Chapter 14: Structural Equation Modelling

Mark Andrews

Contents

Introduction **1**

Factor analysis **2**
The factor analysis model 2
Exploratory versus confirmatory factor analysis 4
Parameter estimation 4
Axis rotation 5
Examples 5

Mediation analysis **11**
Example 1 12
Example 2: Modelling graduate school success 18

Structural Equation Modelling **21**

References **26**

Introduction

In practice, the term *structural equation modelling* (SEM) refers to a collection of related multivariate statistical techniques. These include factor analysis, path analysis, latent variable modelling, causal modelling, and combinations thereof. There is no one definitive definition of SEM. However, as we will see with examples, all SEM models, certainly of the traditional kind, can be described by systems of linear regression models that usually, but not necessarily, involve latent, or unobserved, variables. In addition, these systems of regression models may, and from some perspectives always ought to (see Pearl 2012), represent *causal hypotheses* about the variables being modelled.

Some SEM models aim to discover a set of underlying unobserved variables that explain a set of intercorrelated observed variables. The classic example of this type of analysis is known as *factor analysis*. The seminal work on factor analysis, albeit more focused on psychometric theory rather statistics or mathematics is said to be Spearman (1904). Other SEM methods include the classic path analysis work of Sewell Wright (see, Wright 1921, 1934). In this, causal relationships between a set of observed variables are represented using systems of linear regression models, and the magnitude of direct and indirect causal effects between are then estimated. More recent major developments in SEM primarily include the introduction of the proprietary LISREL (*li*near *s*tructural *rel*ations) software (Joreskog and Van Thillo 1972) and the accompanying standardization of the SEM model specification. In the LISREL model, there are observed outcome variables assumed to be functions of latent variables, as in factor analysis, and in addition, there are systems of the regression models, as in path analysis, between the latent variables and other observed variables.

In this chapter, we will cover classical factor analysis, a special case of path analysis known as mediation analysis, and then the more general classic SEM model that includes elements of both factor analysis and path analysis. In this coverage, we will primarily use the `lavaan` R package.

Factor analysis

As a motivating example for introducing factor analysis, let us consider the following data set that is a subset of the `sat.act` data set in the `psych` package.

```
sat_act <- read_csv('data/sat_act.csv')
```

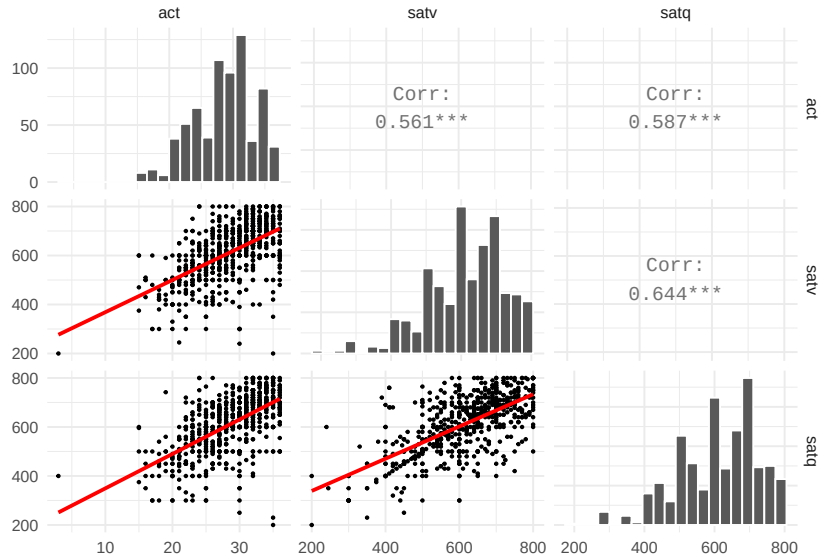


Figure 1: Pairwise scatterplots and histograms of three measures of academic ability: ACT, SAT-V, SAT-Q.

This provides us with scores from 700 students on three measures of academic ability: ACT (American College Testing), SAT-V (Scholastic Aptitude Test, Verbal), SAT-Q (Scholastic Aptitude Test, Quantitative). The histograms and inter-correlation scatterplots of the three variables are shown in Figure 1. As we can see, there is a relatively high degree of positive inter-correlation between scores on these three tests. We could hypothesize, therefore, that students' scores on these three tests are all a result of some underlying ability, which we might refer to as general academic ability. In those individuals with higher values this general academic ability, they are likely to have higher scores on ACT, SAT-V, SAT-Q. Individuals with lower values of general academic ability are likely to have lower scores on ACT, SAT-V, SAT-Q. From this perspective, general academic ability is a *latent* variable. It is not something directly unobserved, and possibly not even directly observable, but as its value changes, the values of the three test scores change too, albeit probabilistically rather than deterministically.

A latent variable model is any statistical model of a phenomenon where we assume that observed variables are probabilistic functions of unobserved variables. There are many different types of latent variable models, but factor analysis is one very widely used one. Amongst other details, as we will see, it assumes that the observed variables are linear functions, plus normally distributed random errors, of one or more latent variables. It is, therefore, essentially a multivariate linear regression model where the predictor variables are unobserved.

The factor analysis model

In factor analysis, the observed variables are vectors. In general therefore, we can denote the observed variables by $\vec{y}_1, \vec{y}_2 \ldots \vec{y}_i \ldots \vec{y}_n$, where each $\vec{y}_i$ is

$$\vec{y}_i = [y_{1i}, y_{2i} \ldots y_{di} \ldots y_{Di}]^\intercal.$$

Here, n is the number of independent observations we have, and D is the number of variables per each observation. For example, using our example above of the academic test scores, $D = 3$ and $n = 700$, and y_{di} is the score of student i on test d. We assume that each $\vec{y}_i \in \mathbb{R}^D$, where $\mathbb{R}^D$ denotes D-dimensional Euclidean

space. In other words, each element of the vector $\vec{y}_i$, i.e. each y_{di}, is assumed to be a random variable over the real line.

For each $\vec{y}_i$, there is a corresponding set of K latent variables, which we can denote as

$$\vec{x}_i = [x_{1i}, x_{2i} \dots x_{Ki}]^\intercal.$$

Here, K can be said to denote the number of *factors*, or equivalently, the dimensionality of the latent space in the model. In principle, at least if we take a Bayesian perspective on factor analysis, K can be be any positive integer. In practice, however, it is the case that $1 \leq K < D$. In fact, often we want K to be as small as possible, and certainly much smaller than D. In any case, we also assume each $\vec{x}_i \in \mathbb{R}^K$.

