Structural Equation Modeling in R with the **sem** Package

An Appendix to *An R Companion to Applied Regression, Second Edition*
by John Fox and Sanford Weisberg

John Fox

last revision: 25 September 2012

Abstract

Structural equation models (SEMs) are multi-equation regression models. Unlike the more traditional multivariate linear model, however, the response variable in one regression equation in an SEM may appear as a predictor in another equation, and variables in an SEM may influence one-another reciprocally, either directly or through other variables as intermediaries.

This appendix to Fox and Weisberg (2011) describes how to use the **sem** package to fit a variety of linear structural equation models to data, including general structural equation models with latent variables.

1 Introduction

Structural equation models (*SEMs*), also called *simultaneous equation models*, are multivariate (i.e., multi-equation) regression models. Unlike the more traditional multivariate linear model, however, the response variable in one regression equation in an SEM may appear as a predictor in another equation; indeed, variables in an SEM may influence one-another reciprocally, either directly or through other variables as intermediaries. These *structural equations* are meant to represent causal relationships among the variables in the model.

A cynical view of SEMs is that their popularity in the social sciences reflects the legitimacy that the models appear to lend to causal interpretation of observational data, when in fact such interpretation is no less problematic than for other kinds of regression models applied to observational data.[1] A more charitable interpretation is that SEMs are close to the kind of informal thinking about causal relationships that is common in social-science theorizing, and that, therefore, these models facilitate translating such theories into data analysis. In economics, in contrast, structural-equation models may stem from *formal* theory.

This appendix briefly describes how to use the **sem** package (Fox et al., 2012) to fit a variety of linear structural equations models in R, including two-stage least-squares estimation of nonrecursive observed-variable models, maximum-likelihood estimation of general, latent-variable structural-equation models, and some other methods. The current version of the **sem** package uses compiled C++ code to immprove the computational efficiency of key calculations.

In addition to the **sem** package, the **systemfit** package (Henningsen and Hamann, 2007), available from the Comprehensive R Archive Network (CRAN), implements a variety of estimators for observed-variable structural equation models, and the **lavaan** package (Rosseel, 2012), also on CRAN, implements methods for estimating latent-variable models. The **OpenMx** package for R

[1] For an extreme version of the argument, with which I have some (if not complete) sympathy, see Freedman (1987), and the ensuing discussion.

is broadly capable structural-equation-modeling software; this package is not currently on CRAN because of licensing issues, is available from <`http://openmx.psyc.virginia.edu/`>.

I assume that the reader is generally familiar with structural equation models. Some references are given in the concluding section (Section 4).

2 Observed-Variables Models and Two-Stage Least-Squares Estimation

2.1 An Example: Klein's Model

Klein's macroeconomic model of the U.S. economy (Klein, 1950) often appears in econometrics texts (e.g., Greene, 2003) as a simple example of a structural equation model:

$$\begin{aligned}
C_t &= \gamma_{10} + \beta_{11}P_t + \gamma_{11}P_{t-1} + \beta_{12}(W_t^p + W_t^g) + \zeta_{1t} \\
I_t &= \gamma_{20} + \beta_{21}P_t + \gamma_{21}P_{t-1} + \gamma_{22}K_{t-1} + \zeta_{2t} \\
W_t^p &= \gamma_{30} + \gamma_{31}A_t + \beta_{31}X_t + \gamma_{32}X_{t-1} + \zeta_{3t} \\
X_t &= C_t + I_t + G_t \\
P_t &= X_t - T_t - W_t^p \\
K_t &= K_{t-1} + I_t
\end{aligned} \tag{1}$$

- The variables on the left-hand side of the structural equations are *endogenous variables* — that is, variables whose values are determined by the model. There is, in general, one structural equation for each endogenous variable in an SEM.[2]

- The ζs (Greek *zeta*) are error variables, also called *structural disturbances* or *errors in equations*; they play a role analogous to the error in a single-equation regression model. It is not generally assumed that different disturbances are independent of one-another, although such assumptions are sometimes made in particular models.[3]

- The remaining variables on the right-hand side of the model are *exogenous variables*, whose values are treated as conditionally fixed; an additional defining characteristic of exogenous variables is that they are assumed to be independent of the errors (much as the predictors in a common regression model are taken to be independent of the error). *Lagged endogenous ("predetermined") variables*, such as P_{t-1} are also independent of the errors ζ_{jt} and so are effectively exogenous.

- The γs (Greek *gamma*) are structural parameters (regression coefficients) relating the endogenous variables to the exogenous variables (including an implicit constant regressor for each of the first three equations) and predetermined endogenous variables.

- Similarly, the βs (Greek *beta*) are structural parameters relating the endogenous variables to one-another.

- The last three equations have no error variables and no structural parameters. These equations are *identities*, and could be substituted out of the model. Our task is to estimate the first three equations, which contain unknown parameters.

[2]Some forms of structural equation models do not require that one endogenous variable in each equation be identified as the response variable.

[3]See, for example, the discussion of recursive models below.

The variables in model (1) have the following definitions:

C_t	Consumption (in year t)
I_t	Investment
W_t^p	Private wages
X_t	Equilibrium demand
P_t	Private profits
K_t	Capital stock
G_t	Government non-wage spending
T_t	Indirect business taxes and net exports
W_t^g	Government wages
A_t	Time trend, year $-$ 1931

The use of the subscript t for observations reflects the fact that Klein estimated the model with annual time-series data for the years 1921 through 1941.[4] Klein's data are in the data frame `Klein` in the **sem** package:

```
> library(sem)
> Klein

   Year    C    P   Wp    I K.lag    X  Wg    G    T
1  1920 39.8 12.7 28.8  2.7 180.1 44.9 2.2  2.4  3.4
2  1921 41.9 12.4 25.5 -0.2 182.8 45.6 2.7  3.9  7.7
3  1922 45.0 16.9 29.3  1.9 182.6 50.1 2.9  3.2  3.9
4  1923 49.2 18.4 34.1  5.2 184.5 57.2 2.9  2.8  4.7
5  1924 50.6 19.4 33.9  3.0 189.7 57.1 3.1  3.5  3.8
6  1925 52.6 20.1 35.4  5.1 192.7 61.0 3.2  3.3  5.5
7  1926 55.1 19.6 37.4  5.6 197.8 64.0 3.3  3.3  7.0
8  1927 56.2 19.8 37.9  4.2 203.4 64.4 3.6  4.0  6.7
9  1928 57.3 21.1 39.2  3.0 207.6 64.5 3.7  4.2  4.2
10 1929 57.8 21.7 41.3  5.1 210.6 67.0 4.0  4.1  4.0
11 1930 55.0 15.6 37.9  1.0 215.7 61.2 4.2  5.2  7.7
12 1931 50.9 11.4 34.5 -3.4 216.7 53.4 4.8  5.9  7.5
13 1932 45.6  7.0 29.0 -6.2 213.3 44.3 5.3  4.9  8.3
14 1933 46.5 11.2 28.5 -5.1 207.1 45.1 5.6  3.7  5.4
15 1934 48.7 12.3 30.6 -3.0 202.0 49.7 6.0  4.0  6.8
16 1935 51.3 14.0 33.2 -1.3 199.0 54.4 6.1  4.4  7.2
17 1936 57.7 17.6 36.8  2.1 197.7 62.7 7.4  2.9  8.3
18 1937 58.7 17.3 41.0  2.0 199.8 65.0 6.7  4.3  6.7
19 1938 57.5 15.3 38.2 -1.9 201.8 60.9 7.7  5.3  7.4
20 1939 61.6 19.0 41.6  1.3 199.9 69.5 7.8  6.6  8.9
21 1940 65.0 21.1 45.0  3.3 201.2 75.7 8.0  7.4  9.6
22 1941 69.7 23.5 53.3  4.9 204.5 88.4 8.5 13.8 11.6
```

Some of the variables in Klein's model have to be constructed from the data:

[4]Estimating a structural equation model for time-series data raises the issue of autocorrelated errors, as it does in regression models fit to time-series data (described in the Appendix on time-series regression). Although I will not address this complication, there are methods for accommodating autocorrelated errors in structural equation models; see, e.g., Greene (2003, Sec. 15.9).

```
> Klein$P.lag <- with(Klein, c(NA, P[-length(P)]))
> Klein$X.lag <- with(Klein, c(NA, X[-length(X)]))
> Klein$A <- Klein$Year - 1931
> head(Klein)

  Year    C    P   Wp    I K.lag    X  Wg   G   T P.lag X.lag   A
1 1920 39.8 12.7 28.8  2.7 180.1 44.9 2.2 2.4 3.4    NA    NA -11
2 1921 41.9 12.4 25.5 -0.2 182.8 45.6 2.7 3.9 7.7  12.7  44.9 -10
3 1922 45.0 16.9 29.3  1.9 182.6 50.1 2.9 3.2 3.9  12.4  45.6  -9
4 1923 49.2 18.4 34.1  5.2 184.5 57.2 2.9 2.8 4.7  16.9  50.1  -8
5 1924 50.6 19.4 33.9  3.0 189.7 57.1 3.1 3.5 3.8  18.4  57.2  -7
6 1925 52.6 20.1 35.4  5.1 192.7 61.0 3.2 3.3 5.5  19.4  57.1  -6
```

Notice, in particular how the lagged variables P_{t-1} and X_{t-1} are created by shifting P_t and X_t forward one time period — placing an `NA` at the beginning of each variable, and dropping the last observation. The first observation for P_{t-1} and X_{t-1} is missing because there are no data available for P_0 and X_0.

Estimating Klein's model is complicated by the presence of endogenous variables on the right-hand side of the structural equations. In general, we cannot assume that an endogenous predictor is uncorrelated with the error variable in a structural equation, and consequently ordinary least-squares (OLS) regression cannot be relied upon to produce consistent estimates of the parameters of the equation. For example, the endogenous variable P_t appears as a predictor in the first structural equation, for C_t; but X_t is a component of P_t, and X_t, in turn, depends upon C_t, one of whose components is the error ζ_{1t}. Thus, indirectly, ζ_{1t} is a component of P_t, and the two are likely correlated. Similar reasoning applies to the other endogenous predictors in the model, as a consequence of the simultaneous determination of the endogenous variables.

2.2 Identification and Instrumental-Variables Estimation

Instrumental-variables estimation provides consistent estimates of the parameters of a structural equation. An *instrumental variable* (also called an *instrument*) is a variable uncorrelated with the error of a structural equation. In the present context, the exogenous variables can serve as instrumental variables, as can predetermined endogenous variables, such as P_{t-1}.

Let us write a structural equation of the model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\delta} + \boldsymbol{\zeta} \tag{2}$$

where $\mathbf{y}$ is the $n \times 1$ vector for the response variable in the equation; $\mathbf{X}$ is an $n \times p$ model matrix, containing the p endogenous and exogenous predictors for the equation, normally including a column of 1s for the constant; $\boldsymbol{\delta}$ (Greek *delta*) is the $p \times 1$ parameter vector, containing the γs and βs for the structural equation; and $\boldsymbol{\zeta}$ is the $n \times 1$ error vector. Let the $n \times p$ matrix $\mathbf{Z}$ contain instrumental variables (again, normally including a column of 1s). Then, multiplying the structural equation through by $\mathbf{Z}'$ produces

$$\mathbf{Z}'\mathbf{y} = \mathbf{Z}'\mathbf{X}\boldsymbol{\delta} + \mathbf{Z}'\boldsymbol{\zeta}$$

In the probability limit, $\frac{1}{n}\mathbf{Z}'\boldsymbol{\zeta}$ goes to $\mathbf{0}$ because of the uncorrelation of the instrumental variables with the error. The instrumental-variables estimator

$$\widehat{\boldsymbol{\delta}} = (\mathbf{Z}'\mathbf{X})^{-1}\mathbf{Z}'\mathbf{y}$$

is therefore a consistent estimator of $\boldsymbol{\delta}$.

I have implicitly assumed two things here: (1) that the number of instrumental variables is equal to the number of predictors p in the structural equation; and (2) that the cross-products matrix $\mathbf{Z}'\mathbf{X}$ is nonsingular.

- If there are *fewer* instrumental variables than predictors (i.e., structural coefficients), then the estimating equations
$$\mathbf{Z}'\mathbf{y} = \mathbf{Z}'\mathbf{X}\widehat{\boldsymbol{\delta}}$$
are under-determined, and the structural equation is said to be *under-identified.*[5]

- If there are p instrumental variables, then the structural equation is said to be *just-identified.*

- If there are *more* instrumental variables than predictors, then the estimating equations will almost surely be over-determined, and the structural equation is said to be *over-identified.*[6] What we have here is an embarrassment of riches, however: We could obtain consistent estimates simply by discarding surplus instrumental variables. To do so would be statistically profligate, however, and there are better solutions to over-identification, including the method of two-stage least squares, to be described presently.

- For $\mathbf{Z}'\mathbf{X}$ to be nonsingular, the instrumental variables must be correlated with the predictors, and we must avoid perfect collinearity.

2.3 Two-Stage Least Squares Estimation

Two-stage least squares (*2SLS*) is so named because it can be thought of as the catenation of two OLS regressions:

1. In the first stage, the predictors $\mathbf{X}$ are regressed on the instrumental variables $\mathbf{Z}$, obtaining fitted values[7]
$$\widehat{\mathbf{X}} = \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}$$

2. In the second stage, the response $\mathbf{y}$ is regressed on the fitted values from the first stage, $\widehat{\mathbf{X}}$, producing the 2SLS estimator of $\boldsymbol{\delta}$:
$$\widehat{\boldsymbol{\delta}} = (\widehat{\mathbf{X}}'\widehat{\mathbf{X}})^{-1}\widehat{\mathbf{X}}'\mathbf{y}$$
This is justified because as linear combinations of the instrumental variables, the columns of $\widehat{\mathbf{X}}$ are (in the probability limit) uncorrelated with the structural disturbances. An alternative, but equivalent, approach to the second stage is to apply the fitted values from the first stage, $\widehat{\mathbf{X}}$, as instrumental variables to the structural equation (2):[8]
$$\widehat{\boldsymbol{\delta}} = (\widehat{\mathbf{X}}'\mathbf{X})^{-1}\widehat{\mathbf{X}}'\mathbf{y}$$

[5]That there must be at least as many instrumental variables as coefficients to estimate in a structural equation is called the *order condition for identification.* It turns out that the order condition is a necessary, but not sufficient, condition for identification. Usually, however, a structural equation model that satisfies the order condition is identified. See the references cited in Section 4.

[6]This over-determination is a product of sampling error, because presumably in the population the estimating equations would hold precisely and simultaneously. If the estimating equations are highly inconsistent, that casts doubt upon the specification of the model.

[7]Columns of $\mathbf{X}$ corresponding to exogenous predictors are simply reproduced in $\widehat{\mathbf{X}}$, because the exogenous variables are among the instrumental variables in $\mathbf{Z}$ — that is, the exogenous predictors are in the column space of $\mathbf{Z}$.

[8]Obviously, for the two approaches to be equivalent, it must be the case that $\widehat{\mathbf{X}}'\widehat{\mathbf{X}} = \widehat{\mathbf{X}}'\mathbf{X}$. Can you see why this equation holds?

The two stages of 2SLS can be combined algebraically, producing the following expression for the estimates:

$$\widehat{\boldsymbol{\delta}} = [\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}]^{-1}\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y}$$

The estimated asymptotic covariance matrix of the coefficients is

$$\widehat{\mathcal{V}}(\widehat{\boldsymbol{\delta}}) = s^2[\mathbf{X}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{X}]^{-1}$$

where s^2 is the estimated error variance for the structural equation,

$$s^2 = \frac{(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\delta}})'(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\delta}})}{n - p}$$

that is, the sum of squared residuals divided by residual degrees of freedom.[9]

To apply 2SLS to the structural equations in Klein's model, we may use the four exogenous variables, the constant, and the three predetermined endogenous variables as instruments. Because there are therefore eight instrumental variables and only four structural parameters to estimate in each equation, the three structural equations are all over-identified.

The `tsls` function in the **sem** package performs 2SLS estimation:

- The structural equation to be estimated is specified by a model formula, as for `lm` (see Chapter 4 of Fox and Weisberg, 2011).
- The instrumental variables are supplied in a *one-sided model formula* via the `instruments` argument
- There are optional `data`, `subset`, `na.action`, `weights`, and `contrasts` arguments that work just like those in `lm` (and which are, again, described in Chapter 4 of the text).
- The `tsls` function returns an object of class `"tsls"`. A variety of methods exist for objects of this class, including `print`, `summary`, `fitted`, `residuals`, `anova`, `coef`, and `vcov` methods. For details, enter `help(tsls)`.

For example, to estimate the structural equations in Klein's model:

```
> eqn.1 <- tsls(C ~ P + P.lag + I(Wp + Wg),
+     instruments= ~ G + T + Wg + A + P.lag + K.lag + X.lag, data=Klein)
> summary(eqn.1)

 2SLS Estimates

Model Formula: C ~ P + P.lag + I(Wp + Wg)

Instruments: ~G + T + Wg + A + P.lag + K.lag + X.lag

Residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -1.890  -0.616  -0.246   0.000   0.885   2.000
```

[9]Because the result is asymptotic, a less conservative alternative is to divide the residual sum of squares by n rather than by $n - p$.

