analysis software. Retrieved on August 31, 2010 from http://sites.google.com/site/r4statistics/popularity

Vance, A. (2009). Data analysts captivated by R's power. Retrieved on August 31, 2010 from http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html