Each $\vec{y}_i$ is a linear function of $\vec{x}_i$ plus normally distributed errors. Given that $\vec{y}_i$ and $\vec{x}_i$ are vectors, the linear relationship between them is easiest to state using matrix notations as follows.

$$\vec{y}_i = A\vec{x}_i + \vec{b} + \vec{\epsilon}_i.$$

Here, A is a $D \times K$ matrix. This is known as the *factor loading matrix*. In addition, $\vec{b}_i$ and $\vec{\epsilon}_i$ are D dimensional vectors. As such, $\vec{y}_i = A\vec{x}_i + \vec{b} + \vec{\epsilon}_i$ can be represented as follows.

$$\begin{bmatrix} y_{1i} \\ y_{2i} \\ \vdots \\ y_{di} \\ \vdots \\ y_{Di} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1K} \\ A_{21} & A_{22} & \dots & A_{2K} \\ \vdots & \vdots & & \vdots \\ A_{d1} & A_{d2} & \dots & A_{dK} \\ \vdots & \vdots & & \vdots \\ A_{D1} & A_{D2} & \dots & A_{DK} \end{bmatrix} \begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{Ki} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \\ \vdots \\ b_D \end{bmatrix} + \begin{bmatrix} \epsilon_{1i} \\ \epsilon_{2i} \\ \vdots \\ \epsilon_{di} \\ \vdots \\ \epsilon_{Di} \end{bmatrix}.$$

Each $\vec{\epsilon}_i$ is assumed to be drawn from a D dimensional multivariate normal distribution with zero mean vector and a *diagonal* covariance matrix Φ, i.e., $\vec{\epsilon}_i \sim N(\vec{0}, \Phi)$, for each $i \in n$. As we have previously seen, this fact also entails that $\vec{y}_i$ itself has a D dimensional multivariate normal distribution, specifically

$$\vec{y}_i \sim N(A\vec{x}_i + \vec{b}, \Phi).$$

From the description of the factor analysis model thus far, we gain two important perspectives. First, we see that each y_{di} is being modelled as a normal (in the sense of normal distribution) linear regression model of $x_{1i}, x_{2i} \dots x_{Ki}$. Specifically, it is modelled as follows:

$$y_{di} = b_d + \sum_{k=1}^{K} A_{dk} x_{ki} + \epsilon_{di} \quad \epsilon_{di} \sim N(0, \phi_d^2),$$

where ϕ_d^2 is the dth element of the main diagonal of the matrix Φ. Thus, factor analysis is a just a multivariate normal linear regression model, albeit with latent predictor variables. From this perspective, and particularly because Φ is a diagonal matrix, we also see that all of $y_{1i}, y_{2i} \dots y_{di} \dots y_{Di}$ are statistically independent of one another conditional on $\vec{x}_i$. In other words, if we know the value of $\vec{x}_i$, then knowing any one or any set of $y_{1i}, y_{2i} \dots y_{di} \dots y_{Di}$ provides no information about the values of any other. The importance of this result is that all the intercorrelations between the elements of the observed variables are explained entirely by the latent variables. This, in fact, is the purpose of factor analysis. It is a means to explain intercorrelations in observed data in terms of a smaller set of latent variables.

The final defining feature of the factor analysis model is that the probability distribution over each $\vec{x}_i$ is a K-dimensional multivariate standard normal distribution. A standard multivariate normal distribution has a mean vector of all zeros, and the identity matrix as its covariance matrix. In other words, we have

$$\vec{x}_i \sim N(\vec{0}, I),$$

where $\vec{0}$ is a vector of K zeros, and I is $D \times D$ matrix with ones along the main diagonal and zeros elsewhere. It should be noted that there is no loss of generality by assuming a zero mean vector and an identity covariance

matrix. Any other choice for the mean vector and covariance matrix is equivalent to different choices of the A matrix and b vector.

From the above model, we obtain the conditional probability distribution of any $\vec{y}_i$ given $\vec{x}_i$, which we can write as $\mathrm{P}(\vec{y}_i|\vec{x}_i, A, \vec{b}, \Phi)$. We also have the marginal distribution of $\vec{x}_i$, which we write as $\mathrm{P}(\vec{x}_i|\vec{0}, I)$. Both of these are multivariate normal distributions. As such, we can calculate $\mathrm{P}(\vec{y}_i|A, b, \Phi)$, which is the marginal distribution of $\vec{y}_i$ marginalizing over $\vec{x}_i$, as follows.

$$\begin{aligned}
\mathrm{P}(\vec{y}_i|A, b, \Phi) &= \int \mathrm{P}(\vec{y}_i|\vec{x}_i, A, \vec{b}, \Phi)\mathrm{P}(\vec{x}_i|\vec{0}, I), \\
&= \int N(\vec{y}_i|A\vec{x}_i + \vec{b}, \Phi)N(\vec{x}_i|\vec{0}, I), \\
&= N(\vec{y}_i|\vec{b}, AA^\intercal + \Phi)
\end{aligned}$$

From this, we see that the unconditional or marginal probability distribution over any $\vec{y}_i$ is itself a multi-dimensional normal distribution with mean vector $\vec{b}$ and covariance matrix $AA^\intercal + \Phi$. That the covariance matrix of the elements of the observed vector can be stated as follows

$$\Sigma = AA^\intercal + \Phi,$$

is sometimes known as the *fundamental theorem of factor analysis*. From this result, if we write σ_d^2 as the dth element of Σ and ϕ_d^2 as the dth element of Φ, then we can see that the variances of element d of the observed vector can be written as follows:

$$\sigma_d^2 = \sum_{k=1}^{K} A_{dk}^2 + \phi_d^2.$$

The first term on the right hand side, $\sum_{k=1}^{K} A_{dk}^2$, is known as the *communality*, or the part of the variance of the observed element d that is explained by the latent factors. The remaining term, ϕ_d^2, is known as the *uniqueness*, or the part of the variance of observed element d that is unique and not due to the latent factors.

Exploratory versus confirmatory factor analysis

In *exploratory* factor analysis the dimensionality of the latent variable space K is assumed to be unknown, and there are no specific hypotheses about how each factor relates to the elements of the observed vectors. In *confirmatory* factor analysis, by contrast, K is assumed to be known and the A matrix is assumed to have zero elements that reflect that certain factors are assumed to not relate to certain elements of the observed vector. As an example, in a confirmatory factor analysis with $D = 5$ elements to each observed vector, we might hypothesize that there are two latent factors, and that the first factor only relates to elements 1 and 2, while the second only relates to elements 4, 4, and 5. Given these hypotheses, the A matrix has zero elements as follows.

$$\begin{bmatrix}
A_{11} & 0, \\
A_{21} & 0, \\
0 & A_{32}, \\
0 & A_{42}, \\
0 & A_{52},
\end{bmatrix}$$

A major issue in exploratory factor analysis, as we see, relates both the number of factors and the optimal rotation of the A matrix. By contrast, neither of these issues arise in confirmatory factor analysis.