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 16.55476    1.46798  11.277 2.59e-09
P            0.01730    0.13120   0.132   0.8966
P.lag        0.21623    0.11922   1.814   0.0874
I(Wp + Wg)   0.81018    0.04474  18.111 1.51e-12

Residual standard error: 1.1357 on 17 degrees of freedom

> eqn.2 <- tsls(I ~ P + P.lag + K.lag,
+     instruments= ~ G + T + Wg + A + P.lag + K.lag + X.lag, data=Klein)
> summary(eqn.2)

 2SLS Estimates

Model Formula: I ~ P + P.lag + K.lag

Instruments: ~G + T + Wg + A + P.lag + K.lag + X.lag

Residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.290  -0.807   0.142   0.000   0.860   1.800

            Estimate Std. Error t value Pr(>|t|)
(Intercept) 20.27821    8.38325   2.419  0.02707
P            0.15022    0.19253   0.780  0.44598
P.lag        0.61594    0.18093   3.404  0.00338
K.lag       -0.15779    0.04015  -3.930  0.00108

Residual standard error: 1.3071 on 17 degrees of freedom

> eqn.3 <- tsls(Wp ~ X + X.lag + A,
+     instruments= ~ G + T + Wg + A + P.lag + K.lag + X.lag, data=Klein)
> summary(eqn.3)

 2SLS Estimates

Model Formula: Wp ~ X + X.lag + A

Instruments: ~G + T + Wg + A + P.lag + K.lag + X.lag

Residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.2900 -0.4730  0.0145  0.0000  0.4490  1.2000

            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.50030    1.27569   1.176 0.255774
X            0.43886    0.03960  11.082 3.37e-09
X.lag        0.14667    0.04316   3.398 0.003422
A            0.13040    0.03239   4.026 0.000876
```

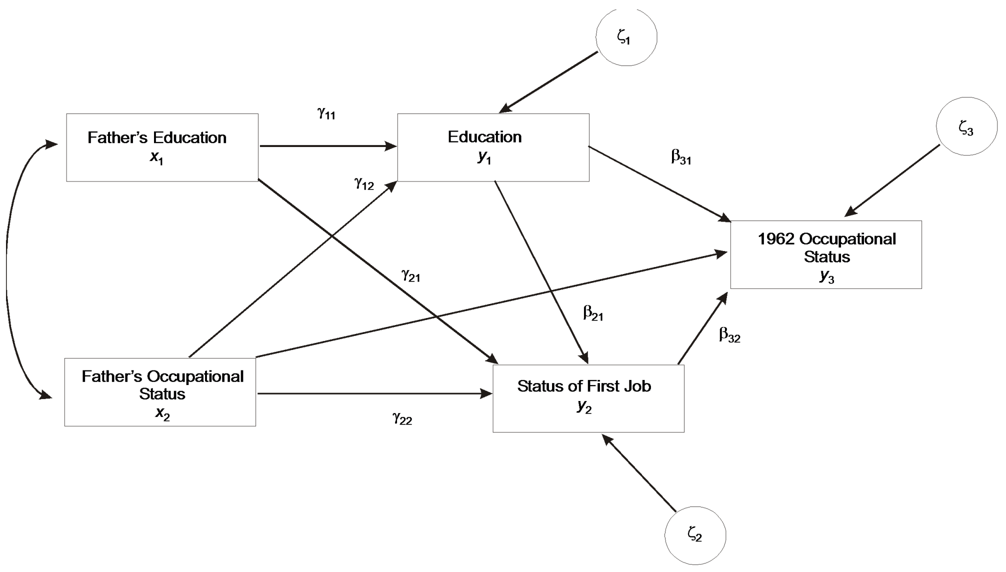


Figure 1: Blau and Duncan's recursive basic stratification model.

```
Residual standard error: 0.7672 on 17 degrees of freedom
```

It was necessary to use the identity function `I` to "protect" the expression `wp + wg` in the first structural equation; as in a linear model, leaving an expression like this unprotected would cause the plus sign to be interpreted as specifying separate terms for the model, rather than as the sum of `wp` and `wg`, which is what is desired here.

2.4 Recursive Models

Outside of economics, it is common to specify a structural equation model in the form of a graph called a *path diagram.* A well known example, Blau and Duncan's basic stratification model (Blau and Duncan, 1967), appears in Figure 1.

The following conventions, some of them familiar from Klein's macroeconomic model, are employed in drawing the path diagram:

- Directly observable variables are enclosed in rectangular boxes.
- Unobservable variables are enclosed in circles (more generally, in ellipses); in this model, the only unobservable variables are the disturbances.
- Exogenous variables are represented by xs; endogenous variables by ys; and disturbances by ζs.
- Directed (i.e., single-headed) arrows represent structural parameters. The endogenous variables are distinguished from the exogenous variables by having directed arrows pointing towards them, while exogenous variables appear only at the tails of directed arrows.
- Bidirectional (double-headed) arrows represent non-causal, potentially nonzero, covariances between exogenous variables (and, more generally, also between disturbances).

- As before, γs are used for structural parameters relating an endogenous to an exogenous variable, while βs are used for structural parameters relating one endogenous variable to another.
- To the extent possible, horizontal ordering of the variables corresponds to their causal ordering: Thus, "causes" appear to the left of "effects."

The structural equations of the model may be read off the path diagram:[10]

$$\begin{aligned}
y_{1i} &= \gamma_{10} + \gamma_{11}x_{1i} + \gamma_{12}x_{2i} + \zeta_{1i} \\
y_{2i} &= \gamma_{20} + \gamma_{21}x_{1i} + \gamma_{22}x_{2i} + \beta_{21}y_{1i} + \zeta_{2i} \\
y_{3i} &= \gamma_{30} + \gamma_{32}x_{2i} + \beta_{31}y_{1i} + \beta_{32}y_{2i} + \zeta_{2i}
\end{aligned}$$

Blau and Duncan's model is a member of a special class of SEMs called *recursive models*. Recursive models have the following two defining characteristics:

1. There are no reciprocal directed paths or feedback loops in the path diagram.
2. Different disturbances are independent of one-another (and hence are unlinked by bidirectional arrows).

As a consequence of these two properties, the predictors in a structural equation of a recursive model are always independent of the error of that equation, and the structural equation may be estimated by OLS regression. Estimating a recursive model is simply a sequence of OLS regressions. In R, we would of course use `lm` to fit the regressions. This is a familiar operation, and therefore I will not pursue the example further, although the `sem` function, described below, can also fit these models.

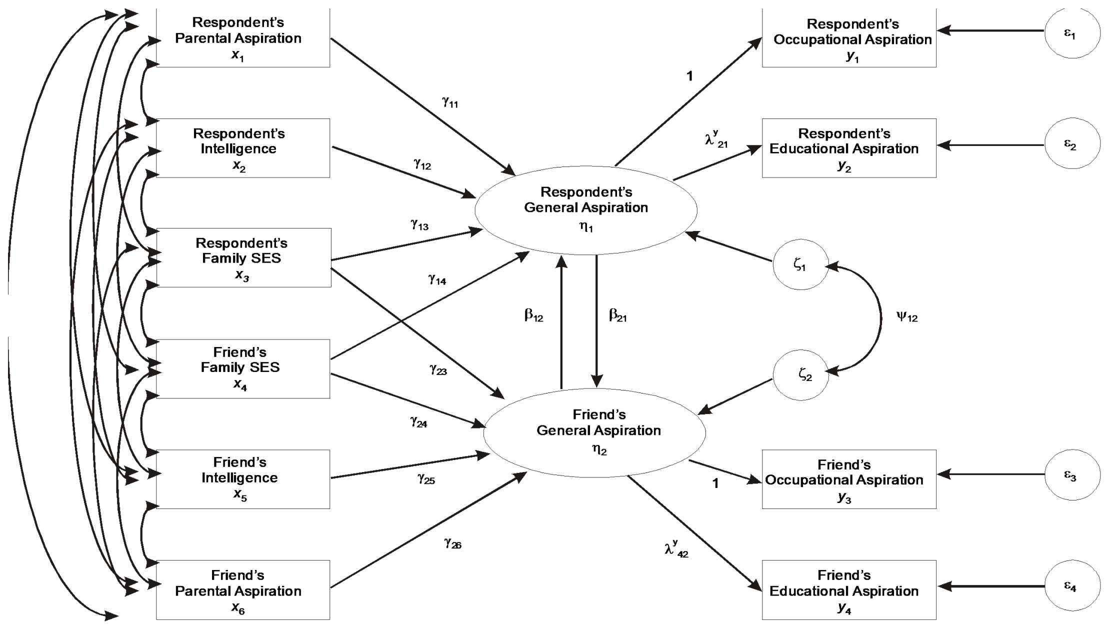
Structural equation models that are not recursive are sometimes termed *nonrecursive* (an awkward and often-confused adjective).

3 General Structural Equation Models

General structural equation models include unobservable exogenous or endogenous variables (also termed *factors* or *latent variables*) in addition to the unobservable disturbances. General structural equation models are sometimes called *LISREL models*, after the first widely available computer program capable of estimating this class of models (Jöreskog, 1973); LISREL is an acronym for *li*near *s*tructural *rel*ations.

Figure 2 shows the path diagram for an illustrative general structural equation model, from path-breaking work by Duncan et al. (1968) concerning peer influences on the aspirations of male high-school students. The most striking new feature of this model is that two of the endogenous variables, Respondent's General Aspirations (η_1) and Friend's General Aspirations (η_2), are unobserved variables. Each of these variables has two observed indicators: The occupational and educational aspirations of each boy — y_1 and y_2 for the respondent, and y_3 and y_4 for his best friend.

[10]In writing out the structural equations from a path diagram, it is common to omit the intercept parameters (here, γ_{10}, γ_{20}, and γ_{30}), for which no paths appear. To justify this practice, we may express all variables as deviations from their expectations (in the sample, as deviations from their means), eliminating the intercept from each regression equation.



Respondent's Parental Aspiration x_1
Respondent's Intelligence x_2
Respondent's Family SES x_3
Friend's Family SES x_4
Friend's Intelligence x_5
Friend's Parental Aspiration x_6
γ_{11}
γ_{12}
γ_{13}
γ_{14}
γ_{23}
γ_{24}
γ_{25}
γ_{26}
Respondent's General Aspiration η_1
Friend's General Aspiration η_2
β_{12}
β_{21}
ζ_1
ζ_2
ψ_{12}
1
λ^y_{21}
1
λ^y_{42}
Respondent's Occupational Aspiration y_1
Respondent's Educational Aspiration y_2
Friend's Occupational Aspiration y_3
Friend's Educational Aspiration y_4
ε_1
ε_2
ε_3
ε_4

Figure 2: Duncan, Haller, and Portes's general structural equation model for peer influences on aspirations.

3.1 The LISREL Model

It is common in general structural equation models such as the peer-influences model to distinguish between two sub-models:

1. A structural submodel, relating endogenous to exogenous variables and to one-another. In the peer-influences model, the endogenous variables are unobserved, while the exogenous variables are directly observed.

2. A measurement submodel, relating latent variables (here only latent endogenous variables) to their indicators.

I have used the following notation, associated with Jöreskog's LISREL model, in drawing the path diagram in Figure 2:

- xs are used to represent observable exogenous variables. If there were *latent* exogenous variables in the model, these would be represented by ξs (Greek *xi*), and xs would be used to represent their observable indicators.

- ys are employed to represent the indicators of the latent endogenous variables, which are symbolized by ηs (Greek *eta*). Were there directly observed endogenous variables in the model, then these too would be represented by ys.

- As before, γs and βs are used, respectively, for structural coefficients relating endogenous variables to exogenous variables and to one-another, and ζs are used for structural disturbances. The parameter ψ_{12} is the covariance between the disturbances ζ_1 and ζ_2 . The variances of the disturbances, ψ_1^2 and ψ_2^2, are not shown on the diagram.

- In the measurement submodel, λs (Greek *lambda*) represent regression coefficients (also called *factor loadings*) relating observable indicators to latent variables. The superscript y in λ^y indicates that the factor loadings in this model pertain to indicators of latent *endogenous* variables. One λ for each factor is set to 1; this is done to identify the scale of the corresponding latent variable.

- The εs (Greek *epsilon*) represent measurement error in the endogenous indicators; if there were *exogenous* indicators in the model, then the measurement errors associated with them would be represented by δs (Greek *delta*).

We are swimming in notation, but we still require some more (not all of which is necessary for the peer-influences model): We use σ_{ij} (Greek *sigma*) to represent the covariance between two observable variables; $\theta^{\varepsilon}_{ij}$ to represent the covariance between two measurement-error variables for endogenous indicators, ε_i and ε_j; θ^{δ}_{ij} to represent the covariance between two measurement-error variables for exogenous indicators, δ_i and δ_j; and ϕ_{ij} to represent the covariance between two latent exogenous variables ξ_i and ξ_j.

The LISREL notation for general structural equation models is summarized in Table 1. The structural and measurement submodels are written as follows:

$$\begin{aligned}
\boldsymbol{\eta}_i &= \mathbf{B}\boldsymbol{\eta}_i + \boldsymbol{\Gamma}\boldsymbol{\xi}_i + \boldsymbol{\varsigma}_i \\
\mathbf{y}_i &= \boldsymbol{\Lambda}_y\boldsymbol{\eta}_i + \boldsymbol{\varepsilon}_i \\
\mathbf{x}_i &= \boldsymbol{\Lambda}_x\boldsymbol{\xi}_i + \boldsymbol{\delta}_i
\end{aligned}$$

In order to identify the model, many of the parameters in $\mathbf{B}, \boldsymbol{\Gamma}, \boldsymbol{\Lambda}_x, \boldsymbol{\Lambda}_y, \boldsymbol{\Phi}, \boldsymbol{\Psi}, \boldsymbol{\Theta}_{\varepsilon}$, and $\boldsymbol{\Theta}_{\delta}$ must be constrained, typically by setting parameters to 0 or 1, or by defining certain parameters to be equal.

3.2 The RAM Formulation

Although LISREL notation is commonly used, there are several equivalent ways to represent general structural equation models. The `sem` function uses the simpler *RAM* (*reticular action model* – don't ask!) formulation of McArdle (1980) and McArdle and McDonald (1984); the notation that I employ below is from McDonald and Hartmann (1992).

The RAM model includes two vectors of variables: $\mathbf{v}$, which contains the indicator variables, directly observed exogenous variables, and the latent exogenous and endogenous variables in the model; and $\mathbf{u}$, which contains directly observed exogenous variables, measurement-error variables, and structural disturbances. The two sets of variables are related by the equation

$$\mathbf{v} = \mathbf{A}\mathbf{v} + \mathbf{u}$$

Thus, the matrix $\mathbf{A}$ includes structural coefficients and factor loadings. For example, for the Duncan, Haller, and Portes model, we have (using LISREL notation for the individual parameters):

Symbol	Meaning
N	Number of observations
m	Number of latent endogenous variables
n	Number of latent exogenous variables
p	Number of indicators of latent endogenous variables
q	Number of indicators of latent exogenous variable
$\underset{(m\times 1)}{\boldsymbol{\eta}_i}$	Latent endogenous variables (for observation i)
$\underset{(n\times 1)}{\boldsymbol{\xi}_i}$	Latent exogenous variables
$\underset{(m\times 1)}{\boldsymbol{\varsigma}_i}$	Structural disturbances (errors in equations)
$\underset{(m\times m)}{\mathbf{B}}$	Structural parameters relating latent endogenous variables
$\underset{(m\times n)}{\boldsymbol{\Gamma}}$	Structural parameters relating latent endogenous to exogenous variables
$\underset{(p\times 1)}{\mathbf{y}_i}$	Indicators of latent endogenous variables
$\underset{(q\times 1)}{\mathbf{x}_i}$	Indicators of latent exogenous variables
$\underset{(p\times 1)}{\boldsymbol{\varepsilon}_i}$	Measurement errors in endogenous indicators
$\underset{(q\times 1)}{\boldsymbol{\delta}_i}$	Measurement errors in exogenous indicators
$\left.\begin{matrix}\underset{(p\times m)}{\boldsymbol{\Lambda}_y} \\ \underset{(q\times n)}{\boldsymbol{\Lambda}_x}\end{matrix}\right\}$	Factor loadings relating indicators to latent variables
$\underset{(n\times n)}{\Phi}$	Covariances among latent exogenous variables
$\underset{(m\times m)}{\Psi}$	Covariances among structural disturbances
$\left.\begin{matrix}\underset{(p\times p)}{\boldsymbol{\Theta}_\varepsilon} \\ \underset{(q\times q)}{\boldsymbol{\Theta}_\delta}\end{matrix}\right\}$	Covariances among measurement errors
$\underset{(p+q\times p+q)}{\boldsymbol{\Sigma}}$	Covariances among observed (indicator) variables

Table 1: Notation for the LISREL model. The order of each vector or matrix is shown in paretheses below its symbol.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda^y_{21} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \lambda^y_{42} \\ \gamma_{11} & \gamma_{12} & \gamma_{13} & \gamma_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{12} \\ 0 & 0 & \gamma_{23} & \gamma_{24} & \gamma_{25} & \gamma_{26} & 0 & 0 & 0 & 0 & \beta_{21} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ \eta_1 \\ \eta_2 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \zeta_1 \\ \zeta_2 \end{bmatrix}$$

It is typically the case that **A** is sparse, containing many 0s. Notice the special treatment of the observed exogenous variables, x_1 through x_6, which are specified to be measured without error, and which consequently appear both in **v** and **u**.

The final component of the RAM formulation is the covariance matrix **P** of **u**.[11] Assuming that all of the error variables have expectations of 0, and that all other variables have been expressed as deviations from their expectations, $\mathbf{P} = E(\mathbf{uu}')$. For the illustrative model,

$$\mathbf{P} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} & \sigma_{15} & \sigma_{16} & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} & \sigma_{25} & \sigma_{26} & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} & \sigma_{35} & \sigma_{36} & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} & \sigma_{45} & \sigma_{46} & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{51} & \sigma_{52} & \sigma_{53} & \sigma_{54} & \sigma_{55} & \sigma_{56} & 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{61} & \sigma_{62} & \sigma_{63} & \sigma_{64} & \sigma_{65} & \sigma_{66} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \theta^\varepsilon_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta^\varepsilon_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta^\varepsilon_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \theta^\varepsilon_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \psi_{11} & \psi_{12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \psi_{21} & \psi_{22} \end{bmatrix}$$

For convenience, I use a double-subscript notation for both covariances and variances; thus, for example, σ_{11} is the variance of x_1 (usually written σ_1^2); θ^ε_{11} is the variance of ε_1; and ψ_{11} is the variance of ζ_1.

The key to estimating the model is the connection between the covariances of the observed variables, which may be estimated directly from sample data, and the parameters in **A** and **P**. Let m denote the the number of variables in **v**, and (without loss of generality) let the first n of these be the observed variables in the model.[12] Define the $m \times m$ *selection matrix* **J** to pick out the observed variables; that is

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

[11]More generally, **P** is a population moment matrix; for example, in a model that includes intercepts, **P** is a raw-moment matrix of expected "uncorrected" squares and cross-products.

[12]Notice the nonstandard use of n to represent the number of observed variables rather than the sample size. The latter is represented by N, as in the LISREL model.

where $\mathbf{I}_n$ is the order-n identity matrix, and the $\mathbf{0}$s are zero matrices of appropriate orders. The model implies the following covariances among the observed variables:

$$\mathbf{C} = E(\mathbf{J}\mathbf{v}\mathbf{v}'\mathbf{J}') = \mathbf{J}(\mathbf{I}_m - \mathbf{A})^{-1}\mathbf{P}(\mathbf{I}_m - \mathbf{A})^{-1'}\mathbf{J}'$$

Let $\mathbf{S}$ denote the observed-variable covariances computed directly from the sample. Fitting the model to the data — that is, estimating the free parameters in $\mathbf{A}$ and $\mathbf{P}$ — entails selecting parameter values that make $\mathbf{S}$ as close as possible to the model-implied covariances $\mathbf{C}$. Under the assumptions that the errors and latent variables are multivariately normally distributed, finding the maximum-likelihood estimates of the free parameters in $\mathbf{A}$ and $\mathbf{P}$ is equivalent to minimizing the criterion (see, e.g., Bollen, 1989, App. 4A and 4B) [13]

$$F(\mathbf{A}, \mathbf{P}) = \text{trace}(\mathbf{S}\mathbf{C}^{-1}) - n + \log_e \det \mathbf{C} - \log_e \det \mathbf{S} \tag{3}$$

3.3 The `sem` Function

By default, the `sem` function computes maximum-likelihood estimates for general structural equation models, using the RAM formulation of the model. There are several required arguments to `sem`:

1. `model`: a symbolic specification, in either character or numeric form, of the single- and double-headed arrows that define the model, along with free and fixed parameters, and possibly starting values for (some of) the free parameters. Normally, the `model` argument is not given directly by the user, but rather is constructed by one of the model-specification functions `specifyModel`, `specifyEquations`, or `cfa` (the latter for confirmatory factor analysis models), possibly in combination with `multigroupModel` to define a multiple-group model (see Sec. 3.7). The use of these functions for model specification is illustrated in the examples given below. Moreover, if a start value isn't given for a parameter — which is, indeed, the usual practice — then a start value will be computed using an adaptation of the method described by McDonald and Hartmann (1992). This method isn't entirely reliable, sometimes producing convergence problems, but it usually works reasonably well.

   If there are fixed exogenous variables in the model (such as variables x_1 through x_6 in the peer-influences model), then the variances and covariances of these variables do not have to be specified explicitly in the `model` argument to `sem`. Rather, the names of the fixed exogenous variables can be supplied via the argument `fixed.x`, as I will illustrate presently.

2. `S`: the sample covariance matrix (or other form of moment matrix) among the observed variables in the model. The covariances may be obtained from a secondary source or computed by the standard R function `var`. If `S` has row and column names, then these are used by default as the names of the observed variables. The `sem` function accepts a lower or upper-triangular covariance matrix, as well as the full (symmetric) covariance matrix. For a multigroup model, `S` is a named list of group covariance (or moment) matrices.

   Models with intercepts and mean structures can be fit by using a raw-moment matrix for `S` in place of the covariance matrix. The `rawMoments` function computes raw-moment matrices from data, and the `readMoments` function facilitates the direct entry of covariance and correlation matrices. Both `rawMoments` and `readMoments` are part of the **sem** package.

[13] Although multinormal maximum-likelihood is the most common criterion for fitting general structural equation models, there are other estimation criteria. The **sem** package, for example, is also capable of fitting a generalized least squares (GLS) estimator.

3. N: the sample size on which the covariance matrix **S** is based. In a multigroup model, N is a named vector of group sample sizes.

4. data and formula: Alternatively the data and formula arguments to sem can be used in place of S and N to provide the data to which the model is to be fit. In this case, data is a data frame and formula is a one-sided formula that is applied to data (and which defaults to ~.) to produce a numeric input data matrix. In a multigroup model, data may either be a named list of data frames, one for each group, or a single data frame with data for all of the groups. In the latter event, the group argument must give the name of the factor in the data set that defines the groups. Also in a multigroup model, there may be a named list of formulas for the separate groups, or a single common formula. If the original data are available, it is generally preferable to provide the data argument; for example, doing so makes possible the computation of robust coefficient standard errors.

Enter help(sem) for a description of the various optional arguments to sem (and see Section 3.8).

The Duncan, Haller and Portes model was estimated for standardized variables, so the input covariance matrix is a correlation matrix:[14]