Parameter estimation

There are a variety of methods used to estimate the parameters of the factor analysis model. We will only consider maximum likelihood estimation here, but this is a major and widely used method in factor analysis.

As we've seen, the observed data in the factor analysis model are the n vectors $\vec{y}_1, \vec{y}_2 \ldots \vec{y}_i \ldots \vec{y}_n$. The parameters to be estimated are the $D \times K$ matrix A, the vector with D element vector $\vec{b}$, and the $D \times D$

diagonal matrix Φ. The log of the likelihood of the data given these parameters is as follows:

$$\begin{aligned} L(A, \vec{b}, \Phi | \vec{y}_1 \dots \vec{y}_n) &= \sum_{i=1}^{n} \log \mathrm{P}(\vec{y}_i | A, b, \Phi), \\ &= -\frac{nD}{2} \log(2\pi) - \frac{n}{2} \log |\Sigma| - \frac{1}{2} \sum_{i=1}^{n} (\vec{y}_i - \vec{b})^\intercal \Sigma^{-1} (\vec{y}_i - \vec{b}). \end{aligned}$$

The maximum of this function with respect to the vector $\vec{b}$ is obtained by setting $\vec{b}$ equal to the sample mean of the observed vectors, i.e.,

$$\underset{\vec{b}}{\operatorname{argmax}} L(A, \vec{b}, \Phi | \vec{y}_1 \dots \vec{y}_n) = \bar{y} = \frac{1}{n} \sum_{i=1}^{n} \vec{y}_i.$$

By substituting $\vec{b}$ with $\bar{y}$, the log-likelihood function over A and Φ is as follows:

$$L(A, \Phi | \vec{y}_1 \dots \vec{y}_n) = -\frac{n}{2} \left(D \log(2\pi) + \log |\Sigma| + \operatorname{Tr}(\Sigma^{-1} S) \right),$$

where S is the sample covariance matrix of the data, i.e.

$$S = \frac{1}{N} \sum_{i=1}^{N} (\vec{y}_i - \bar{y})(\vec{y}_i - \bar{y})^\intercal.$$

Maximizing $L(A, \Phi | \vec{y}_1 \dots \vec{y}_n)$ with respect to A and Φ is complicated by the fact that if $\hat{A}$ maximizes this function, so too does any orthogonal rotation of $\hat{A}$, and so therefore there are an infinite number of solutions to this optimization problem. However, a common solution to this idenitifiability problem is to require the matrix $A^\intercal \Phi^{-1} A$ to be diagonal. With this constraint, Jöreskog (1967) introduced an iterative algorithm for maximum likelihood estimation. This algorithm is equivalent to positioning the latent factors on the *principal axes*: The first axis has the maximum variance, the second axis is orthogonal to the first and has the second greatest variance, and so on.

Axis rotation

In exploratory factor analysis, the estimated A matrix, and as a consequence, the axis of the latent space are not always initially ideally suited for interpretation. Ideally, we often require a so-called *simple structure* in A. This is where, for each element of the observed vector, a single factor alone primarily accounts its variance, and each factor primarily accounts for the variance of only a subset, rather than all, the observed elements.

To achieve this imprecisely defined goal of simple structure, a plethora of different rotation methods may be employed. Some of these rotations are othogonal. The most well known of these is *varimax*, which attempts to maximize the sum of variances on any given element of the observed vector. Other rotation methods are *oblique*, which means that the axes are no longer orthogonal. After an oblique rotation, values of the elements of the latent vectors are now correlated with one another. Some of the most well known of the oblique rotations are *promax* and *oblimin*.

Examples

Here, we will consider exploratory factor analysis examples. Confirmatory factor analysis, by contract, will be covered when we consider SEM more generally in a later section.

We will begin with the `sat_act` data set mentioned above. For simplicity, however, we will remove rows with `NA` elements first.

```
sat_act %<>% na.omit()
```

We will perform a factor analysis with $K = 1$ (the default), using maximum likelihood as the estimation method, and, initially, with no rotation. For this, we will use the `factanal` function from the `stats` package. In this case, it is used as follows.

```
M <- factanal(~ act + satv + satq,
              data = sat_act,
              factors = 1,
              rotate = 'none')
```

It should be noted that `factanal` will normalize the data. In other words, it will subtract the mean of the three scores from all scores, and divide by the standard deviation. This is generally done for convenience, but does not affect the results. Amongst other things, this standardization makes the sample covariance matrix identical to sample correlation matrix.

The A factor loading matrix is obtained as follows.

```
M$loadings
#>
#> Loadings:
#>      Factor1
#> act  0.715
#> satv 0.784
#> satq 0.822
#>
#>                Factor1
#> SS loadings      1.801
#> Proportion Var   0.600
```

Because we have only one factor, the communality for each element of the observed vector is the square of the values of A. In other words, the communalities for the three elements are as follows.

```
M$loadings %>% as.vector() %>% raise_to_power(2)
#> [1] 0.5107335 0.6150754 0.6749130
```

The sum of the communalities for each factor are 1.801, and these are listed under `SS loadings`. The uniqueness, which are the diagonal elements of the diagonal covariance matrix Φ is obtained as follows.

```
M$uniquenesses
#>       act      satv      satq
#> 0.4892665 0.3849246 0.3250870
```

These sum to 1.199. Given that the sum of uniquesnesses and the sum of the communalities equal the sum of the variances, from this, we can see that the communalities account for 1.801 out of 1.801 plus 1.199, which is 0.6, as given by `Proportion Var` above.

Using the factor loadings and the uniquenesses, we can estimate the values of the latent vector corresponding to $\vec{y}$ as follows.

$$\hat{\vec{x}}_i = A^\intercal \Sigma^{-1} \vec{y}_i,$$

where $\Sigma = AA^\intercal + \Psi$ as above. These estimates latent vectors can be obtained if we set `scores = "regression"` in the call of `factanal` above.

```
M <- factanal(~ act + satv + satq,
              data = sat_act,
              factors = 1,
              rotate = 'none',
              scores = 'regression')
```

The estimates are then available as `scores`. For example, we can see the first few inferred values as follows.

```
x_est <- M$scores %>% head()
x_est
#>      Factor1
#> 1 -1.0151634
```

```
#> 2 -0.1189794
#> 3 -1.3513613
#> 4 -0.6757088
#> 5 -0.1393982
#> 6  0.1717458
```

In general, given the estimated values of each $\vec{x}_i$, the predicted values of the corresponding $\vec{y}_i$ according to the model can be obtained follows.

$$\hat{\vec{y}}_i = A\hat{\vec{x}}_i + \vec{b}.$$

In the present example, the original data were standardized, as mentioned above. Therefore, we obtain predictions of $\vec{y}_i$ by $A\hat{\vec{x}}_i$, and then multiplying by the sample standard deviations and adding the sample means. In the following, we will do this for the predictions corresponding to the `x_est` above.

```
y_bar <- apply(sat_act, 2, mean)
y_sd <- apply(sat_act, 2, sd)

y_pred <- M$loadings %*% t(x_est)

sweep(y_pred, 1, y_sd, `*`) %>%
  sweep(1, y_bar, `+`) %>%
  t()
#>        act     satv     satq
#> 1 25.04539 522.1341 513.7751
#> 2 28.13944 601.7631 598.9137
#> 3 23.88467 492.2617 481.8359
#> 4 26.21735 552.2958 546.0237
#> 5 28.06895 599.9488 596.9739
#> 6 29.14317 627.5950 626.5329
```

For comparison, we can compare these predictions with the corresponding values of the original data, which are as following.

```
head(sat_act)
#> # A tibble: 6 x 3
#>     act  satv  satq
#>   <dbl> <dbl> <dbl>
#> 1    24   500   500
#> 2    35   600   500
#> 3    21   480   470
#> 4    26   550   520
#> 5    31   600   550
#> 6    28   640   640
```

Now, let us consider another example. For this, we will use the `bfi` data set from the `psych` package. This provides data on 25 personality variables from 2800 participants in a psychology study. In the following code, we select just the personality variables from `bfi`, and reverse code selected items as required.

```
data(bfi, package = 'psych')
bfi_df <- bfi %>%
  select(A1:O5) %>%
  # reverse code selected items
  mutate_at(c('A1', 'C4', 'C5', 'E1', 'E2', 'O2', 'O5'),
            ~ 7 - .)
```

In Figure 2, we show the correlation matrix heatmap of `bfi_df`. This is produced by calculating the correlation matrix use `stats::cor` and then using `geom_tile` to generate the heatmap, as in the following code.

```
bfi_df %>%
  as.matrix() %>%
  cor(use = 'complete.obs') %>%
  as_tibble(rownames = 'x') %>%
  pivot_longer(cols = -x, names_to = 'y', values_to = 'cor') %>%
  ggplot(mapping = aes(x = x, y = y, fill = cor)) +
  geom_tile(colour = 'white') +
  scale_fill_gradient(low = "white", high = "steelblue")
```

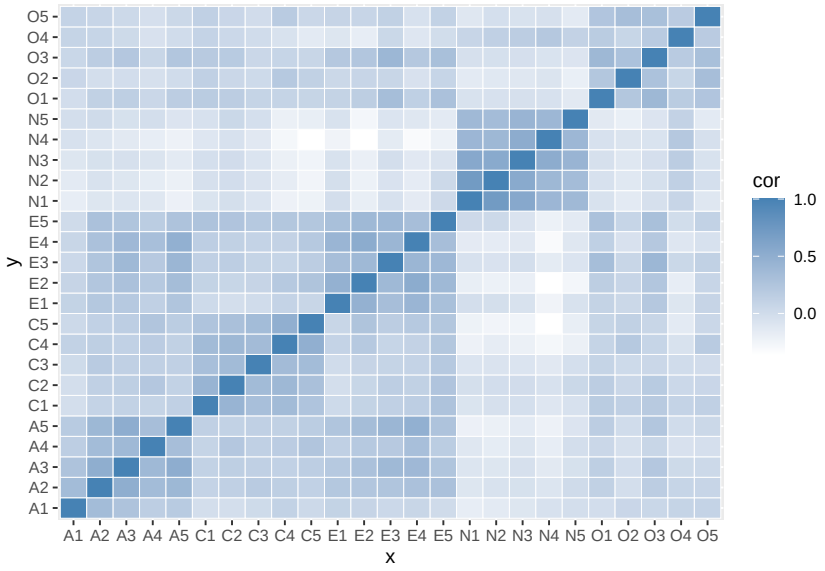


Figure 2: Heatmap of the correlation matrix of 25 personality variables.

As can be seen on the diagonal from lower left to top right, there seem to be 5 clusters of 5 items each of which have relatively high intercorrelation. However, this is not the only notable intercorrelation. There is also a relatively high intercorrelation between the 15 `A`, `C`, and `E` items. There are some negative correlation (shown by near white colours) between some sets of items.

For factor analysis of this data, we will use the `fa` function from the `psych` package. This function is more powerful and versatile than the previously used `factanal`. In the following code, we perform a factor analysis with 5 factors (`nfactors`) using factoring method of maximum likelihood (`fm = "ml"`), and also request no rotation.

```
library(psych)
M <- fa(bfi_df, nfactors = 5, fm="ml", rotate = 'none')
```

The factor loading matrix of a `psych::fa` model can viewed using the `print` function on `M$loadings` with a `cutoff` value to suppress relatively low values, as in the following example.

```
print(M$loadings, cutoff = 0.3)
```

We show the output of this `print` function in Figure 3 on the left hand side. As is clear, in this model, the values of first column of A, which corresponds to the factor labelled `ML1`, has the highest absolute values. This is confirmed by the `SS loadings`, listed at the botton of 3 left, that provide the sum of the square of these values.

```
apply(M$loadings^2, 2, sum)
#>       ML1       ML2       ML3       ML4       ML5
#> 4.3669334 2.3428640 1.5051772 1.1850394 0.9530152
```

```
Loadings:
   ML1    ML2    ML3    ML4    ML5
A1                                
A2  0.390  0.346                0.339
A3  0.460  0.391                0.328
A4  0.366                           
A5  0.540                           
C1                0.463             
C2                0.515             
C3                0.407             
C4  0.438         0.491             
C5  0.484         0.345             
E1  0.358  0.310                     
E2  0.581                     -0.329
E3  0.455  0.430                     
E4  0.539  0.337                     
E5  0.414  0.421                     
N1 -0.593  0.560                     
N2 -0.578  0.545                     
N3 -0.529  0.487                     
N4 -0.584                           
N5 -0.417  0.305                     
O1                        0.402       
O2                        0.368       
O3  0.337  0.337         0.484       
O4                        0.329       
O5                        0.435       

                ML1   ML2   ML3   ML4   ML5
SS loadings    4.367 2.343 1.505 1.185 0.953
Proportion Var 0.175 0.094 0.060 0.047 0.038
Cumulative Var 0.175 0.268 0.329 0.376 0.414
```

```
Loadings:
   ML2    ML1    ML3    ML5    ML4
A1                0.364           
A2                0.584           
A3                0.648           
A4                0.441           
A5         0.340  0.585           
C1                0.523           
C2                0.621           
C3                0.547           
C4                0.629           
C5                0.565           
E1         0.587                  
E2         0.679                  
E3         0.482         0.336  0.301
E4         0.601         0.370       
E5         0.486  0.314           
N1  0.800                         
N2  0.782                         
N3  0.716                         
N4  0.558 -0.356                  
N5  0.522                         
O1                              0.521
O2                              0.434
O3                              0.611
O4                              0.375
O5                              0.511

                ML2   ML1   ML3   ML5   ML4
SS loadings    2.668 2.254 1.967 1.947 1.518
Proportion Var 0.107 0.090 0.079 0.078 0.061
Cumulative Var 0.107 0.197 0.276 0.353 0.414
```

Figure 3: The A factor loading matrix for the factor analysis model with no rotation (left), and with *varimax* rotation (right). The rotated solution achieves a *simple structure*.