```
> R.dhp <- readMoments(diag=FALSE, names=c("ROccAsp", "REdAsp", "FOccAsp",
+                 "FEdAsp", "RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp"))
1:     .6247
2:     .3269  .3669
4:     .4216  .3275  .6404
7:     .2137  .2742  .1124  .0839
11:     .4105  .4043  .2903  .2598  .1839
16:     .3240  .4047  .3054  .2786  .0489  .2220
22:     .2930  .2407  .4105  .3607  .0186  .1861  .2707
29:     .2995  .2863  .5191  .5007  .0782  .3355  .2302  .2950
37:     .0760  .0702  .2784  .1988  .1147  .1021  .0931 -.0438  .2087
46:

Read 45 items

> R.dhp

        ROccAsp REdAsp FOccAsp FEdAsp RParAsp    RIQ   RSES    FSES    FIQ
ROccAsp  1.0000 0.0000  0.0000 0.0000  0.0000 0.0000 0.0000  0.0000 0.0000
REdAsp   0.6247 1.0000  0.0000 0.0000  0.0000 0.0000 0.0000  0.0000 0.0000
FOccAsp  0.3269 0.3669  1.0000 0.0000  0.0000 0.0000 0.0000  0.0000 0.0000
FEdAsp   0.4216 0.3275  0.6404 1.0000  0.0000 0.0000 0.0000  0.0000 0.0000
RParAsp  0.2137 0.2742  0.1124 0.0839  1.0000 0.0000 0.0000  0.0000 0.0000
RIQ      0.4105 0.4043  0.2903 0.2598  0.1839 1.0000 0.0000  0.0000 0.0000
RSES     0.3240 0.4047  0.3054 0.2786  0.0489 0.2220 1.0000  0.0000 0.0000
FSES     0.2930 0.2407  0.4105 0.3607  0.0186 0.1861 0.2707  1.0000 0.0000
FIQ      0.2995 0.2863  0.5191 0.5007  0.0782 0.3355 0.2302  0.2950 1.0000
```

[14]Using correlation-matrix input raises a complication: The standard deviations employed to standardize variables are estimated from the data, and are therefore an additional source of uncertainty in the estimates of the standardized coefficients. I will simply bypass this issue, however, which is tantamount to analyzing the data on scales conditional on the sample standard deviations.

```
FParAsp  0.0760 0.0702  0.2784 0.1988  0.1147 0.1021 0.0931 -0.0438 0.2087
        FParAsp
ROccAsp       0
REdAsp        0
FOccAsp       0
FEdAsp        0
RParAsp       0
RIQ           0
RSES          0
FSES          0
FIQ           0
FParAsp       1
```

The model specification may be read off the path diagram (Figure 2), remembering that the error variables do not appear explicitly, and that we do not have to define explicit variance and covariance parameters for the six fixed exogenous variables:

```
> model.dhp <- specifyModel()
1: RParAsp  -> RGenAsp, gam11
2: RIQ      -> RGenAsp, gam12
3: RSES     -> RGenAsp, gam13
4: FSES     -> RGenAsp, gam14
5: RSES     -> FGenAsp, gam23
6: FSES     -> FGenAsp, gam24
7: FIQ      -> FGenAsp, gam25
8: FParAsp  -> FGenAsp, gam26
9: FGenAsp  -> RGenAsp, beta12
10: RGenAsp  -> FGenAsp, beta21
11: RGenAsp  -> ROccAsp,  NA,     1
12: RGenAsp  -> REdAsp,  lam21
13: FGenAsp  -> FOccAsp,  NA,     1
14: FGenAsp  -> FEdAsp,  lam42
15: RGenAsp <-> FGenAsp, ps12
16:

Read 15 records
NOTE: adding 6 variances to the model

> model.dhp

   Path                Parameter  StartValue
1  RParAsp  -> RGenAsp gam11
2  RIQ      -> RGenAsp gam12
3  RSES     -> RGenAsp gam13
4  FSES     -> RGenAsp gam14
5  RSES     -> FGenAsp gam23
6  FSES     -> FGenAsp gam24
7  FIQ      -> FGenAsp gam25
8  FParAsp  -> FGenAsp gam26
```

```
9  FGenAsp  -> RGenAsp beta12
10 RGenAsp  -> FGenAsp beta21
11 RGenAsp  -> ROccAsp <fixed>    1
12 RGenAsp  -> REdAsp  lam21
13 FGenAsp  -> FOccAsp <fixed>    1
14 FGenAsp  -> FEdAsp  lam42
15 RGenAsp <-> FGenAsp ps12
16 RGenAsp <-> RGenAsp V[RGenAsp]
17 FGenAsp <-> FGenAsp V[FGenAsp]
18 ROccAsp <-> ROccAsp V[ROccAsp]
19 REdAsp <-> REdAsp   V[REdAsp]
20 FOccAsp <-> FOccAsp V[FOccAsp]
21 FEdAsp <-> FEdAsp   V[FEdAsp]
```

By default, `specifyModel` reads the paths in the model from the input stream, although these could optionally be provided in a file. The numeric prompts (`1:`, `2:`, etc.) are provided by the function. Each path is given by a single-headed arrow, indicating a structural paramter, or a double-headed arrow, indicating a variance or covariance. Double-headed arrows linking endogenous variables represent error variances or covariances in the RAM formulation of the model. When an arrow is associated with a name, then the name (e.g., `gam11` for `RParAsp -> RGenAsp`) represents a free parameter to be estimated from the data. If two or more parameters are given the same name, then the corresponding parameters are constrained to be equal. If no parameter name is given (or if the name is `NA`), then the value of the parameter is fixed, and the fixed value must be specified. For example, the path `RGenAsp -> ROccAsp` is fixed to `1`. Values may also be specified for free parameters, in which case they are used as starting values in the iterative estimation process.

Also by default, `specifyModel` adds error variances for endogenous variables if these aren't given directly: see the documentation for the argument `endog.variances` in `?specifyModel` and also the arguments `exog.variances` and `covs`.

To fit the model, I note that the Duncan, Haller, and Portes data set comprises $N = 329$ observations, and that six of the variables in the model are fixed exogenous variables:

```
> sem.dhp <- sem(model.dhp, R.dhp, N=329,
+     fixed.x=c("RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp"))
> sem.dhp

 Model Chisquare =  26.7   Df =  15

     gam11      gam12      gam13      gam14      gam23      gam24      gam25
   0.16122    0.24965    0.21840    0.07184    0.06189    0.22887    0.34904
     gam26     beta12     beta21      lam21      lam42       ps12 V[RGenAsp]
   0.15953    0.18423    0.23548    1.06268    0.92973   -0.02261    0.28099
V[FGenAsp] V[ROccAsp]  V[REdAsp] V[FOccAsp]  V[FEdAsp]
   0.26384    0.41215    0.33615    0.31119    0.40460

 Iterations =  32
```

Specifying `fixed.x = c("RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp")` makes it unnecessary to specify all of the variances and covariances among these variables as free parameters.

The `sem` function returns an object of class `c("objectiveML", "sem")`, because by default the model was fit by multinormal maximum likelihood; the `print` method for `"objectiveML"` objects displays parameter estimates, together with the likelihood-ratio chi-square statistic for the model, contrasting the model with a just-identified (or *saturated*) model, which perfectly reproduces the sample covariance matrix. The degrees of freedom for this test are equal to the degree of over-identification of the model — the difference between the number of covariances among observed variables, $n(n+1)/2$, and the number of independent parameters in the model.[15]

As is typical, more information is provided by the `summary` method for `"objectiveML"` objects:

```
> summary(sem.dhp)

 Model Chisquare =  26.7   Df =  15 Pr(>Chisq) = 0.0313
 Goodness-of-fit index =  0.9844
 Adjusted goodness-of-fit index =  0.9428
 RMSEA index =  0.04876   90% CI: (0.01452, 0.07831)
 Bentler-Bonnett NFI =  0.9694
 Tucker-Lewis NNFI =  0.9576
 Bentler CFI =  0.9859
 SRMR =  0.0202
 AIC =  64.7
 AICc =  29.16
 BIC =  136.8
 CAIC =  -75.24

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8000 -0.1180  0.0000 -0.0120  0.0397  1.5700

 R-square for Endogenous Variables
RGenAsp FGenAsp ROccAsp  REdAsp FOccAsp  FEdAsp
 0.5220  0.6170  0.5879  0.6639  0.6888  0.5954

 Parameter Estimates
          Estimate Std Error z value Pr(>|z|)
gam11      0.16122 0.03879    4.1560 3.238e-05 RGenAsp <--- RParAsp
gam12      0.24965 0.04398    5.6763 1.376e-08 RGenAsp <--- RIQ
gam13      0.21840 0.04420    4.9415 7.751e-07 RGenAsp <--- RSES
gam14      0.07184 0.04971    1.4453 1.484e-01 RGenAsp <--- FSES
gam23      0.06189 0.05172    1.1966 2.315e-01 FGenAsp <--- RSES
gam24      0.22887 0.04416    5.1824 2.190e-07 FGenAsp <--- FSES
gam25      0.34904 0.04529    7.7067 1.291e-14 FGenAsp <--- FIQ
gam26      0.15953 0.03883    4.1089 3.975e-05 FGenAsp <--- FParAsp
beta12     0.18423 0.09489    1.9416 5.219e-02 RGenAsp <--- FGenAsp
beta21     0.23548 0.11939    1.9724 4.857e-02 FGenAsp <--- RGenAsp
```

[15]For the model to be identified the degrees of freedom must be 0 or greater — that is, there must be at least as many observable covariances as free parameters of the model. Unlike the order condition for the identification of observed-variable SEMs, however, it is common for this requirement to be met and yet for the model to be under-identified.