Clearly, this unrotated solution does not have a *simple structure* whereby each element of the observed variable is accounted for by primarily one factor, and each factor primarily accounts for a small subset of the observed elements. In the following code, therefore, we request a *varimax* rotation.

```
Mv <- fa(bfi_df, nfactors = 5, fm="ml", rotate = 'varimax')
```

Again, we can print this loading matrix in the following code, and this is shown in Figure 3 on the right hand side.

```
print(Mv$loadings, cutoff = 0.3)
```

In this case, we can see that each variable is primarily accounted for one factor and each factor accounts for a small number of items. Moreover, and not unexpectedly given our knowledge of this data set, each factor primarily accounts for all the items in one of the `A`, `C`, `E`, `N`, or `O` sets.

Model fit statistics

To evaluate model fits in factor analysis, we can in principle use methods that are standard throughout all of statistics for model evaluation. Nonetheless, in factor analysis and, as we will see, in SEM generally, a particular special set of model fit indices are widely used. There are, in fact, dozens of these *global fit indices*, but here we will concentrate on some of the more widely used ones.

The *model chi-square* is defined as

$$\chi^2_M = n' f_{\text{mle}},$$

where f_{mle} is the minimum value of the objective function that is being minimized to maximize the log of likelihood. In other words, to maximize the log of the likelihood in a factor analysis, an alternative function that is the negative of the log of the likelihood, plus some constant terms, is minimized. The value of this

objective function at its minimum, which is equivalent to the value of this objective function using the maximum likelihood estimates of the parameters, is f_{mle}. The value of n' on the other hand is primarily based on the sample size. In some cases, n' is exactly the sample size, i.e. n using our terminology above. In other cases, it is the sample size minus one. In others, it is $n - 1 - (2D + 5)/6 - 2K/3$, and this is what is used in `psych::fa`.

In `psych::fa`, we can obtain the value of f_{mle} as follows.

```
Mv$objective
#> [1] 0.6279861
```

The value of χ^2_M is obtained as follows.

```
Mv$STATISTIC
#> [1] 1749.883
```

Given that χ^2_M a function of f_{mle} scaled primarily by sample size, then the *lower* the χ^2_M, the better the fit. Furthermore, for the hypothesis that the model is an exact fit of the observed data, χ^2_M will be distributed as χ^2 distribution whose degrees of freedom are so-called *model degrees of freedom*. The model degrees of freedom are the number of observed correlations in the data minus the number of parameters in the model.

$$\underbrace{\frac{D(D-1)}{2}}_{\text{Correlations}} - \underbrace{\left(DK - \frac{K(K-1)}{2}\right)}_{\text{parameters}}.$$

Note that the number of parameters is less than $D \times K$ given the constraints that we impose on the A matrix. In `psych::fa`, the model degrees of freedom are obtained as follows.

```
Mv$dof
#> [1] 185
```

Thus, according to the hypothesis of exact fit, the expected value of χ^2_M will be 185. If χ^2_M was exactly 185, this would correspond to a p-value of close to 0.5. On the other hand, values of χ^2_M much greater than the expected value of 185 will correspond to low p-values. In the case of model `Mv`, the value of χ^2_M is much greater than 185 and so the corresponding p-value is very low. We can obtain this p-value as follows.

```
Mv$PVAL
#> [1] 1.394484e-252
```

In addition to χ^2_M, the *root mean square error of approximation* (RMSEA) is a widely used measure of model fit. It is defined as follows.

$$\text{rmsea} = \sqrt{\frac{\chi^2_M - \text{df}_M}{\text{df}_M(N-1)}},$$

where df_M is the model's degrees of freedom. However, if $\chi^2_M < \text{df}_M$, rmsea is defined as zero. In `psych::fa`, RMSEA is calculated by the following related formula.

$$\sqrt{\frac{f_{\text{mle}}}{\text{df}_M} - \frac{1}{\text{df}_M - 1}}$$

This can be obtained as follows, which provides the RMSEA score and also the 90% confidence interval.

```
Mv$RMSEA
#>      RMSEA      lower      upper confidence
#> 0.05496255 0.05263677 0.05734094 0.90000000
```

RMSEA is usually interpreted as departure from close, as opposed to perfect, fit. Thus, the greater the value of $\chi^2_M - \text{df}_M$, the further the departure from close fit. There is no consensus on what counts sufficiently low values of RMSEA to indicate a good fit, but traditionally, values less than 0.05 or 0.01 are usually taken to indicate good and very good fits, respectively.

Inferring the number of factors

Thus far we have assumed that the number of factor is known. This is often not the case. Ultimately, the problem of inferring or estimating the number of parameters is an example of the standard problem of model comparison that is ubiquitous problem in statistics generally. However, in the context of (exploratory) factor analysis, a number of special procedures are usually followed to decide on the number of factors. Here, we will describe the method of *parallel analysis* Horn (1965) implemented using `psych::fa.parallel`. This method is related to the widely used *scree* method whereby eigenvalues of a principal axis factoring in the factor analysis are plotted in descending order. The eigenvalues will indicate the amount of variance accounted for by each of the principal axes. Usually, we simply look to try to indentify where these eigenvalues begin to tail off. By contrast, the parallel analysis compares the scree plot to eigenvalues from principal axis factoring of random correlation matrices of the same size as that of the data.

In the following code, we perform the same factor analyses as we used previously, and compare the scree plot these factor analyses to the average scree plot of the from `n.iter = 100` random matrices.

```
fa.parallel(bfi_df, fm = 'ml', fa = 'fa', n.iter = 100)
```

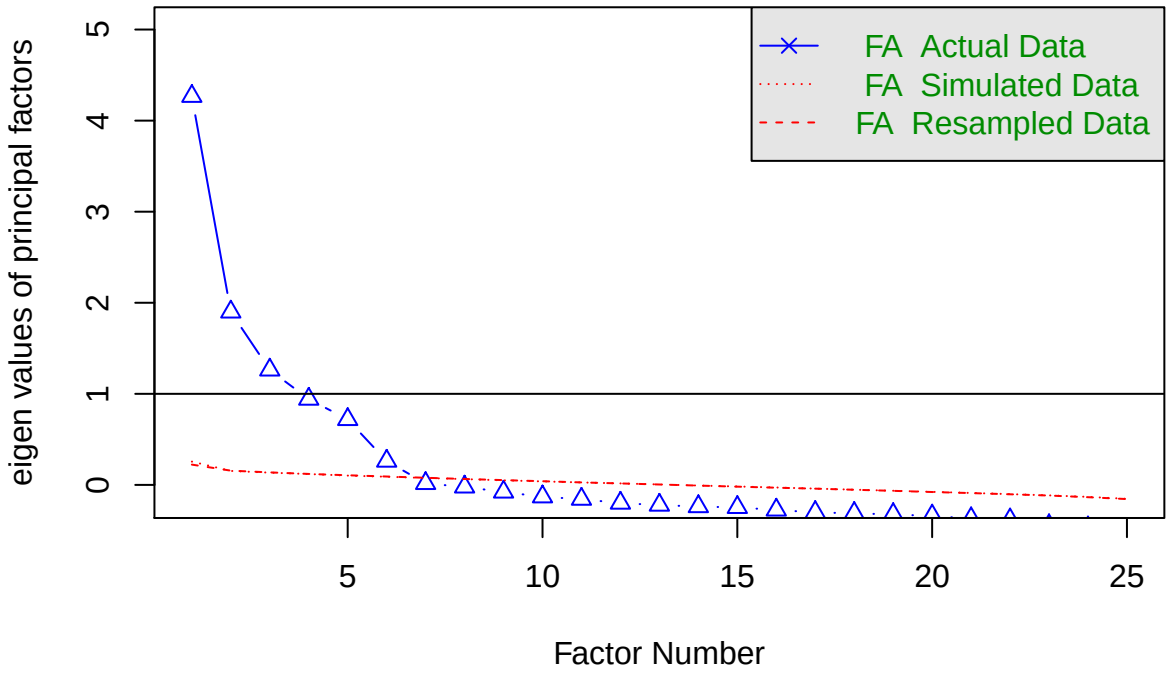


Figure 4: Parallel analyses scree plots to indentify the number of factors to use the factor analysis.

The resulting scree plots are shown in Figure 4. As we can see, 6 factors have eigenvalues greater than the corresponding average eigenvalues of the random matrices.

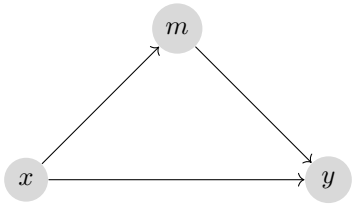
Mediation analysis

In a mediation model, the effect of one variable x on another y is due to its effect on a third variable m, which then affects y. Changes in the variable x lead to changes in m that then lead to changes in y. As

an example of a mediation effect, it is widely appreciated that tobacco smoking x raises the probability of lung cancer y, and that this effect is due to tar (tobacco residue) produced by the burning of the tobacco accumulating the lungs m. This tar contains the carcinogenic substances that cause the lung cancer.

In general, a mediation model describes a chain of effects. One possibility, known as *pure* or *full* mediation model, assumes that the effect of x on y is entirely due to its effect on m. This can be depicted by the following path diagram.

Another possibility is a *partial mediation* model. In this case, we assume that x affects m and m affects y as before, but there is also a direct effect of x on y, as in the following diagram.



Assuming that we are dealing with a normal linear model[1], we can write a pure mediation model as follows:

$$\begin{aligned} \text{for } i \ldots 1 \ldots n, \quad y_i &\sim N(\mu_i^y, \sigma_y^2), \quad \mu_i^y = \beta_{y0} + \beta_{ym} m_i, \\ m_i &\sim N(\mu_i^m, \sigma_m^2), \quad \mu_i^m = \beta_{m0} + \beta_{mx} x_i, \end{aligned}$$

which can also be written

$$\begin{aligned} \text{for } i \ldots 1 \ldots n, \quad y_i &= \beta_{y0} + \beta_{ym} m_i + \epsilon_i^y, \quad \epsilon_i^y \sim N(0, \sigma_y^2), \\ m_i &= \beta_{m0} + \beta_{mx} x_i + \epsilon_i^m, \quad \epsilon_i^m \sim N(0, \sigma_m^2). \end{aligned}$$

By contrast, the partial mediation model can be written as follows.

$$\begin{aligned} \text{for } i \ldots 1 \ldots n, \quad y_i &\sim N(\mu_i^y, \sigma_y^2), \quad \mu_i^y = \beta_{y0} + \beta_{ym} m_i + \beta_{yx} x_i, \\ m_i &\sim N(\mu_i^m, \sigma_m^2), \quad \mu_i^m = \beta_{m0} + \beta_{mx} x_i, \end{aligned}$$

or equivalently as

$$\begin{aligned} \text{for } i \ldots 1 \ldots n, \quad y_i &= \beta_{y0} + \beta_{ym} m_i + \beta_{yx} x_i + \epsilon_i^y, \quad \epsilon_i^y \sim N(0, \sigma_y^2), \\ m_i &= \beta_{m0} + \beta_{mx} x_i + \epsilon_i^m, \quad \epsilon_i^m \sim N(0, \sigma_m^2). \end{aligned}$$

A note on nomenclature. In the mathematical descriptions of structural equation models and related models, there is an avoidable proliferation of symbols, subscripts and superscripts. For the most part, we aim to keep the notation and symbols as consistent with other models as possible. For example, we will continue to use β for coeficients. When will use double subscripts for the coefficients to indicate that it is the coefficient to one node from another. For example, by β_{yx}, we mean the coefficient to y from x, and by β_{mx}, we mean the coefficient to m from x. For intercept terms, which are not *from* anywhere, we will write, for example, β_{m0} or β_{y0}

Example 1

In order to explore mediation models, let us begin with data generated according to a specific model. While our aim is always to model real world data, using generated data can be very useful when we are learning how and why the model works. The data we will generate is from a partial mediation model.