```
lam21        1.06268 0.09014   11.7894 4.429e-32 REdAsp <--- RGenAsp
lam42        0.92973 0.07028   13.2287 5.993e-40 FEdAsp <--- FGenAsp
ps12        -0.02261 0.05119   -0.4416 6.587e-01 FGenAsp <--> RGenAsp
V[RGenAsp]   0.28099 0.04623    6.0778 1.218e-09 RGenAsp <--> RGenAsp
V[FGenAsp]   0.26384 0.04467    5.9067 3.490e-09 FGenAsp <--> FGenAsp
V[ROccAsp]   0.41215 0.05122    8.0458 8.565e-16 ROccAsp <--> ROccAsp
V[REdAsp]    0.33615 0.05210    6.4519 1.104e-10 REdAsp <--> REdAsp
V[FOccAsp]   0.31119 0.04593    6.7758 1.237e-11 FOccAsp <--> FOccAsp
V[FEdAsp]    0.40460 0.04618    8.7606 1.942e-18 FEdAsp <--> FEdAsp

 Iterations =  32
```

- The marginally significant chi-square statistic indicates that the model can be rejected. Because virtually any over-identified model can be rejected in a sufficiently large sample, structural-equation modelers typically attend to the descriptive adequacy of the model as well as to this formal *over-identification test*.

- The *goodness-of-fit index* (*GFI*) and the *adjusted goodness-of-fit index* (*AGFI*) are ad-hoc measures of the descriptive adequacy of the model, included in the output of the `summary` method for `objectiveML` objects because they are in common use. The GFI and AGFI are defined as follows:

$$\begin{aligned}\text{GFI} &= 1 - \frac{\text{trace}\{[\mathbf{C}^{-1}(\mathbf{S}-\mathbf{C})]^2\}}{\text{trace}[(\mathbf{C}^{-1}\mathbf{S})^2]}\\ \text{AGFI} &= 1 - \frac{n(n+1)}{2\times\text{df}}(1-\text{GFI})\end{aligned}$$

  where df is the degrees of freedom for the model. Although the GFI and AGFI are thought of as proportions, comparing the value of the fitting criterion for the model with the value of the fitting criterion when no model is fit to the data, these indices are not constrained to the interval 0 to 1. Several rough cutoffs for the GFI and AGFI have been proposed; a general theme is that they should be close to 1. It is probably fair to say that the GFI and AGFI are of little pratical value.

- There is a veritable cottage industry in ad-hoc fit indices and their evaluation, and the model summary provides a number of these indices. See, for example, the papers in the volume edited by Bollen and Long (1993). One index that is perhaps more attractive than the others is the RMSEA (*root mean-squared error approximation*), which is an estimate of fit of the model relative to a saturated model in the population, and is computed as

$$\text{RMSEA} = \sqrt{\max\left(\frac{F}{\text{df}} - \frac{1}{N-1}, 0\right)}$$

  Here, F is the minimized fitting criterion, from equation (3). Small values of the RMSEA indicate that the model fits nearly as well as a saturated model; RMSEA ≤ 0.05 is generally taken as a good fit to the data. It is possible, moreover, to compute a confidence interval for the RMSEA. The RMSEA for the peer-influences model is a bit smaller than 0.05.

- In contrast with ad-hoc fit indices, the *Bayesian information criterion* (*BIC*) has a sound statistical basis (see Raftery, 1993). The BIC adjusts the likelihood-ratio chi-square statistic

L^2 for the number of parameters in the model, the number of observed variables, and the sample size:

$$\text{BIC} = L^2 - \text{df} \times \log_e nN$$

Negative values of BIC indicate a model that has greater support from the data than the just-identified model, for which BIC is 0. *Differences* in BIC may be used to compare alternative over-identified models; indeed, the BIC is used in a variety of contexts for model selection, not just in structural-equation modeling. Raftery suggests that a BIC difference of 5 is indicative of "strong evidence" that one model is superior to another, while a difference of 10 is indicative of "conclusive evidence." The AIC, AICc, and CAIC are alternative information criteria for model selection.

- The **sem** package provides several methods for calculating *residual covariances*, which compare the observed and model-implied covariance matrices, $\mathbf{S}$ and $\mathbf{C}$: Enter `?residuals.sem` for details. The `summary` method for `objectiveML` objects prints summary statistics for the distribution of the *normalized residual covariances*, which are defined as

$$\frac{s_{ij} - c_{ij}}{\sqrt{\dfrac{c_{ii}c_{jj} + c_{ij}^2}{N}}}$$

  Squared multiple correlations, R^2s, for the observed and latent endogenous variables in the model are also reported.

All of the structural coefficients in the peer-influences model are statistically significant, except for the coefficients linking each boy's general aspiration to the other boy's family socioeconomic status (SES).[16]

To illustrate setting parameter-equality constraints, I take advantage of the symmetry of the model to specify that all coefficients and error variances in the top half of the path diagram (Figure 2) are the same as the corresponding parameters in the lower half.[17] These constraints are plausible in light of the parameter estimates in the initial model, because corresponding estimates have similar values. The equality constraints are imposed as follows, using `specifyEquations` as an alternative to `specifyModel`:

```
> model.dhp.2 <- specifyEquations()
1: REdAsp = lamy*RGenAsp
2: ROccAsp = 1*RGenAsp
3: FEdAsp = lamy*FGenAsp
4: FOccAsp = 1*FGenAsp
5: RGenAsp = gam1*RParAsp + gam2*RIQ + gam3*RSES + gam4*FSES + beta*FGenAsp
6: FGenAsp = gam1*FParAsp + gam2*FIQ + gam3*FSES + gam4*RSES + beta*RGenAsp
7: V(RGenAsp) = psi
8: V(FGenAsp) = psi
9: C(RGenAsp, FGenAsp) = psi12
10: V(ROccAsp) = theps1
```

[16]The path from friend's to respondent's general aspiration is statistically significant by a one-sided test, which is appropriate here because the coefficient was expected to be positive.

[17]Although this specification makes some sense, the data are not entirely symmetric: Boys nominated their best friends, but this selection was not necessarily reciprocated.

```
11: V(REdAsp) = theps2
12: V(FOccAsp) = theps1
13: V(FEdAsp) = theps2
14:

Read 13 items
```

Using `specifyEquations` to define the model is usually simpler, and less error-prone, than using `specifyModel`. The equation-based syntax for `specifyEquations` is straightforward:

- One equation is provided for each endogenous variable in the model, which appears on the left-hand side of the equation. Each term on the right-hand side of the equation consists of a coefficient times (i.e., `*`) a variable.
- A parameter is given a fixed value by specifying a numeric constant for the coefficient — e.g., `1` for the coefficient of `RGenAsp` in line 2.
- Giving two or more parameters the same name (e.g., `lamy` in lines 1 and 3) imposes an equality constraint on the parameters.
- Variances and covariances are specified by `V()` and `C()` [or `v()` and `c()`]. Supplying a name for a variance or covariance makes it a free parameter; supplying a numeric constant (not illustrated in this example; e.g., `v(factor) = 1`) makes it a fixed parameter.
- Start values for free parameters (also not illustrated in this example) may be given in parentheses after the parameter name — e.g., `REdAsp = lamy(1)*RGenAsp`.
- Error variances for endogenous variables are given directly in this model in order to impose equality constraints. More generally, however, if error variances aren't given directly, they are automatically supplied by `specifyEquations` by default: see the documentation for the arguments `endog.variances`, `exog.variances`, and `covs` in `?specifyEquations`.

```
> sem.dhp.2 <- sem(model.dhp.2, R.dhp, N=329,
+     fixed.x=c("RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp"))
> summary(sem.dhp.2)

 Model Chisquare =  32.65   Df =  24 Pr(>Chisq) = 0.1117
 Goodness-of-fit index =  0.9805
 Adjusted goodness-of-fit index =  0.9552
 RMSEA index =  0.03314   90% CI: (NA, 0.05936)
 Bentler-Bonnett NFI =  0.9626
 Tucker-Lewis NNFI =  0.9804
 Bentler CFI =  0.9895
 SRMR =  0.02266
 AIC =  52.65
 AICc =  33.34
 BIC =  90.61
 CAIC =  -130.5

 Normalized Residuals
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8770 -0.2050  0.0000 -0.0167  0.1110  1.0400

 R-square for Endogenous Variables
RGenAsp  REdAsp ROccAsp FGenAsp  FEdAsp FOccAsp
 0.5671  0.6237  0.6380  0.5736  0.6272  0.6415

 Parameter Estimates
       Estimate Std Error z value Pr(>|z|)
lamy    0.98876 0.05569   17.7539 1.609e-70 REdAsp <--- RGenAsp
gam1    0.15709 0.02773    5.6643 1.476e-08 RGenAsp <--- RParAsp
gam2    0.30174 0.03288    9.1772 4.424e-20 RGenAsp <--- RIQ
gam3    0.22105 0.03184    6.9421 3.862e-12 RGenAsp <--- RSES
gam4    0.07280 0.03633    2.0042 4.505e-02 RGenAsp <--- FSES
beta    0.20496 0.07690    2.6653 7.693e-03 RGenAsp <--- FGenAsp
psi     0.27483 0.03309    8.3062 9.883e-17 RGenAsp <--> RGenAsp
psi12  -0.01408 0.05117   -0.2752 7.832e-01 FGenAsp <--> RGenAsp
theps1  0.36026 0.03434   10.4895 9.651e-26 ROccAsp <--> ROccAsp
theps2  0.37456 0.03431   10.9179 9.470e-28 REdAsp <--> REdAsp

 Iterations =  25
```

Because several pairs of parameters are constrained to be equal, this model has fewer free parameters, and correspondingly more degrees of freedom, than the original model. We can perform a likelihood-ratio test for the parameter constraints by taking differences in the model chi-square statistics and degrees of freedom:

$$\begin{aligned} L^2 &= 32.647 - 26.697 = 5.950 \\ df &= 24 - 15 = 9 \\ p &= .74 \end{aligned}$$

Equivalently,

```
> anova(sem.dhp, sem.dhp.2)

LR Test for Difference Between Models

          Model Df Model Chisq Df LR Chisq Pr(>Chisq)
sem.dhp         15         26.7
sem.dhp.2       24         32.6  9     5.95       0.74
```

Thus, the data appear to be consistent with the parameter constraints. Moreover, the more parsimonious constrained model has a much smaller BIC than the original model, and the constrained model has a non-significant over-identification test; the RMSEA has also improved.

So-called *modification indices* are score test statistics for fixed and constrained parameters in a structural equation model. If, for example, a parameter is incorrectly constrained to 0, then the test statistic for this parameter should be large. Of course, modifying a model based on the data raises issues of data-dredging, suggesting some form of model validation.

Applying `modIndices` to the respecified peer-influences model produces the following result:

```
> modIndices(sem.dhp.2)

 5 largest modification indices, A matrix:
 ROccAsp<-FEdAsp  FEdAsp<-ROccAsp FOccAsp<-ROccAsp    ROccAsp<-RSES
          5.195            5.093            3.982            3.786
RGenAsp<-FOccAsp
          2.894

  5 largest modification indices, P matrix:
 FEdAsp<->ROccAsp FOccAsp<->ROccAsp    RSES<->ROccAsp     RSES<->REdAsp
          11.930             9.774             4.356             4.200
RGenAsp<->FOccAsp
           4.077
```

The modIndices function returns an object of class "modIndices"; the print method for objects of this class reports the largest modification indices for parameters in the **A** and **P** matrices of the RAM model. These are chi-square statistics, each on one degree of freedom. There is a problem of simultaneous inference in examining the largest of many test statistics. Nevertheless, the modification indices can suggest improvements to an ill-fitting model. The summary method for "modIndices" objects prints the full matrices of modification indices, along with estimated changes in the parameter estimates upon freeing individual parameters.

Although the respecified peer-influences model fits quite well, I pursue the modification indices for the purpose of illustration. None of the modification indices for coefficients in **A** are very large, but there are a couple of moderately large modification indices for the covariances in **P**. Both of these involve measurement-error covariances between indicators of general aspirations for the respondent and for the best friend. Correlated measurement errors between friend's educational aspiration and respondent's occupational aspiration (the covariance with the largest modification index) do not seem substantively compelling, but correlated errors between the two indicators of occupational aspirations (corresponding to the second-largest modification index) make more sense.

Respecifying the model to accomodate the error correlation for the two educational-aspiration indicators yields a substantial decrease in the chi-square statistic for the model (a bit more than 10 — as is common, slightly larger than the modification index), a small decrease in the BIC, and an RMSEA of 0. I illustrate how to modify the previous model using the update function, which, as here, can often be simpler than specifying the model *de novo*:[18]

```
> model.dhp.3 <- update(model.dhp.2)
1: add, FOccAsp <-> ROccAsp, theps24
2:

 Read 1 record

> sem.dhp.3 <- sem(model.dhp.3, R.dhp, N=329,
+     fixed.x=c("RParAsp", "RIQ", "RSES", "FSES", "FIQ", "FParAsp"))
> summary(sem.dhp.3)

 Model Chisquare =  22.47   Df =  23 Pr(>Chisq) = 0.4923
 Goodness-of-fit index =  0.9864
```

[18]See ?update.semmod for details.

```
 Adjusted goodness-of-fit index =  0.9676
 RMSEA index =  0   90% CI: (NA, 0.04419)
 Bentler-Bonnett NFI =  0.9742
 Tucker-Lewis NNFI =  1.001
 Bentler CFI =  1
 SRMR =  0.02169
 AIC =  44.47
 AICc =  23.3
 BIC =  86.22
 CAIC =  -133.8

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.9910 -0.1160  0.0000 -0.0246  0.1980  0.6760

 R-square for Endogenous Variables
RGenAsp  REdAsp ROccAsp FGenAsp  FEdAsp FOccAsp
 0.5802  0.6066  0.6630  0.5864  0.6102  0.6664

 Parameter Estimates
        Estimate Std Error z value Pr(>|z|)
lamy     0.95409 0.05116   18.6509 1.241e-77 REdAsp <--- RGenAsp
gam1     0.16071 0.02812    5.7155 1.094e-08 RGenAsp <--- RParAsp
gam2     0.30723 0.03317    9.2621 2.005e-20 RGenAsp <--- RIQ
gam3     0.22607 0.03220    7.0219 2.188e-12 RGenAsp <--- RSES
gam4     0.07253 0.03693    1.9637 4.956e-02 RGenAsp <--- FSES
beta     0.20436 0.07682    2.6604 7.806e-03 RGenAsp <--- FGenAsp
psi      0.27850 0.03481    7.9996 1.248e-15 RGenAsp <--> RGenAsp
psi12    0.01449 0.05410    0.2679 7.888e-01 FGenAsp <--> RGenAsp
theps1   0.33714 0.03441    9.7982 1.146e-22 ROccAsp <--> ROccAsp
theps2   0.39157 0.03438   11.3901 4.682e-30 REdAsp <--> REdAsp
theps24 -0.09879 0.03133   -3.1532 1.615e-03 ROccAsp <--> FOccAsp

 Iterations =  28
```

3.3.1 Standardized Coefficients

Standardized coefficients for a fitted structural equation model can be obtained with the `standardizedCoefficients` command; for example:

```
> standardizedCoefficients(sem.dhp.3)

           Std. Estimate
1     lamy       0.77887  REdAsp <--- RGenAsp
2                0.81428 ROccAsp <--- RGenAsp
3     lamy       0.78115  FEdAsp <--- FGenAsp
4                0.81631 FOccAsp <--- FGenAsp
5     gam1       0.19731 RGenAsp <--- RParAsp
6     gam2       0.37721     RGenAsp <--- RIQ
```

```
7     gam3       0.27756    RGenAsp <--- RSES
8     gam4       0.08905    RGenAsp <--- FSES
9     beta       0.20589 RGenAsp <--- FGenAsp
10    gam1       0.19584 FGenAsp <--- FParAsp
11    gam2       0.37440     FGenAsp <--- FIQ
12    gam3       0.27550    FGenAsp <--- FSES
13    gam4       0.08838    FGenAsp <--- RSES
14    beta       0.20284 FGenAsp <--- RGenAsp
15     psi       0.41981 RGenAsp <--> RGenAsp
16     psi       0.41360 FGenAsp <--> FGenAsp
17   psi12       0.02168 FGenAsp <--> RGenAsp
18  theps1       0.33696 ROccAsp <--> ROccAsp
19  theps2       0.39336   REdAsp <--> REdAsp
20  theps1       0.33363 FOccAsp <--> FOccAsp
21  theps2       0.38980   FEdAsp <--> FEdAsp
22 theps24      -0.09824 ROccAsp <--> FOccAsp
23               1.00000 RParAsp <--> RParAsp
24               0.18390     RIQ <--> RParAsp
25               1.00000         RIQ <--> RIQ
26               0.04890    RSES <--> RParAsp
27               0.22200        RSES <--> RIQ
28               1.00000       RSES <--> RSES
29               0.01860    FSES <--> RParAsp
30               0.18610        FSES <--> RIQ
31               0.27070       FSES <--> RSES
32               1.00000       FSES <--> FSES
33               0.07820     FIQ <--> RParAsp
34               0.33550         FIQ <--> RIQ
35               0.23020        FIQ <--> RSES
36               0.29500        FIQ <--> FSES
37               1.00000         FIQ <--> FIQ
38               0.11470 FParAsp <--> RParAsp
39               0.10210     FParAsp <--> RIQ
40               0.09310    FParAsp <--> RSES
41              -0.04380    FParAsp <--> FSES
42               0.20870     FParAsp <--> FIQ
43               1.00000 FParAsp <--> FParAsp
```

The fixed as well as free parameters are standardized, as are the covariances among the fixed exogenous variables — here to no effect because the fixed exogenous variables were standardized in the input covariance (correlation) matrix.

3.4 Fitting Structural Equation Models to Data Sets

As mentioned, if the original data for a structural equation model are available, then it's advantageous to start with them rather than with a covariance or raw-moment matrix. I illustrate with an example from Bollen (1989, Ch. 8), using data on industrialization and democracy for 75 developing nations. The data set is included in the **sem** package in the `Bollen` data frame; unfortunately, the names of the countries aren't given with the data:

```
> head(Bollen)

     y1    y2     y3    y4    y5    y6     y7     y8    x1    x2    x3
1  2.50 0.000  3.333 0.000 1.250 0.000  3.726 3.3333 4.443 3.638 2.558
2  1.25 0.000  3.333 0.000 6.250 1.100  6.667 0.7370 5.384 5.063 3.568
3  7.50 8.800 10.000 9.200 8.750 8.094 10.000 8.2118 5.961 6.256 5.224
4  8.90 8.800 10.000 9.200 8.908 8.128 10.000 4.6151 6.286 7.568 6.267
5 10.00 3.333 10.000 6.667 7.500 3.333 10.000 6.6667 5.864 6.819 4.574
6  7.50 3.333  6.667 6.667 6.250 1.100  6.667 0.3685 5.533 5.136 3.892
```

The data comprise four measures of democracy at two points in time, 1960 and 1965, and three measures of industrialization in 1960. The variables are labelled as in Bollen (1989):

- y_1: freedom of the press, 1960
- y_2: freedom of political opposition, 1960
- y_3: fairness of elections, 1960
- y_4: effectivness of elected legislature, 1960
- y_5: freedom of the press, 1965
- y_6: freedom of political opposition, 1965
- y_7: fairness of elections, 1965
- y_8: effectivness of elected legislature, 1965
- x_1: GNP per capita, 1960
- x_2: energy consumption per capita, 1960
- x_3: percentage of labor force in industry, 1960

Letting η_1 represent the latent endogenous variable *political democracy in 1960*, η_2 the latent endogenous variable *political democracy in 1965*, and ξ_1 the latent exogeous variable *industrialization in 1960*, Bollen specified the following recursive structural model

$$\begin{aligned}\eta_1 &= \gamma_{11}\xi_1 + \zeta_1 \\ \eta_2 &= \beta_{21}\eta_1 + \gamma_{21}\xi_1 + \zeta_2\end{aligned}$$

and the measurement submodel

$$\begin{aligned}
y_1 &= \eta_1 + \varepsilon_1 \\
y_2 &= \lambda_2\eta_1 + \varepsilon_2 \\
y_3 &= \lambda_3\eta_1 + \varepsilon_3 \\
y_4 &= \lambda_4\eta_1 + \varepsilon_4 \\
y_5 &= \eta_2 + \varepsilon_5 \\
y_6 &= \lambda_2\eta_2 + \varepsilon_6 \\
y_7 &= \lambda_3\eta_2 + \varepsilon_7 \\
y_8 &= \lambda_4\eta_2 + \varepsilon_8 \\
x_1 &= \xi_1 + \delta_1 \\
x_2 &= \lambda_6\xi_1 + \delta_2 \\
x_3 &= \lambda_7\xi_1 + \delta_3
\end{aligned}$$

Notice the equality constraints in the λs ("factor loadings") for the endogenous indicators (the ys). Bollen also specified nonzero error covariances for some of the endogenous indicators: $\theta^{\varepsilon}_{15}$, $\theta^{\varepsilon}_{26}$, $\theta^{\varepsilon}_{37}$, $\theta^{\varepsilon}_{48}$, $\theta^{\varepsilon}_{24}$, and $\theta^{\varepsilon}_{68}$. Establishing the indentification status of a model like this is a nontrivial endeavor, but Bollen shows that the model is identified.

We can specify and estimate Bollen's model as follows:

```
> model.bollen <- specifyEquations()
1: y1 = 1*Demo60
2: y2 = lam2*Demo60
3: y3 = lam3*Demo60
4: y4 = lam4*Demo60
5: y5 = 1*Demo65
6: y6 = lam2*Demo65
7: y7 = lam3*Demo65
8: y8 = lam4*Demo65
9: x1 = 1*Indust
10: x2 = lam6*Indust
11: x3 = lam7*Indust
12: c(y1, y5) = theta15
13: c(y2, y4) = theta24
14: c(y2, y6) = theta26
15: c(y3, y7) = theta37
16: c(y4, y8) = theta48
17: c(y6, y8) = theta68
18: Demo60 = gamma11*Indust
19: Demo65 = gamma21*Indust + beta21*Demo60
20: v(Indust) = phi
21:

Read 20 items
NOTE: adding 13 variances to the model

> sem.bollen <- sem(model.bollen, data=Bollen)
> summary(sem.bollen)
```

```
 Model Chisquare =  39.64   Df =  38 Pr(>Chisq) = 0.3966
 Goodness-of-fit index =  0.9197
 Adjusted goodness-of-fit index =  0.8606
 RMSEA index =  0.02418   90% CI: (NA, 0.08619)
 Bentler-Bonnett NFI =  0.945
 Tucker-Lewis NNFI =  0.9964
 Bentler CFI =  0.9975
 SRMR =  0.05577
 AIC =  95.64
 AICc =  74.95
 BIC =  160.5
 CAIC =  -162.4

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-1.1400 -0.3780 -0.0211 -0.0399  0.2780  1.0500 

 R-square for Endogenous Variables
Demo60     y1     y2     y3     y4 Demo65     y5     y6     y7     y8     x1 
0.2004 0.7232 0.4755 0.5743 0.7017 0.9645 0.6673 0.5697 0.6425 0.6870 0.8464 
    x2     x3 
0.9465 0.7606 

 Parameter Estimates
          Estimate Std Error z value Pr(>|z|)                        
lam2      1.19078  0.14020    8.4934 2.007e-17 y2 <--- Demo60        
lam3      1.17454  0.12121    9.6899 3.328e-22 y3 <--- Demo60        
lam4      1.25098  0.11757   10.6400 1.941e-26 y4 <--- Demo60        
lam6      2.17966  0.13932   15.6453 3.577e-55 x2 <--- Indust        
lam7      1.81821  0.15290   11.8913 1.314e-32 x3 <--- Indust        
theta15   0.59042  0.36307    1.6262 1.039e-01 y5 <--> y1            
theta24   1.45958  0.70251    2.0777 3.774e-02 y4 <--> y2            
theta26   2.21251  0.75242    2.9405 3.277e-03 y6 <--> y2            
theta37   0.72120  0.62333    1.1570 2.473e-01 y7 <--> y3            
theta48   0.36771  0.45324    0.8113 4.172e-01 y8 <--> y4            
theta68   1.39033  0.58859    2.3621 1.817e-02 y8 <--> y6            
gamma11   1.47133  0.39496    3.7253 1.951e-04 Demo60 <--- Indust    
gamma21   0.60047  0.22722    2.6427 8.224e-03 Demo65 <--- Indust    
beta21    0.86504  0.07538   11.4765 1.732e-30 Demo65 <--- Demo60    
phi       0.45466  0.08846    5.1399 2.749e-07 Indust <--> Indust    
V[Demo60] 3.92769  0.88312    4.4475 8.686e-06 Demo60 <--> Demo60    
V[y1]     1.87971  0.44229    4.2500 2.138e-05 y1 <--> y1            
V[y2]     7.68379  1.39404    5.5119 3.550e-08 y2 <--> y2            
V[y3]     5.02264  0.97586    5.1469 2.648e-07 y3 <--> y3            
V[y4]     3.26806  0.73807    4.4278 9.519e-06 y4 <--> y4            
V[Demo65] 0.16669  0.23159    0.7198 4.717e-01 Demo65 <--> Demo65    
V[y5]     2.34430  0.48850    4.7989 1.595e-06 y5 <--> y5            
V[y6]     5.03533  0.93992    5.3572 8.454e-08 y6 <--> y6            
```

```
V[y7]     3.60814  0.72394    4.9840 6.228e-07 y7 <--> y7
V[y8]     3.35240  0.71788    4.6699 3.014e-06 y8 <--> y8
V[x1]     0.08249  0.01986    4.1538 3.271e-05 x1 <--> x1
V[x2]     0.12206  0.07105    1.7178 8.584e-02 x2 <--> x2
V[x3]     0.47297  0.09197    5.1427 2.709e-07 x3 <--> x3

 Iterations =  178
```

3.4.1 Robust Standard Errors

One of the advantages of fitting the model to the original data rather than to a moment matrix is the ability to compute robust standard errors and tests for the parameter estimates (see Satorra and Bentler, 1988; Bentler and Dudgeon, 1996). Robust standard errors and tests are obtained by specifying the argument `robust=TRUE` to the `summary` method for the `"objectiveML"` object produced by `sem`; for example, for the Bollen model:

```
> summary(sem.bollen, robust=TRUE)

Satorra-Bentler Corrected Fit Statistics:

 Corrected Model Chisquare =  43.06   Df =  38 Pr(>Chisq) = 0.2635
 Corrected Chisquare (null model) =  783.1   Df =  55
 Corrected Bentler-Bonnett NFI =  0.9494
 Corrected Tucker-Lewis NNFI =  0.9899
 Corrected Bentler CFI =  0.993

Uncorrected Fit Statistics:

 Model Chisquare =  39.64   Df =  38 Pr(>Chisq) = 0.3966
 Goodness-of-fit index =  0.9197
 Adjusted goodness-of-fit index =  0.8606
 RMSEA index =  0.02418   90% CI: (NA, 0.08619)
 Bentler-Bonnett NFI =  0.945
 Tucker-Lewis NNFI =  0.9964
 Bentler CFI =  0.9975
 SRMR =  0.05577
 AIC =  95.64
 AICc =  74.95
 BIC =  160.5
 CAIC =  -162.4

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.1400 -0.3780 -0.0211 -0.0399  0.2780  1.0500

 R-square for Endogenous Variables
Demo60     y1     y2     y3     y4 Demo65     y5     y6     y7     y8     x1
0.2004 0.7232 0.4755 0.5743 0.7017 0.9645 0.6673 0.5697 0.6425 0.6870 0.8464
    x2     x3
```

```
0.9465 0.7606

 Parameter Estimates (with Robust Standard Errors)
          Estimate Corrected SE z value Pr(>|z|)
lam2      1.19078  0.11285      10.5517 4.991e-26 y2 <--- Demo60
lam3      1.17454  0.12688       9.2570 2.102e-20 y3 <--- Demo60
lam4      1.25098  0.10340      12.0979 1.083e-33 y4 <--- Demo60
lam6      2.17966  0.12055      18.0814 4.463e-73 x2 <--- Indust
lam7      1.81821  0.12132      14.9872 8.896e-51 x3 <--- Indust
theta15   0.59042  0.35708       1.6535 9.824e-02 y5 <--> y1
theta24   1.45958  0.69505       2.1000 3.573e-02 y4 <--> y2
theta26   2.21251  0.78493       2.8187 4.821e-03 y6 <--> y2
theta37   0.72120  0.53296       1.3532 1.760e-01 y7 <--> y3
theta48   0.36771  0.39335       0.9348 3.499e-01 y8 <--> y4
theta68   1.39033  0.69859       1.9902 4.657e-02 y8 <--> y6
gamma11   1.47133  0.30074       4.8924 9.960e-07 Demo60 <--- Indust
gamma21   0.60047  0.19441       3.0887 2.010e-03 Demo65 <--- Indust
beta21    0.86504  0.07222      11.9777 4.652e-33 Demo65 <--- Demo60
phi       0.45466  0.07038       6.4600 1.047e-10 Indust <--> Indust
V[Demo60] 3.92769  0.80718       4.8659 1.139e-06 Demo60 <--> Demo60
V[y1]     1.87971  0.30957       6.0720 1.263e-09 y1 <--> y1
V[y2]     7.68379  1.27749       6.0147 1.802e-09 y2 <--> y2
V[y3]     5.02264  0.99313       5.0574 4.251e-07 y3 <--> y3
V[y4]     3.26806  0.75569       4.3246 1.528e-05 y4 <--> y4
V[Demo65] 0.16669  0.22212       0.7504 4.530e-01 Demo65 <--> Demo65
V[y5]     2.34430  0.49119       4.7727 1.817e-06 y5 <--> y5
V[y6]     5.03533  0.82857       6.0772 1.223e-09 y6 <--> y6
V[y7]     3.60814  0.51656       6.9849 2.850e-12 y7 <--> y7
V[y8]     3.35240  0.81249       4.1261 3.690e-05 y8 <--> y8
V[x1]     0.08249  0.01447       5.7010 1.191e-08 x1 <--> x1
V[x2]     0.12206  0.07464       1.6352 1.020e-01 x2 <--> x2
V[x3]     0.47297  0.07629       6.1996 5.660e-10 x3 <--> x3

 Iterations =  178
```

In this case, most of the normal-theory and robust standard errors are quite similar:

```
> round(sqrt(diag(vcov(sem.bollen)))/sqrt(diag(vcov(sem.bollen, robust=TRUE))), 2)
```

```
     lam2      lam3      lam4      lam6      lam7   theta15   theta24   theta26
     1.24      0.96      1.14      1.16      1.26      1.02      1.01      0.96
  theta37   theta48   theta68   gamma11   gamma21    beta21       phi V[Demo60]
     1.17      1.15      0.84      1.31      1.17      1.04      1.26      1.09
    V[y1]     V[y2]     V[y3]     V[y4] V[Demo65]     V[y5]     V[y6]     V[y7]
     1.43      1.09      0.98      0.98      1.04      0.99      1.13      1.40
    V[y8]     V[x1]     V[x2]     V[x3]
     0.88      1.37      0.95      1.21
```

3.5 Bootstrapping Structural Equation Models: A Model for Ordinal Data

The `CNES` data set in the **sem** package includes responses to four statements meant to tap respondents' attitudes towards "traditional values." The statements appeared in the mailback-questionnaire component of the 1997 Canadian National Election Study, and each provided the four response categories "strongly disagree," "disagree," "agree," and "strongly agree":

- `MBSA2`: "We should be more tolerant of people who choose to live according to their own standards, even if they are very different from our own."
- `MBSA7`: "Newer lifestyles are contributing to the breakdown of our society."
- `MBSA8`: "The world is always changing and we should adapt our view of moral behaviour to these changes."
- `MBSA9`: "This country would have many fewer problems if there were more emphasis on traditional family values."

These variables are ordered factors in the `CNES` data frame:

```
> head(CNES)

          MBSA2            MBSA7            MBSA8         MBSA9
1 StronglyAgree            Agree         Disagree      Disagree
2         Agree    StronglyAgree StronglyDisagree StronglyAgree
3         Agree         Disagree         Disagree         Agree
4 StronglyAgree            Agree StronglyDisagree StronglyAgree
5         Agree StronglyDisagree            Agree      Disagree
6         Agree         Disagree            Agree         Agree
```

I will entertain a one-factor confirmatory factor analysis (CFA) model for the `CNES` data:

$$\begin{aligned}
x_1 &= \lambda_1\xi + \delta_1\\
x_2 &= \lambda_2\xi + \delta_2\\
x_3 &= \lambda_3\xi + \delta_3\\
x_4 &= \lambda_4\xi + \delta_3\\
V(\xi) &= 1
\end{aligned}$$

The simplest way to specify a CFA model in the **sem** package is via the `cfa` function:

```
> model.cnes <- cfa()
1: F: MBSA2, MBSA7, MBSA8, MBSA9
2:

Read 1 item
NOTE: adding 4 variances to the model

> model.cnes
```

```
  Path            Parameter    StartValue
1 F -> MBSA2      lam[MBSA2:F]
2 F -> MBSA7      lam[MBSA7:F]
3 F -> MBSA8      lam[MBSA8:F]
4 F -> MBSA9      lam[MBSA9:F]
5 F <-> F         <fixed>      1
6 MBSA2 <-> MBSA2 V[MBSA2]
7 MBSA7 <-> MBSA7 V[MBSA7]
8 MBSA8 <-> MBSA8 V[MBSA8]
9 MBSA9 <-> MBSA9 V[MBSA9]
```

Each input directive to `cfa`, here a single line, contains the name of a factor (i.e., latent variable — `F` in the example), followed by a colon and the names of the observed variables that load on the factor, separated by commas; this variable list can, if necessary, extend over several input lines. Like `specifyEquations`, `cfa` translates the model into RAM format. By default, factor variances are fixed to 1, and, if there is more than one factor, their covariances (correlations) are specified as free parameters to be estimated from the data. Finally, error-variance parameters for the observed variables are automatically added to the model. See `?cfa` for all of the arguments to `cfa` and their defaults.

Fitting this model directly to the `CNES` data isn't appropriate because the variables in the data set are ordinal, not numeric. One approach to modeling ordinal data is to begin by computing polychoric correlations among the ordered factors, a procedure that assumes that each ordinal variable represents the dissection of a corresponding latent continuous variable into categories at unknown thresholds or cut-points. The polychoric correlations are each estimated, along with the thresholds, assuming that the corresponding pair of latent variables is bivariately normally distributed.

If there are both ordinal and numeric variables in a data set, which is not the case for the `CNES` data, then polychoric correlations can be computed between pairs of ordinal variables, polyserial correlations between ordinal and numeric variables, and Pearson product-moment correlations between numeric variables. The `hetcor` function in the **polycor** package (Fox, 2010) computes such "heterogeneous" correlation matrices. I write a small function, `hcor`, to extract the correlation matrix from the object returned by `hetcor`, which includes information in addition to the correlations themselves:

```
> library(polycor)
> hcor <- function(data) hetcor(data, std.err=FALSE)$correlations
> (R.cnes <- hcor(CNES))
```

```
        MBSA2   MBSA7   MBSA8   MBSA9
MBSA2  1.0000 -0.3018  0.2821 -0.2230
MBSA7 -0.3018  1.0000 -0.3422  0.5450
MBSA8  0.2821 -0.3422  1.0000 -0.3207
MBSA9 -0.2230  0.5450 -0.3207  1.0000
```

Next, I fit the CFA model to the data, using the polychoric correlations as input

```
> summary(sem.cnes <- sem(model.cnes, R.cnes, N=1529))
```

```
 Model Chisquare =  33.21   Df =  2 Pr(>Chisq) = 6.141e-08
 Goodness-of-fit index =  0.9893
 Adjusted goodness-of-fit index =  0.9467
 RMSEA index =  0.1011   90% CI: (0.07261, 0.1326)
 Bentler-Bonnett NFI =  0.9663
 Tucker-Lewis NNFI =  0.9043
 Bentler CFI =  0.9681
 SRMR =  0.03536
 AIC =  49.21
 AICc =  33.31
 BIC =  91.87
 CAIC =  16.55

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.030   0.208   0.848   1.040   3.830

 R-square for Endogenous Variables
 MBSA2  MBSA7  MBSA8  MBSA9
0.1516 0.6052 0.2197 0.4717

 Parameter Estimates
             Estimate Std Error z value Pr(>|z|)
lam[MBSA2:F] -0.3893  0.02875   -13.54   9.127e-42 MBSA2 <--- F
lam[MBSA7:F]  0.7779  0.02997    25.96  1.379e-148 MBSA7 <--- F
lam[MBSA8:F] -0.4687  0.02840   -16.50   3.478e-61 MBSA8 <--- F
lam[MBSA9:F]  0.6868  0.02922    23.51  3.346e-122 MBSA9 <--- F
V[MBSA2]      0.8484  0.03281    25.86  2.117e-147 MBSA2 <--> MBSA2
V[MBSA7]      0.3948  0.03568    11.07   1.798e-28 MBSA7 <--> MBSA7
V[MBSA8]      0.7803  0.03152    24.75  2.864e-135 MBSA8 <--> MBSA8
V[MBSA9]      0.5283  0.03213    16.44   9.207e-61 MBSA9 <--> MBSA9

 Iterations =  13
```

A problem, however, is that the standard errors and statistical tests reported by `sem` are not valid for polychoric-correlation input. One way to deal with this problem is to use the bootstrap for statistical inference. The **sem** package includes the function `bootSem`, which implements a simple case bootstrap for structural equation models fit by `sem`.[19] The `bootSem` function creates an object of class `c("bootsem", "boot")`. The **sem** package includes `print` and `summary` methods for `"bootsem"` objects, and these objects can also be processed by functions in the **boot** package (Davison and Hinkley, 1997; Canty and Ripley, 2012). Indeed, `bootSem` uses the `boot` function to perform the bootstrapping, and for more complex bootstrapping schemes, `boot` can be called directly.

The bootstrap computations are time-consuming, mostly because of the iterative calculation of polychoric correlations for each bootstrap sample, and so I perform only 100 bootstrap replications. With so few replications, I report normal-theory bootstrap confidence intervals based on the bootstrap standard errors, rather than more reliable percentile-based BCa intervals. In a real application, I would use a much larger number of bootstrap replications.

[19]See Fox and Weisberg (2011, Sec. 4.3.7) and Fox and Weisberg (2012) for a discussion of bootstrapping regression models in R.