[1] There is no necessary restriction to normal and linear models in mediation analysis or in structural equation modelling generally.

```
N <- 100

b_m0 <- 1.25; b_mx <- 1.25;
b_y0 <- -0.5; b_ym <- 1.75; b_yx <- 0.75;
sigma_m <- 1.5; sigma_y <- 2.0

mediation_df <- tibble(x = rnorm(N, sd = 2),
                       m = b_m0 + b_mx * x + rnorm(N, sd = sigma_m),
                       y = b_y0 + b_ym * m + b_yx * x + rnorm(N, sd = sigma_y)
)
```

Let us now set up this model using `lavaan`.

```
library(lavaan)

mediation_model_spec_1 <- '
y ~ m + x
m ~ x
'
```

Notice that what have done so far is to simply creare a string `mediation_model_spec_1`. In this string, we write two R formulas, using the same syntax that we have used for formulas all models so far, i.e. using the `~` symbol. This symbol has an identical interpretation to how it used in, for example, `lm`, `glm`, etc. For example, `m ~ x` means `m` is regressed on `x` and, or `m` is dependent on `x`, or `m` is a random variable that is a function of `x`, etc. As we will see, there are more model specification symbols in `lavaan` than are usually used in regression models in R, but for the present mediation model, we only need the `~` symbol. Thus, to specify the partial mediation model, we need only state that y is dependent on `m` and `x`, and `m` is dependent on `x`. Just as in, for example, linear regression using `lm`, the presence of an intercept term is assumed. In other words, by writing `y ~ m + x`, we are assuming that for each i, $y_i = \beta_{y0} + \beta_{ym} m_i + \beta_{yx} x_i + \epsilon_i^y$. However, by default, unless we explicitly state in the formula that we are using an intercept term, we will not get information about it. Therefore, we can re-write `mediation_model_spec_1` as follows.

```
mediation_model_spec_1 <- '
y ~ 1 + m + x
m ~ 1 + x
'
```

Now that we have a model specification, we call `lavaan::sem` with reference to `mediation_model_spec_1`, and this fits the model using maximum likelihood estimation.

```
mediation_model_1 <- sem(mediation_model_spec_1,
                         data = mediation_df)
```

For now, let us just look at the parameter estimates of `mediation_model_1`.

```
parameterEstimates(mediation_model_1)
#>   lhs op rhs    est    se      z pvalue ci.lower ci.upper
#> 1   y ~1     -0.493 0.243 -2.030  0.042   -0.970   -0.017
#> 2   y  ~   m  1.843 0.118 15.648  0.000    1.612    2.073
#> 3   y  ~   x  0.639 0.180  3.558  0.000    0.287    0.991
#> 4   m ~1      1.256 0.164  7.665  0.000    0.935    1.577
#> 5   m  ~   x  1.280 0.083 15.471  0.000    1.118    1.443
#> 6   y ~~   y  3.633 0.514  7.071  0.000    2.626    4.640
#> 7   m ~~   m  2.620 0.371  7.071  0.000    1.894    3.347
#> 8   x ~~   x  3.826 0.000     NA     NA    3.826    3.826
#> 9   x ~1      0.305 0.000     NA     NA    0.305    0.305
```

The first thing to note about this output is that it gives us the estimates for all our coefficients, and also the

variances σ_y^2 and σ_m^2. These variances are labelled in the output with `y ~~ y` and `m ~~ m`, which is `lavaan` syntax for specifying a variance or covariance, as we will see. It also provides estimates of the mean and variance x. This is an important point in that it shows that it is creating a probabilistic model for all three variables in the model. In regression models, by contrast, variables like x are treated as given and their values are not modelled. Next, we note that, as expected, the estimated values of coefficients and variances are all close to the true values that we used to generate the data.

Model comparison

In mediation analysis, a major aim is evaluating first whether there is evidence of a mediation of the effect of x on y by m, and then whether this is pure or partial mediation. To do so, we first specify and then fit the full mediation model using similar syntax and commands to what we used for `mediation_model_spec_1` an d `mediation_model_0`.

```
mediation_model_spec_0 <- '
  y ~ 1 + m
  m ~ 1 + x
'
mediation_model_0 <- sem(mediation_model_spec_0,
                         data = mediation_df)
```

Now, let us look at how well these two models fit the data using AIC.

```
mediation_models <- c(model_0 = mediation_model_0,
                      model_1 = mediation_model_1)

map_dbl(mediation_models, AIC)
#>  model_0  model_1
#> 816.8326 806.9121
```

As we can see, the AIC for `mediation_model_1` is lower than that of `mediation_model_0` by approximately 9.92. By the standards of AIC where differences of 10 or more indicate that the model with the lower value is overwhelmingly better able to generalize to new data, this indicatest that the data is explained best by a partial rather than a pure mediation model.

Even if we just have three variables x, y, and m, and assume that there may be, or may not be, a directed arrow between x and m, m and y, and x and y, then there are exactly $2^3 = 8$ possible models to consider. These are shown in shown in Figure 5. In the following code, we create a list with 8 elements that are the specification strings for each of these 8 model versions (using the same a-h labels as in Figure 5), explicitly stating the intercept terms for each one.

```
mediation_models_specs <- within(list(),{

  model_a <- '
    x ~ 1
    m ~ 1
    y ~ 1
  '
  model_b <- '
    x ~ 1
    m ~ 1 + x
    y ~ 1
  '
  model_c <- '
    x ~ 1
    m ~ 1
    y ~ 1 + m
```