```
> set.seed(12345) # for reproducibility
> system.time(boot.cnes <- bootSem(sem.cnes, R=100, cov=hcor, data=CNES))

   user  system elapsed
  36.19    0.27   36.51

> (boot.summary <- summary(boot.cnes, type="norm"))

Call: bootSem(model = sem.cnes, R = 100, cov = hcor, data = CNES)

Lower and upper limits are for the 95 percent norm confidence interval

             Estimate        Bias Std.Error   Lower   Upper
lam[MBSA2:F]  -0.3893  0.00569070   0.03434 -0.4623 -0.3277
lam[MBSA7:F]   0.7779 -0.00154918   0.03398  0.7129  0.8461
lam[MBSA8:F]  -0.4687  0.00005861   0.03170 -0.5309 -0.4066
lam[MBSA9:F]   0.6868  0.00366638   0.03053  0.6233  0.7430
V[MBSA2]       0.8484  0.00323127   0.02622  0.7938  0.8966
V[MBSA7]       0.3948  0.00126492   0.05278  0.2901  0.4970
V[MBSA8]       0.7803 -0.00094025   0.03034  0.7218  0.8407
V[MBSA9]       0.5283 -0.00597260   0.04275  0.4505  0.6181
```

For curiousity, I compare the coefficient standard errors computed by `sem` directly with the presumably more valid boostrap standard errors. For most of the coefficients, the two sets of standard errors are reasonably similar:

```
> round(sqrt(diag(vcov(sem.cnes)))/boot.summary$table[, "Std.Error"], 2)

lam[MBSA2:F] lam[MBSA7:F] lam[MBSA8:F] lam[MBSA9:F]     V[MBSA2]     V[MBSA7]
        0.84         0.88         0.90         0.96         1.25         0.68
    V[MBSA8]     V[MBSA9]
        1.04         0.75
```

3.6 Estimation of Structural Equation Models with Missing Data

3.6.1 Maximum-Likelihood Estimation with Missing Data

The development version of the **sem** package supports maximum-likelihood estimation of structural equation models in the presence of missing data. The method implemented assumes that the observed variables in the data set are multivariately normally distributed and that the missing data are *missing at random* (*MAR*) in the sense of Rubin (1976). The ML estimator in this setting is often called *full-information maximum-likelihood* (abbreviated *FIML* — see, e.g., Enders, 2001) in the structural equation modeling literature, though this term is equally descriptive of the ML estimator for models *without* missing data, because all of the equations of the model are estimated simultaneously. Maximizing the likelihood *separately* for each structural equation of the model produces the so-called *limited-information maximum-likelihood* estimator. The implementation in the **sem** package of the ML estimator in the presence of missing data is still under active development.

To illustrate, I will use a small data set of mental tests described in the **SAS** manual (SAS Institute, 2010, Example 25.13) and included in the **sem** package:

```
> head(Tests)

  x1 x2 x3 y1 y2 y3
1 23 NA 16 15 14 16
2 29 26 23 22 18 19
3 14 21 NA 15 16 18
4 20 18 17 18 21 19
5 25 26 22 NA 21 26
6 26 19 15 16 17 17

> nrow(Tests)

[1] 32
```

The first three variables in the data set, x1, x2, and x3, are meant to tap a verbal factor, while the remaining variables, y1, y2, and y3, are meant to tap a math factor. Consequently, I define a confirmatory factor analysis model for the data as follows:

```
> mod.cfa.tests <- cfa(raw=TRUE)
1: verbal: x1, x2, x3
2: math: y1, y2, y3
3:

Read 2 items
NOTE: specify fixed.x="Intercept" in call to sem
NOTE: adding 6 variances to the model

> mod.cfa.tests

   Path               Parameter      StartValue
1  verbal -> x1       lam[x1:verbal]
2  verbal -> x2       lam[x2:verbal]
3  verbal -> x3       lam[x3:verbal]
4  math -> y1         lam[y1:math]
5  math -> y2         lam[y2:math]
6  math -> y3         lam[y3:math]
7  verbal <-> verbal  <fixed>        1
8  math <-> math      <fixed>        1
9  Intercept -> x1    intercept(x1)
10 Intercept -> x2    intercept(x2)
11 Intercept -> x3    intercept(x3)
12 Intercept -> y1    intercept(y1)
13 Intercept -> y2    intercept(y2)
14 Intercept -> y3    intercept(y3)
15 verbal <-> math    C[verbal,math]
16 x1 <-> x1          V[x1]
17 x2 <-> x2          V[x2]
18 x3 <-> x3          V[x3]
19 y1 <-> y1          V[y1]
20 y2 <-> y2          V[y2]
21 y3 <-> y3          V[y3]
```

I have included an intercept in each equation because the ML estimators of the observed-variable means are not the corresponding complete-cases sample means, as would be true if there were no missing data. It is therefore necessary in the presence of missing data to model the variable means, and the model will be fit to a raw-moment matrix rather than to a covariance matrix. The `Intercept` variable in the model represents the constant regressor of 1s, which appears in the raw-moment matrix.

To fit the model, I call `sem` with the arguments `na.action=na.pass` (so that the missing data are not filtered out by the default `na.action`, which is `na.omit`); `objective=objectiveFIML` (in place of the default objective function, `objectiveML`); and `fixed.x="Intercept"` (reflecting the constant regressor in the equations of the model). Because `na.action=na.pass`, the `raw` argument to `sem` is set to `TRUE` by default, causing `sem` to generate and analyze a raw-moment matrix.

```
> cfa.tests <- sem(mod.cfa.tests, data=Tests, na.action=na.pass,
+                  objective=objectiveFIML, fixed.x="Intercept")
> summary(cfa.tests, saturated=TRUE)

Model fit to raw moment matrix.

 Model Chisquare =  6.625   Df =  8 Pr(>Chisq) = 0.5776
 AIC =  44.63
 AICc =  69.96
 BIC =  72.47
 CAIC =  -29.1

 Normalized Residuals
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.03340 -0.01570 -0.00194 -0.00166  0.00682  0.05250

 Parameter Estimates
               Estimate Std Error z value Pr(>|z|)
lam[x1:verbal]  5.5001  1.0347     5.316   1.063e-07 x1 <--- verbal
lam[x2:verbal]  5.7133  1.0155     5.626   1.846e-08 x2 <--- verbal
lam[x3:verbal]  4.4418  0.7598     5.846   5.030e-09 x3 <--- verbal
lam[y1:math]    4.9275  0.6864     7.179   7.046e-13 y1 <--- math
lam[y2:math]    4.1215  0.5746     7.172   7.374e-13 y2 <--- math
lam[y3:math]    3.3834  0.6041     5.601   2.133e-08 y3 <--- math
intercept(x1)  20.0990  1.1789    17.049   3.573e-65 x1 <--- Intercept
intercept(x2)  18.6535  1.1815    15.788   3.786e-56 x2 <--- Intercept
intercept(x3)  18.5651  0.9137    20.319   8.805e-92 x3 <--- Intercept
intercept(y1)  17.8799  0.9082    19.687   2.766e-86 y1 <--- Intercept
intercept(y2)  17.7271  0.7619    23.266  9.868e-120 y2 <--- Intercept
intercept(y3)  17.8636  0.7420    24.074  4.701e-128 y3 <--- Intercept
C[verbal,math]  0.5013  0.1515     3.309   9.358e-04 math <--> verbal
V[x1]          12.7274  4.7797     2.663   7.750e-03 x1 <--> x1
V[x2]           9.3617  4.9519     1.891   5.868e-02 x2 <--> x2
V[x3]           5.6730  2.7978     2.028   4.259e-02 x3 <--> x3
V[y1]           1.8686  1.4747     1.267   2.051e-01 y1 <--> y1
V[y2]           1.4987  1.0625     1.411   1.584e-01 y2 <--> y2
```

```
V[y3]           5.2496  1.5679     3.348   8.136e-04 y3 <--> y3

 Iterations =  134
```

The summary output for the model is familiar. The argument `saturated=TRUE` produces a comparison of the fitted model with a saturated model; the default is `FALSE` because, unlike in a model without missing data, fitting the saturated model requires an additional, often time-consuming, optimization in which there is one free parameter for each observed-variable moment.

According to the description of the `Tests` data set in the **SAS** manual, the missing data were introduced artificially and are missing completely at random (MCAR). Consequently, a complete-case analysis should produce consistent parameter estimates and valid statistical inferences, if estimates that are less efficient than the FIML estimates. I proceed to fit the CFA model to the complete cases as follows:

```
> cfa.tests.cc <- sem(mod.cfa.tests, data=Tests, raw=TRUE, fixed.x="Intercept")
```

The argument `raw=TRUE` is needed now because the default is `FALSE` for the default `objectiveML` objective function. Comparing the parameter estimates and standard errors for the two analyses, via the `compareCoefs` functions in the **car** package, reveals reasonably similar results, with most, but not all, standard errors smaller for the FIML estimator:

```
> library(car)  # for compareCoefs()
> compareCoefs(cfa.tests, cfa.tests.cc)

                Est. 1   SE 1 Est. 2   SE 2
lam[x1:verbal]  5.500  1.035  4.949  1.229
lam[x2:verbal]  5.713  1.016  5.447  1.178
lam[x3:verbal]  4.442  0.760  4.719  1.070
lam[y1:math]    4.927  0.686  4.312  0.800
lam[y2:math]    4.122  0.575  3.734  0.778
lam[y3:math]    3.383  0.604  2.550  0.695
intercept(x1)  20.099  1.179 21.750  1.480
intercept(x2)  18.654  1.182 19.375  1.492
intercept(x3)  18.565  0.914 19.313  1.331
intercept(y1)  17.880  0.908 19.000  1.093
intercept(y2)  17.727  0.762 18.125  1.011
intercept(y3)  17.864  0.742 17.750  0.822
C[verbal,math]  0.501  0.152  0.705  0.142
V[x1]          12.727  4.780 10.573  4.718
V[x2]           9.362  4.952  5.934  3.861
V[x3]           5.673  2.798  6.069  3.277
V[y1]           1.869  1.475  0.536  1.378
V[y2]           1.499  1.063  2.419  1.340
V[y3]           5.250  1.568  4.309  1.614
```

3.6.2 Multiple Imputation of Missing Data

The `miSem` function (currently in the development version of the **sem** package) uses the facilities of the **mi** package (Su et al., 2011) to create multiple imputations of missing data, and then fits a structural equation model to the completed data sets, summarizing the results. For general discussions of estimation with missing data, including multiple imputation (MI), see Little and Rubin (2002) and Allison (2002).

To illustrate, I again use the `Tests` data set, and the previously defined model `mod.cfa.tests`:

```
> imps <- miSem(mod.cfa.tests, data=Tests, fixed.x="Intercept",
+     raw=TRUE, seed=12345)

Beginning Multiple Imputation ( Thu Sep 20 19:49:36 2012 ):
Iteration 1
 Chain 1 : x1*  x2*  x3*  y1*  y2*  y3*
 Chain 2 : x1*  x2*  x3*  y1*  y2*  y3*
 Chain 3 : x1*  x2*  x3*  y1*  y2*  y3*
 Chain 4 : x1*  x2*  x3*  y1*  y2*  y3*
 Chain 5 : x1*  x2*  x3*  y1*  y2*  y3*
Iteration 2
 Chain 1 : x1*  x2*  x3   y1   y2*  y3*
 Chain 2 : x1*  x2   x3   y1   y2   y3*
 Chain 3 : x1   x2*  x3*  y1   y2   y3
 Chain 4 : x1   x2*  x3*  y1*  y2   y3
 Chain 5 : x1   x2*  x3*  y1   y2   y3*
Iteration 3
 Chain 1 : x1*  x2*  x3   y1*  y2*  y3
 Chain 2 : x1   x2   x3   y1   y2*  y3
 Chain 3 : x1*  x2*  x3   y1   y2*  y3
 Chain 4 : x1   x2*  x3   y1   y2   y3*
 Chain 5 : x1*  x2   x3   y1   y2*  y3
 . . .
Iteration 26
 Chain 1 : x1   x2   x3   y1   y2   y3
 Chain 2 : x1   x2*  x3   y1*  y2*  y3
 Chain 3 : x1   x2   x3   y1   y2   y3*
 Chain 4 : x1   x2   x3   y1   y2   y3
 Chain 5 : x1   x2*  x3   y1   y2   y3
mi converged ( Thu Sep 20 19:49:55 2012 )
Run 20 more iterations to mitigate the influence of the noise...
Beginning Multiple Imputation ( Thu Sep 20 19:49:55 2012 ):
Iteration 1
 Chain 1 : x1   x2   x3   y1   y2   y3
 Chain 2 : x1   x2   x3   y1   y2   y3
 Chain 3 : x1   x2   x3   y1   y2   y3
 Chain 4 : x1   x2   x3   y1   y2   y3
 Chain 5 : x1   x2   x3   y1   y2   y3
 . . .
Iteration 20
 Chain 1 : x1   x2   x3   y1   y2   y3
 Chain 2 : x1   x2   x3   y1   y2   y3
 Chain 3 : x1   x2   x3   y1   y2   y3
 Chain 4 : x1   x2   x3   y1   y2   y3
 Chain 5 : x1   x2   x3   y1   y2   y3
mi converged ( Thu Sep 20 19:50:09 2012 )

> summary(imps)

Coefficients by imputation:
               Imputation 1 Imputation 2 Imputation 3 Imputation 4 Imputation 5
lam[x1:verbal]        5.140        5.159        6.238         5.76         5.07
lam[x2:verbal]        5.017        5.480        5.645         5.13         5.00
lam[x3:verbal]        4.512        4.414        4.383         4.29         4.76
lam[y1:math]          4.930        4.833        4.816         4.85         4.82
lam[y2:math]          4.130        4.220        4.282         4.16         4.06
lam[y3:math]          3.375        3.345        3.432         3.37         3.35
intercept(x1)        20.147       20.312       20.003        19.93        20.19
intercept(x2)        19.180       19.413       18.543        19.24        18.78
intercept(x3)        18.348       18.451       18.349        18.33        18.51
intercept(y1)        17.889       17.725       17.846        17.81        17.76
intercept(y2)        17.713       17.769       17.810        17.73        17.65
intercept(y3)        18.031       17.882       18.016        17.84        17.50
C[verbal,math]        0.515        0.467        0.428         0.53         0.50
V[x1]                14.261       12.736        8.077        10.33        15.81
V[x2]                15.642       14.512        9.314        14.14        14.28
V[x3]                 5.116        4.537        7.968         6.03         4.36
V[y1]                 2.073        2.418        2.219         1.97         2.20
V[y2]                 1.224        1.023        1.037         1.19         1.40
V[y3]                 4.695        4.820        5.824         4.82         6.30
               Averaged Initial Fit
lam[x1:verbal]    5.473       4.949
lam[x2:verbal]    5.254       5.447
lam[x3:verbal]    4.471       4.719
lam[y1:math]      4.849       4.312
lam[y2:math]      4.170       3.734
lam[y3:math]      3.374       2.550
intercept(x1)    20.117      21.750
intercept(x2)    19.032      19.375
intercept(x3)    18.397      19.313
intercept(y1)    17.805      19.000
intercept(y2)    17.736      18.125
intercept(y3)    17.853      17.750
C[verbal,math]    0.488       0.705
V[x1]            12.244      10.573
V[x2]            13.577       5.934
V[x3]             5.603       6.069
V[y1]             2.176       0.536
V[y2]             1.174       2.419
V[y3]             5.291       4.309

Coefficients:
```

```
                Estimate Std. Error z value Pr(>|z|)
lam[x1:verbal]    5.473      1.122    4.88  1.1e-06
lam[x2:verbal]    5.254      1.025    5.13  2.9e-07
lam[x3:verbal]    4.471      0.768    5.82  5.9e-09
lam[y1:math]      4.849      0.672    7.22  5.3e-13
lam[y2:math]      4.170      0.572    7.29  3.0e-13
lam[y3:math]      3.374      0.593    5.69  1.3e-08
intercept(x1)    20.117      1.163   17.30  < 2e-16
intercept(x2)    19.032      1.201   15.85  < 2e-16
intercept(x3)    18.397      0.899   20.46  < 2e-16
intercept(y1)    17.805      0.899   19.81  < 2e-16
intercept(y2)    17.736      0.765   23.20  < 2e-16
intercept(y3)    17.853      0.760   23.50  < 2e-16
C[verbal,math]    0.488      0.155    3.15   0.0016
V[x1]            12.244      5.704    2.15   0.0318
V[x2]            13.577      5.377    2.53   0.0116
V[x3]             5.603      3.138    1.79   0.0742
V[y1]             2.176      1.258    1.73   0.0838
V[y2]             1.174      0.894    1.31   0.1890
V[y3]             5.291      1.653    3.20   0.0014
```

By default, the initial fit uses the FIML estimator when, as here, the model is fit to raw moments. For purposes of computational efficiency, this initial fit provides start values for the fits to the completed data sets, and provides a point of comparison for the MI estimator. Also by default, five multiple imputations are performed. This and other aspects of the model fit and multiple-imputation process are controlled by several optional arguments to `miSem`; for details, see `?miSem`. Because the object returned by `miSem` includes the multiple-imputation object created by the `mi` function, the facilities of the **mi** package can be used to check the quality of the imputations; see the documentation for **mi** (`help(package="mi")`) and Su et al. (2011).

3.7 Fitting Multigroup Structural Equation Models

It is fairly common to want to fit structural equation models to data divided into independent sub-samples based on the values of one or more categorical variables. The **sem** package is capable of fitting such so-called *multi-group* models. The implementation of multi-groups models in the package is quite general and can handle entirely different sub-models and variables for the groups, Typical applications, however, employ sub-models and variables that are similar, if not identical, in the various groups, and may have cross-group parameter constraints.

I will illustrate by fitting a multi-group confirmatory factor analysis model to Hozlinger and Swineford's classical mental-tests data (Holzinger and Swineford, 1939). The data are in the data frame `HS.data` in the **MBESS** package:

```
> library(MBESS)
> data(HS.data)
> head(HS.data)

  id Gender grade agey agem  school visual cubes paper flags general paragrap
1  1   Male     7   13    1 Pasteur     20    31    12     3      40        7
2  2 Female     7   13    7 Pasteur     32    21    12    17      34        5
```

```
3  3 Female     7   13    1 Pasteur     27     21     12     15       20        3
4  4   Male     7   13    2 Pasteur     32     31     16     24       42        8
5  5 Female     7   12    2 Pasteur     29     19     12      7       37        8
6  6 Female     7   14    1 Pasteur     32     20     11     18       31        3
  sentence wordc wordm addition code counting straight wordr numberr figurer
1       23    22     9       78   74      115      229   170      89      96
2       12    22     9       87   84      125      285   184      86      96
3        7    12     3       75   49       78      159   170      85      95
4       18    21    17       69   65      106      175   181      80      91
5       16    25    18       85   63      126      213   187      99     104
6       12    25     6      100   92      133      270   164      84     104
  object numberf figurew deduct numeric problemr series arithmet paperrev
1      6       9      16      3      14       34      5       24       NA
2      6       9      16      3      14       34      5       24       NA
3      1       5       6      3       9       18      7       20       NA
4      5       3      10      2      10       22      6       19       NA
5     15      14      14     29      15       19      4       20       NA
6      6       6      14      9       2       16     10       22       NA
  flagssub
1       NA
2       NA
3       NA
4       NA
5       NA
6       NA
```

The data are for grade 7 and 8 students in two schools, Pasteur and Grant-White. The various tests are meant to tap several abilities, including spatial, verbal, memory, and math factors. I consequently define a confirmatory factor analysis model as follows, with the intention of initially fitting the model independently to male and female students:

```
> mod.hs <- cfa()
1: spatial: visual, cubes, paper, flags
2: verbal: general, paragrap, sentence, wordc, wordm
3: memory: wordr, numberr, figurer, object, numberf, figurew
4: math: deduct, numeric, problemr, series, arithmet
5:

Read 4 items
NOTE: adding 20 variances to the model
```

The `multigroupModel` function in the **sem** package facilitates the definition of multigroup structural equation models, creating an object of class `"semmodList"`. Here,

```
> mod.mg <- multigroupModel(mod.hs, groups=c("Female", "Male"))
> class(mod.mg)

[1] "semmodList"
```

By default, when (as here) only one model is given in the initial arguments to `multigroupModel`, the function appends the group names to the parameters to create distinct parameters in the different groups. More generally, the initial arguments to `multigroupModel` may be named for the various groups, each specifying a corresponding intra-group model. The object returned by `multigroupModel` can then be used as the `model` argument to `sem`:

```
> sem.mg <- sem(mod.mg, data=HS.data, group="Gender",
+               formula = ~ visual + cubes + paper + flags +
+               general + paragrap + sentence + wordc + wordm +
+               wordr + numberr + figurer + object + numberf + figurew +
+               deduct + numeric + problemr + series + arithmet
+               )
> summary(sem.mg)
```

```
 Model Chisquare = 425.2  Df = 328  Pr(>Chisq) = 0.0002326
 Chisquare (null model) = 2611  Df = 380
 Goodness-of-fit index = 0.8825
 Adjusted goodness-of-fit index = 0.8567
 RMSEA index = 0.04453 90% CI: (0.03131, 0.05613)
 Bentler-Bonnett NFI = 0.8372
 Tucker-Lewis NNFI = 0.9495
 Bentler CFI = 0.9564
 SRMR = 0.0676
 AIC = 609.2
 AICc = 507.5
 BIC = 950.3

Iterations: initial fits, 366 309   final fit, 1


  Gender: Female

 Model Chisquare =  213.4   Df =  164 Pr(>Chisq) = 0.005736
 Goodness-of-fit index =  0.8836
 Adjusted goodness-of-fit index =  0.851
 RMSEA index =  0.04421   90% CI: (0.02491, 0.06008)
 Bentler-Bonnett NFI =  0.8528
 Tucker-Lewis NNFI =  0.9546
 Bentler CFI =  0.9608
 SRMR =  0.06678
 AIC =  305.4
 AICc =  253.4
 BIC =  445.4
 CAIC =  -777.8

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.3500 -0.5170  0.0000 -0.0004  0.4600  2.6300
```

```
 R-square for Endogenous Variables
  visual    cubes    paper    flags  general paragrap sentence    wordc
  0.5695   0.2999   0.2708   0.3648   0.7292   0.6853   0.7332   0.5910
   wordm    wordr  numberr  figurer   object  numberf  figurew   deduct
  0.7846   0.2527   0.1884   0.4402   0.2287   0.2955   0.1974   0.3282
 numeric problemr   series arithmet
  0.3618   0.5154   0.6391   0.4393

 Parameter Estimates
                           Estimate Std Error z value Pr(>|z|)
lam[visual:spatial].Female    5.4726  0.56504   9.685   3.479e-22
lam[cubes:spatial].Female     2.3681  0.35862   6.603   4.021e-11
lam[paper:spatial].Female     1.4427  0.23176   6.225   4.820e-10
lam[flags:spatial].Female     5.0791  0.68519   7.413   1.237e-13
lam[general:verbal].Female   10.2881  0.79023  13.019   9.526e-39
lam[paragrap:verbal].Female   2.9724  0.23977  12.397   2.718e-35
lam[sentence:verbal].Female   4.5209  0.34576  13.075   4.559e-39
lam[wordc:verbal].Female      4.3300  0.39017  11.098   1.285e-28
lam[wordm:verbal].Female      7.0221  0.50811  13.820   1.928e-43
lam[wordr:memory].Female      5.0575  0.87801   5.760   8.403e-09
lam[numberr:memory].Female    3.1664  0.64670   4.896   9.767e-07
lam[figurer:memory].Female    5.3143  0.67198   7.908   2.606e-15
lam[object:memory].Female     2.3752  0.43596   5.448   5.091e-08
lam[numberf:memory].Female    2.3787  0.37805   6.292   3.134e-10
lam[figurew:memory].Female    1.8611  0.37048   5.024   5.073e-07
lam[deduct:math].Female       9.7393  1.31094   7.429   1.092e-13
lam[numeric:math].Female      2.8898  0.36658   7.883   3.191e-15
lam[problemr:math].Female     6.2852  0.63456   9.905   3.973e-23
lam[series:math].Female       7.4354  0.64576  11.514   1.119e-30
lam[arithmet:math].Female     3.2911  0.36933   8.911   5.063e-19
C[spatial,verbal].Female      0.4947  0.07927   6.240   4.374e-10
C[spatial,memory].Female      0.6640  0.08154   8.143   3.850e-16
C[spatial,math].Female        0.7936  0.05830  13.611   3.448e-42
C[verbal,memory].Female       0.4695  0.08323   5.640   1.699e-08
C[verbal,math].Female         0.8485  0.03627  23.396  4.710e-121
C[memory,math].Female         0.6519  0.07467   8.730   2.548e-18
V[visual].Female             22.6436  3.95514   5.725   1.034e-08
V[cubes].Female              13.0909  1.67188   7.830   4.878e-15
V[paper].Female               5.6038  0.70413   7.958   1.743e-15
V[flags].Female              44.9262  5.99262   7.497   6.534e-14
V[general].Female            39.3016  5.55893   7.070   1.549e-12
V[paragrap].Female            4.0581  0.54824   7.402   1.340e-13
V[sentence].Female            7.4390  1.05743   7.035   1.993e-12
V[wordc].Female              12.9768  1.64889   7.870   3.546e-15
V[wordm].Female              13.5372  2.09442   6.463   1.023e-10
V[wordr].Female              75.6449  9.71771   7.784   7.014e-15
V[numberr].Female            43.1877  5.33321   8.098   5.592e-16
V[figurer].Female            35.9125  5.57297   6.444   1.163e-10
```

```
V[object].Female            19.0260  2.40599   7.908   2.620e-15
V[numberf].Female           13.4917  1.78890   7.542   4.633e-14
V[figurew].Female           14.0786  1.74737   8.057   7.819e-16
V[deduct].Female           194.1403 23.37476   8.306   9.935e-17
V[numeric].Female           14.7331  1.79063   8.228   1.906e-16
V[problemr].Female          37.1387  4.81325   7.716   1.201e-14
V[series].Female            31.2218  4.51088   6.921   4.470e-12
V[arithmet].Female          13.8222  1.72592   8.009   1.160e-15

lam[visual:spatial].Female  visual <--- spatial
lam[cubes:spatial].Female   cubes <--- spatial
lam[paper:spatial].Female   paper <--- spatial
lam[flags:spatial].Female   flags <--- spatial
lam[general:verbal].Female  general <--- verbal
lam[paragrap:verbal].Female paragrap <--- verbal
lam[sentence:verbal].Female sentence <--- verbal
lam[wordc:verbal].Female    wordc <--- verbal
lam[wordm:verbal].Female    wordm <--- verbal
lam[wordr:memory].Female    wordr <--- memory
lam[numberr:memory].Female  numberr <--- memory
lam[figurer:memory].Female  figurer <--- memory
lam[object:memory].Female   object <--- memory
lam[numberf:memory].Female  numberf <--- memory
lam[figurew:memory].Female  figurew <--- memory
lam[deduct:math].Female     deduct <--- math
lam[numeric:math].Female    numeric <--- math
lam[problemr:math].Female   problemr <--- math
lam[series:math].Female     series <--- math
lam[arithmet:math].Female   arithmet <--- math
C[spatial,verbal].Female    verbal <--> spatial
C[spatial,memory].Female    memory <--> spatial
C[spatial,math].Female      math <--> spatial
C[verbal,memory].Female     memory <--> verbal
C[verbal,math].Female       math <--> verbal
C[memory,math].Female       math <--> memory
V[visual].Female            visual <--> visual
V[cubes].Female             cubes <--> cubes
V[paper].Female             paper <--> paper
V[flags].Female             flags <--> flags
V[general].Female           general <--> general
V[paragrap].Female          paragrap <--> paragrap
V[sentence].Female          sentence <--> sentence
V[wordc].Female             wordc <--> wordc
V[wordm].Female             wordm <--> wordm
V[wordr].Female             wordr <--> wordr
V[numberr].Female           numberr <--> numberr
V[figurer].Female           figurer <--> figurer
V[object].Female            object <--> object
```

```
V[numberf].Female           numberf <--> numberf
V[figurew].Female           figurew <--> figurew
V[deduct].Female            deduct <--> deduct
V[numeric].Female           numeric <--> numeric
V[problemr].Female          problemr <--> problemr
V[series].Female            series <--> series
V[arithmet].Female          arithmet <--> arithmet

  Gender: Male

 Model Chisquare =  211.9   Df =  164 Pr(>Chisq) = 0.006991
 Goodness-of-fit index =  0.8813
 Adjusted goodness-of-fit index =  0.8481
 RMSEA index =  0.04486   90% CI: (0.02459, 0.06133)
 Bentler-Bonnett NFI =  0.8176
 Tucker-Lewis NNFI =  0.9429
 Bentler CFI =  0.9508
 SRMR =  0.06848
 AIC =  303.9
 AICc =  255.5
 BIC =  441.1
 CAIC =  -769.5

 Normalized Residuals
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.2900 -0.4570  0.0570  0.0269  0.5720  2.3000

 R-square for Endogenous Variables
  visual    cubes    paper    flags  general paragrap sentence    wordc
  0.4687   0.1459   0.1509   0.4123   0.7212   0.6706   0.7604   0.5041
   wordm    wordr  numberr  figurer   object  numberf  figurew   deduct
  0.6587   0.4065   0.3244   0.4041   0.2815   0.2426   0.2469   0.4761
 numeric problemr   series arithmet
  0.3552   0.3671   0.4893   0.3365

 Parameter Estimates
                          Estimate Std Error z value Pr(>|z|)
lam[visual:spatial].Male    4.5922  0.59560   7.710  1.257e-14
lam[cubes:spatial].Male     1.9245  0.47072   4.088  4.342e-05
lam[paper:spatial].Male     1.0817  0.25984   4.163  3.142e-05
lam[flags:spatial].Male     6.0457  0.83762   7.218  5.288e-13
lam[general:verbal].Male   10.7661  0.86776  12.407  2.402e-35
lam[paragrap:verbal].Male   2.7382  0.23347  11.728  9.141e-32
lam[sentence:verbal].Male   4.3893  0.33919  12.940  2.660e-38
lam[wordc:verbal].Male      4.0681  0.42575   9.555  1.235e-21
lam[wordm:verbal].Male      6.0151  0.51990  11.570  5.864e-31
lam[wordr:memory].Male      8.0586  1.07007   7.531  5.038e-14
lam[numberr:memory].Male    4.6443  0.70492   6.588  4.444e-11
```

```
lam[figurer:memory].Male    4.6147  0.61490   7.505  6.151e-14
lam[object:memory].Male     2.4132  0.39768   6.068  1.293e-09
lam[numberf:memory].Male    2.2728  0.40761   5.576  2.462e-08
lam[figurew:memory].Male    1.9805  0.35174   5.631  1.796e-08
lam[deduct:math].Male      14.0935  1.60988   8.754  2.052e-18
lam[numeric:math].Male      2.6430  0.36284   7.284  3.237e-13
lam[problemr:math].Male     5.9098  0.79512   7.433  1.065e-13
lam[series:math].Male       6.2173  0.69770   8.911  5.049e-19
lam[arithmet:math].Male     2.6476  0.37562   7.049  1.807e-12
C[spatial,verbal].Male      0.4112  0.09410   4.370  1.241e-05
C[spatial,memory].Male      0.4953  0.10080   4.914  8.922e-07
C[spatial,math].Male        0.7599  0.07531  10.090  6.102e-24
C[verbal,memory].Male       0.2168  0.09697   2.236  2.538e-02
C[verbal,math].Male         0.5385  0.07503   7.177  7.115e-13
C[memory,math].Male         0.6864  0.07299   9.405  5.211e-21
V[visual].Male             23.8998  4.21477   5.671  1.424e-08
V[cubes].Male              21.6905  2.71415   7.992  1.332e-15
V[paper].Male               6.5860  0.82633   7.970  1.584e-15
V[flags].Male              52.0891  8.30988   6.268  3.649e-10
V[general].Male            44.8079  6.94513   6.452  1.106e-10
V[paragrap].Male            3.6825  0.53363   6.901  5.172e-12
V[sentence].Male            6.0693  1.01436   5.983  2.185e-09
V[wordc].Male              16.2770  2.10824   7.721  1.157e-14
V[wordm].Male              18.7459  2.68298   6.987  2.809e-12
V[wordr].Male              94.8291 13.90769   6.818  9.202e-12
V[numberr].Male            44.9268  6.12885   7.330  2.295e-13
V[figurer].Male            31.4003  4.59403   6.835  8.199e-12
V[object].Male             14.8654  1.96875   7.551  4.331e-14
V[numberf].Male            16.1239  2.08645   7.728  1.093e-14
V[figurew].Male            11.9667  1.55219   7.710  1.262e-14
V[deduct].Male            218.5875 31.24563   6.996  2.638e-12
V[numeric].Male            12.6803  1.66728   7.605  2.840e-14
V[problemr].Male           60.2083  7.96808   7.556  4.151e-14
V[series].Male             40.3461  5.83836   6.911  4.829e-12
V[arithmet].Male           13.8248  1.80034   7.679  1.603e-14

lam[visual:spatial].Male  visual <--- spatial
lam[cubes:spatial].Male   cubes <--- spatial
lam[paper:spatial].Male   paper <--- spatial
lam[flags:spatial].Male   flags <--- spatial
lam[general:verbal].Male  general <--- verbal
lam[paragrap:verbal].Male paragrap <--- verbal
lam[sentence:verbal].Male sentence <--- verbal
lam[wordc:verbal].Male    wordc <--- verbal
lam[wordm:verbal].Male    wordm <--- verbal
lam[wordr:memory].Male    wordr <--- memory
lam[numberr:memory].Male  numberr <--- memory
lam[figurer:memory].Male  figurer <--- memory
```

```
lam[object:memory].Male   object <--- memory
lam[numberf:memory].Male  numberf <--- memory
lam[figurew:memory].Male  figurew <--- memory
lam[deduct:math].Male     deduct <--- math
lam[numeric:math].Male    numeric <--- math
lam[problemr:math].Male   problemr <--- math
lam[series:math].Male     series <--- math
lam[arithmet:math].Male   arithmet <--- math
C[spatial,verbal].Male    verbal <--> spatial
C[spatial,memory].Male    memory <--> spatial
C[spatial,math].Male      math <--> spatial
C[verbal,memory].Male     memory <--> verbal
C[verbal,math].Male       math <--> verbal
C[memory,math].Male       math <--> memory
V[visual].Male            visual <--> visual
V[cubes].Male             cubes <--> cubes
V[paper].Male             paper <--> paper
V[flags].Male             flags <--> flags
V[general].Male           general <--> general
V[paragrap].Male          paragrap <--> paragrap
V[sentence].Male          sentence <--> sentence
V[wordc].Male             wordc <--> wordc
V[wordm].Male             wordm <--> wordm
V[wordr].Male             wordr <--> wordr
V[numberr].Male           numberr <--> numberr
V[figurer].Male           figurer <--> figurer
V[object].Male            object <--> object
V[numberf].Male           numberf <--> numberf
V[figurew].Male           figurew <--> figurew
V[deduct].Male            deduct <--> deduct
V[numeric].Male           numeric <--> numeric
V[problemr].Male          problemr <--> problemr
V[series].Male            series <--> series
V[arithmet].Male          arithmet <--> arithmet
```

By employing the same names for parameters in the groups, we can create cross-group parameter constraints. For example, to constrain *all* corresponding parameters equal, I specify

```
> mod.mg.eq <- multigroupModel(mod.hs, groups=c("Female", "Male"), allEqual=TRUE)
> sem.mg.eq <- sem(mod.mg.eq, data=HS.data, group="Gender",
+              formula = ~ visual + cubes + paper + flags +
+                general + paragrap + sentence + wordc + wordm +
+                wordr + numberr + figurer + object + numberf + figurew +
+                deduct + numeric + problemr + series + arithmet
+              )
> sem.mg.eq
```

```
 Model Chisquare = 507.3  Df = 374
 lam[visual:spatial]   lam[cubes:spatial]   lam[paper:spatial]
              5.0441               2.1488               1.2670
  lam[flags:spatial]  lam[general:verbal] lam[paragrap:verbal]
              5.5058              10.4620               2.8647
lam[sentence:verbal]    lam[wordc:verbal]    lam[wordm:verbal]
              4.4614               4.2018               6.5519
   lam[wordr:memory]  lam[numberr:memory]  lam[figurer:memory]
              6.5190               3.9688               4.9357
  lam[object:memory]  lam[numberf:memory]  lam[figurew:memory]
              2.4103               2.2945               1.9277
    lam[deduct:math]    lam[numeric:math]   lam[problemr:math]
             11.4687               2.8188               6.2030
    lam[series:math]   lam[arithmet:math]    C[spatial,verbal]
              6.8467               2.9633               0.4655
   C[spatial,memory]      C[spatial,math]     C[verbal,memory]
              0.5704               0.7845               0.3446
      C[verbal,math]       C[memory,math]            V[visual]
              0.7157               0.6657              23.4621
            V[cubes]             V[paper]             V[flags]
             17.3285               6.1141              49.0979
          V[general]          V[paragrap]          V[sentence]
             43.2439               3.8560               6.7404
            V[wordc]             V[wordm]             V[wordr]
             14.6045              16.0789              87.1183
          V[numberr]           V[figurer]            V[object]
             43.9048              34.2354              16.9286
          V[numberf]           V[figurew]            V[deduct]
             14.9229              13.0245             219.6448
          V[numeric]          V[problemr]            V[series]
             13.4805              47.1317              35.9918
         V[arithmet]
             14.0203
```

The resulting model is much more parsimonious than the original one. The difference between the two models is highly statistically significant, but the BIC nevertheless strongly prefers the simpler model:

```
> anova(sem.mg.eq, sem.mg)

LR Test for Difference Between Models

          Model Df Model Chisq Df LR Chisq Pr(>Chisq)
sem.mg.eq      374         507
sem.mg         328         425 46     82.1    0.00085

> BIC(sem.mg)

[1] 950.3

> BIC(sem.mg.eq)
```

```
[1] 769.8
```

Were we seriously intereted in this analysis, we could follow up with a closer examination of the model, for example by computing modification indices (with the usual caveat concerning data dredging):

```
> modIndices(sem.mg.eq)

 Gender: Female

 5 largest modification indices, A matrix:
   wordm<-general  figurer<-spatial      visual<-math     numberr<-math
            23.67             20.44             17.80             17.46
general<-paragrap
            17.40

  5 largest modification indices, P matrix:
 wordm<->general numberf<->object object<->figurer    math<->deduct
         12.375           10.995           10.043            9.784
  verbal<->flags
          7.250


 Gender: Male

 5 largest modification indices, A matrix:
  memory<-wordr verbal<-general  verbal<-series   math<-general  deduct<-memory
          39.09           31.48           30.82           26.25           23.75

  5 largest modification indices, P matrix:
   memory<->wordr  verbal<->general  memory<->figurer   numberr<->wordr
            28.90             17.98             14.71             12.74
figurew<->numberr
            12.51
```

3.8 Arguments to the `sem` Function

I have introduced in the course of this appendix the most important arguments to the `sem` function:

`model`: a model description, typically created by `specifyEquations`, `cfa`, or `specifyModel`, possibly in combination with `multigroupModel` for a multigroup model.

`S`: the moment matrix (e.g., a covariance matrix) for the observed variables in the model. For a multigroup model, `S` is a named list of moment matrices, one for each group.

`N`: the number of observations in the data set that produced `S`. For a multigroup model, `N` is a named vector of group sample sizes.

`data`: a data frame containing the data for the model, as a preferable alternative to specifying `S` and `N`, if the original data set is available. For a multigroup model, `data` may be a single data frame containing the data for all groups, or a named list of data frames, with one data frame for each group; in the former case, the group argument is used to define the groups (see below).

raw: if TRUE, a raw-moment matrix (as opposed to covariance matrix) is analyzed. The default is FALSE, unless na.action=na.pass, which normally would entail FIML estimation in the presence of missing data.

fixed.x: a character vector of names of fixed exogenous variables (if there are any).

na.action: a function to be applied to data to process missing values. The default is na.omit, which produces a complete-case analysis.

formula: a one-sided R "model" formula, to be applied to data to create a numeric data matrix. In a multigroup model, alternatively a list one one-sided formulas can be given, to be applied individually to the groups The default is ~..

group: for a multigroup model, the name of the factor (or a variable that can be coerced to a factor) that defines groups.

As mentioned, the model argument to sem is typically an object created by specifyEquations, cfa, or specifyModel, or by multigroupModel. In the former case, the semmod method of sem is invoked, which sets up a call to the default method; in the latter case, the semmodList method is invoked, which sets up a call to the msemmod method. In principle, users can employ the default and msemmod methods of sem directly, but that isn't intended.

Some of the arguments of these various methods are not meant to be specified by the user, but others are passed from one method to another, and may be of direct use:

observed.variables: a character vector of the names of the observed variables; defaults to the row names of the S matrix, as either given directly or computed from the data.

robust: if TRUE, statistics are computed for robust standard errors and tests, and stored in the returned object. This option is only available when the data argument is supplied, in which case TRUE is the default.

debug: TRUE to show how sem codes the model and to display the iteration history; the default is FALSE.

objective: a function that returns an objective function to be minimized. The default for single-group models is objectiveML, and for multigroup models, msemObjectiveML. Other objective-function generators that are provided include objectiveGLM, objectiveFIML, and msemObjectiveGLS, which use compiled code, and objectiveML2, objectiveGLS2, objectiveFIML2, and msemObjectiveML2, which are coded purely in R. If necessary, users can provide their own objective-generator functions.

optimizer: a function to use in minimizing the objective function. The default for single-group models is optimizerSem, which, in combination with objectiveML, objectiveGLS, or objectivbeFIML uses compiled code for the optimization. The default for multigroup models is optimizerMsem, which in combination with msemObjectiveML or msemObjectiveGLS uses compiled code for the optimization. Other optimizers provided include optimizerNlm, optimizerOptim, and optimizerNlminb. If necessary, users can provide their own optimizers.

`use.means`: when raw data are supplied and intercepts are included in the model, use the observed-variable means as start values for the intercepts; the default is `TRUE`.

`analytic.gradient`: if `TRUE` (the default), then analytic first derivatives are used in the optimization of the objective function, if the optimzer employed will accept them and if the objective-function generator can compute them; otherwise numeric derivatives are used, again if the optimizer will compute them.

`warn`: if `TRUE`, warnings produced by the optimization function will be printed. This should generally not be necessary, because `sem` prints its own warning, and saves information about convergence. The default is `FALSE`.

`maxiter`: the maximum number of iterations for the optimization of the objective function, to be passed to the optimizer.

`par.size`: the anticipated relative sizes of the free parameters; if `"ones"`, a vector of 1s is used; if `"startvalues"`, taken from the start values. The default is `"startvalues"` if the largest observed variance is at least 100 times the smallest, and `"ones"` otherwise. Whether this argument is actually used depends upon the optimizer employed.

`start.tol`: if the magnitude of an automatic start value is less than `start.tol`, then it is set to `start.tol`; defaults to `1E-6`.

The following two argument are for multigroup models only:

`startvalues`: if `"initial.fit"` (the default), start values for a multi-group model are computed by first fitting the intra-group models separately by group; if `"startvalues"`, then start values are computed as for a single-group model. In some cases, the intra-group models may not be identified even if the multi-group model is, and then `startvalues="startvalues"` should be used.

`initial.maxiter`: if `startvalues="initial.fit"` for a multi-group model, then `initial.maxiter` gives the maximum number of iterations for each initial intra-group fit.

3.9 Avoiding and Solving Common Problems

Specifying and fitting a structural equation model can be a complicated process that doesn't always work out well. Sometimes the user is at fault, sometimes the data, and sometimes the software.

Some common user-related problems are:

- Trying to estimate an under-identified model. It is not always easy to figure out whether a structural equation model with latent variables is identified, but there are some easy-to-check necessary conditions for identification (see the references given in the last section of this appendix). No software can validly estimate an under-identified model.

- Misspelling a variable or parameter name. As far as `sem` is concerned, if a variable name doesn't appear in the data, then it is a latent variable. Misspelling the name of an observed variable therefore inadvertently creates a latent variable, and misspelling the name of a latent variable creates a distinct latent variable. Misspelling a parameter name is generally benign, unless an equality constraint is intended.

Remember that in `sem`, as in R generally, variable and parameter names are case-sensitive; thus, for example, `beta11`, `Beta11`, and `BETA11` represent *distinct* parameters, and if `income` is an observed variable in the data set, `Income` will be treated as a latent variable (assuming, of course, that it is not also in the data).

- Forgetting about variances and covariances. Structural equation models include parameters for variances and covariances of observed and latent variables and for error variances. In the RAM formulation of the model, these parameters are represented by double-headed arrows, self-directed for variances and error-variances, and linking two variables for covariances and error-covariances. The model-specification functions `specifyEquations`, `cfa`, and `specify-Model` by default will include error variances for endogenous variables without the user having to specify them directly. These functions also have a variety of features for conveniently specifying other variances and covariances: see, e.g., the `exog.variances`, `endog.variances`, and `covs` arguments in `?specifyEquations`, and also the `fixed.x` argument to `sem`.

Even when a model is identified and properly specified, `sem` may have trouble fitting it to data. Sometimes the problem is with the data themselves, which may be ill-conditioned — a situation analogous to close-collinearity in least-squares regression. When the data are ill-conditioned, the objective function can be nearly flat (or, in an extreme case, perfectly flat) near its minimum, and the estimates of the model parameters are consequently hard to determine (or, again in an extreme case, not unique). In other instances, the objective function may have multiple local minima, and the optimization may become trapped in a local minimum. To complicate matters, it can be difficult to distinguish a model that is under-identified *regardless* of the data from a model that is as a practical matter incapable of being estimated *because* of the data, a phenomenon sometimes termed *empirical under-identification.*

Convergence problems often can be solved by modifying some of the arguments of the `sem` function. In many of the problematic cases that I've encountered, simply setting the argument `par.size = "startvalues"` has done the trick, and I recommend trying this first. Sometimes examining how `sem` has parsed the model by setting `debug=TRUE` (which may reveal a spelling error), or examining the iteration history (via the same argument) will provide clues to solve the problem. If the observed variables in the data set have hugely different scales, then it might help to rescale some of them. Finally, the automatic start values provided by `sem` may not be reasonable. Examining these start values and providing better ones may help produce convergence.

4 Complementary Reading and References

Structural equation modeling is a large subject. Relatively brief introductions may be found in Duncan (1975) and in Fox (1984, Ch. 4). Paxton et al. (2011) present an accessible introduction to nonrecursive observed-variable structural-equation models. Bollen (1989) provides a standard book-length treatment of general structural-equation models, now slightly dated but in my opinion still unsurpassed. Most general econometric texts (e.g., Greene, 2003, Ch. 15; Judge et al., 1985, Part 5) take up at least observed-variables structural equation models.

Acknowledgments

The **sem** package for R described in this appendix was written jointly by John Fox, Zhenghua Nie, and Jarrett Byrnes. This work was partly supported by grants to John Fox from the Social Sciences and Humanities Research Council of Canada, and from the Senator William McMaster Chair in Social Statistics, McMaster University.

References

Allison, P. D. (2002). *Missing Data.* Thousand Oaks CA.

Bentler, P. M. and Dudgeon, P. (1996). Covariance structure analysis: Statistical practice, theory, and directions. *Annual Review of Psychology*, 47:563–592.

Blau, P. M. and Duncan, O. D. (1967). *The American Occupational Structure.* New York.

Bollen, K. A. (1989). *Structural Equations with Latent Variables.* New York.

Bollen, K. A. and Long, J. S., editors (1993). *Testing Structural Equation Models.* Sage, Newbury Park CA.

Canty, A. and Ripley, B. D. (2012). *boot: Bootstrap R (S-Plus) Functions.* R package version 1.3-5.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and their Application.* Cambridge University Press, Cambridge.

Duncan, O. D. (1975). *Introduction to Structural Equation Models.* Academic Press, New York.

Duncan, O. D., Haller, A. O., and Portes, A. (1968). peer influences on aspirations: a reinterpretation. *American Journal of Sociology*, 74:119–137.

Enders, C. K. (2001). A primer on maximum likelihood algorithms available for use with missing data. *Structural Equation Modeling*, 8:128–141.

Fox, J. (1984). *Linear Statistical Models and Related Methods.* New York.

Fox, J. (2010). *polycor: Polychoric and Polyserial Correlations.* R package version 0.7-8.

Fox, J., Nie, Z., and Byrnes, J. (2012). *sem: Structural Equation Models.* R package version 3.0-0.

Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression.* Sage, Thousand Oaks, CA, second edition.

Fox, J. and Weisberg, S. (2012). *Bootstrapping Regression Models in R: An Appendix to An R Companion to Applied Regression, Second Edition.* revision 5 June 2012.

Freedman, D. A. (1987). As others see us: A case study in path analysis (with discussion). *Journal of Educational Statistics*, 12:101–223.

Greene, W. H. (2003). *Econometric Analysis.* Prentice-Hall, Upper Saddle River, NJ, fifth edition.

Henningsen, A. and Hamann, J. D. (2007). systemfit: A package for estimating systems of simultaneous equations in r. *Journal of Statistical Software*, 23(4):1–40.

Holzinger, K. J. and Swineford, F. A. (1939). *A study in factor analysis: The stability of a bi-factor solution. Supplementary Education Monograph No. 48.* University of Chicago, Chicago.

Jöreskog, K. G. (1973). A general method for estimating a linear structural equation system. In Goldberger, A. S. and Duncan, O. D., editors, *Structural Equation Models in the Social Sciences*, pages 85–112. Seminar Press, New York.

Judge, G. G., Griffiths, W. E., Hill, R. C., Lütkepohl, H., and Lee, T.-C. (1985). *The Theory and Practice of Econometrics.* Wiley, New York, second edition.

Klein, L. (1950). *Eonomic Flucturations in the United States 1921–1941.* New York.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data.* Hoboken NJ.

McArdle, J. J. (1980). causual modeling applied to psychonomic systems simulation. *Behavior Research Methods and Instrumentation*, 12:193–209.

McArdle, J. J. and McDonald, R. P. (1984). some algebraic properties of the recticular action model. *Journal of Mathematical and Statistical Psychology*, 37:234–251.

McDonald, R. P. and Hartmann, W. (1992). a procedure for obtaining initial values of parameters in the RAM model. *Multivariate Behavioral Research*, 27:57–76.

Paxton, P. M., Hipp, J. R., and Marquart-Pyatt, S. (2011). *Nonrecursive Models: Endogeneity, Reciprocal Relationships, and Feedback Loops.* Thousand Oaks CA.

Raftery, A. E. (1993). Bayesian model selection in structural equation models. In Bollen, K. A. and Long, J. S., editors, *Testing Structural Equation Models*, pages 163–180. Newbury Park CA.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software*, 48(2):1–36.

Rubin, D. B. (1976). inference and missing data (with discusison). 63:581–592.

SAS Institute (2010). *SAS/STAT 9.22 User's Guide.*

Satorra, A. and Bentler, P. M. (1988). Scaling corrections for chi-square statistics in covariance structure analysis. In *Proceedings of the Business and Economics Statistics Section*, pages 308–313. American Statistical Association.

Su, Y.-S., Gelman, A., Hill, J., and Yajima, M. (2011). Multiple imputation with diagnostics (mi) in R: Opening windows into the black box. *Journal of Statistical Software*, 45(2):1–31.