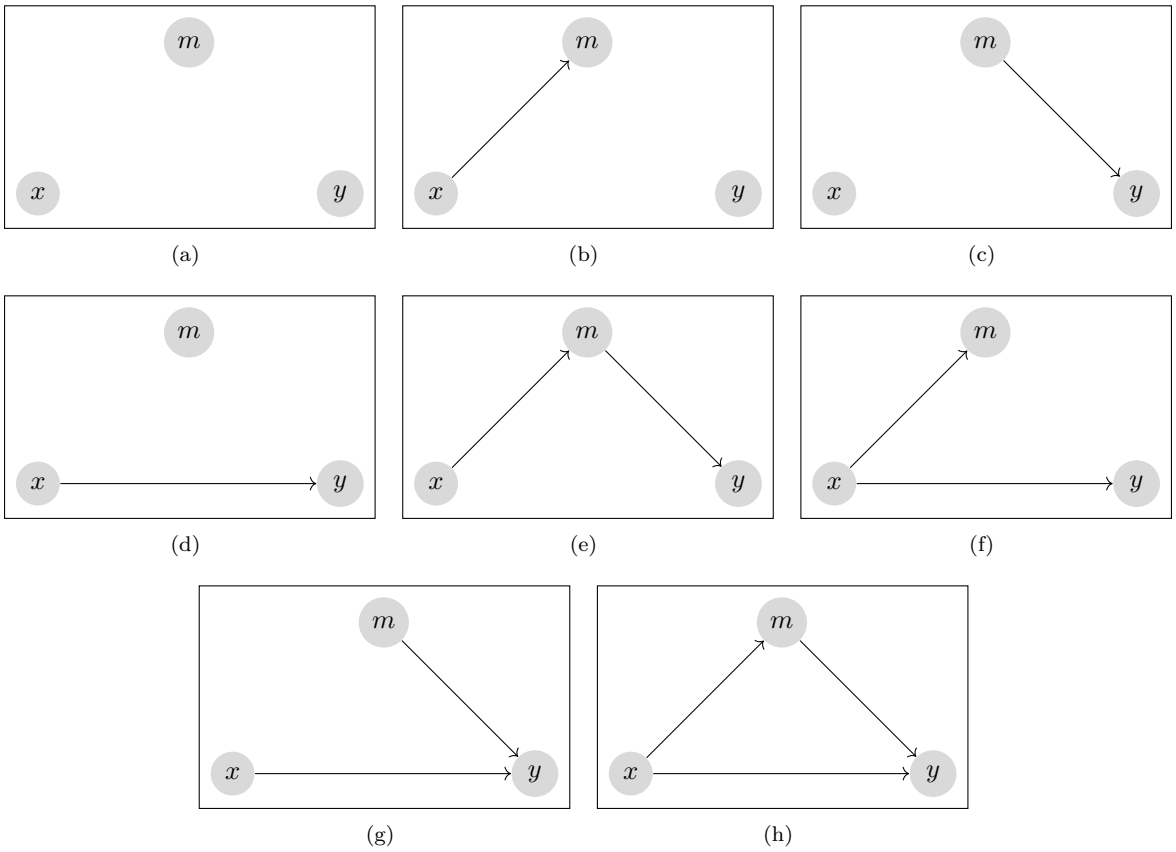


Figure 5: The $2^3 = 8$ possible versions of a simple mediation model with variables x, m, and y, assuming that there may be or may not be a directed arrow between x and m, m and y, and x and y. In Version (a), all three variables are independnet. Version (e) is the pure mediation model. Version (g) is equivalent to a regression model, but where x and m have models. Version (h) is the partial mediation model.

```
'
model_d <- '
  x ~ 1
  m ~ 1
  y ~ 1 + x
'
model_e <- '
  x ~ 1
  m ~ 1 + x
  y ~ 1 + m
'
model_f <- '
  x ~ 1
  m ~ 1 + x
  y ~ 1 + x

  # Force independence of y and m
  y ~~ 0*m
'
model_g <- '
```

```
    x ~ 1
    m ~ 1
    y ~ 1 + x + m
  '
  model_h <- '
    x ~ 1
    m ~ 1 + x
    y ~ 1 + x + m
  '
})
```

Having each model specified as an element of a list, we now use `purrr::map` to fit each model and calculate its AIC score.

```
mediation_models <- map(mediation_models_specs,
                        ~sem(., data = mediation_df)
)

map_dbl(mediation_models, AIC) %>%
  sort()
#>  model_h  model_e  model_g  model_f  model_c  model_d  model_b  model_a
#> 1228.884 1238.804 1349.074 1350.682 1358.995 1470.873 1480.709 1600.900
```

These results are as we expect. The version with the lowest AIC is `model_h`, which is the partial mediation model. This is followed by `model_e`, which is the full mediation model.

Before we proceed, let us now generate some data, using a procedure similar to above, from a pure mediation model, and then fit all 8 possible versions of the x, m, y mediation model to the data, and evaluate the fit.

```
mediation_df_new <- tibble(x = rnorm(N, sd = 2),
                           m = b_m0 + b_mx * x + rnorm(N, sd = sigma_m),
                           y = b_y0 + b_ym * m + rnorm(N, sd = sigma_y)
)
```

We used all the same settings to generate `mediation_df_new` as we used to generate `mediation_df`, but we removed the `b_yx * x` term, which is equivalent to setting `b_yx` to zero. Now, let us fit the 8 models to `mediation_df_new` and evaluate the fit.

```
mediation_models_new <- map(mediation_models_specs,
                            ~sem(., data = mediation_df_new)
)

map_dbl(mediation_models_new, AIC) %>%
  sort()
#>  model_e  model_h  model_f  model_c  model_g  model_b  model_d  model_a
#> 1268.894 1270.717 1363.463 1401.127 1402.949 1466.873 1495.696 1599.106
```

Results here are close to what we would expect. The version with the lowest AIC is `model_e`, which is the pure mediation model. This is followed by `model_h`, which is the partial mediation model. Note, however, that the AIC value of the partial mediation model is 1270.72, while the AIC of the full model is 1268.89, which is only 1.82 less. By the standards of AIC, this is not a noteworthy difference and thus there is not much to distinguish between `model_e` and `model_h`. However, it is interesting to examine the parameter estimates of `model_h`.

```
mediation_models_new %>%
  extract2('model_h') %>%
  parameterEstimates()
#>   lhs op rhs    est    se      z pvalue ci.lower ci.upper
```

```
#> 1   x ~1     -0.186 0.209 -0.892  0.372   -0.596    0.223
#> 2   m ~1      1.049 0.161  6.501  0.000    0.733    1.365
#> 3   m  ~   x  1.293 0.077 16.816  0.000    1.143    1.444
#> 4   y ~1     -0.335 0.265 -1.263  0.207   -0.855    0.185
#> 5   y  ~   x  0.087 0.207  0.422  0.673   -0.319    0.494
#> 6   y  ~   m  1.732 0.138 12.567  0.000    1.462    2.002
#> 7   m ~~   m  2.583 0.365  7.071  0.000    1.867    3.299
#> 8   y ~~   y  4.906 0.694  7.071  0.000    3.546    6.266
#> 9   x ~~   x  4.367 0.618  7.071  0.000    3.156    5.577
```

As we can see, the parameter estimate for β_{yx} is close to zero, with an estimate of 0.087 and a confidence interval of $(-0.319, 0.494)$. In that sense, `model_h` is essentially a pure mediation model.

Direct versus indirect effects

In a standard linear regression model of the following kind

$$y_i \sim N(\mu_i, \sigma^2), \quad \mu_i = \beta_0 + \beta_1 x_i, \quad i \in 1 \dots n,$$

a change in any x_i by 1 unit, i.e., $x_i + 1$, would always lead to a change of β_1 in the expected, i.e. the average, value of the outcome variable. This is easy to see. Let $x'_i = x_i + 1$, and

$$\begin{aligned} \mu_i = \beta_0 + \beta_1 x_i, \quad \mu'_i &= \beta_0 + \beta_1 x'_i, \\ &= \beta_0 + \beta_1 (x_i + 1), \\ &= \beta_0 + \beta_1 x_i + \beta_1, \\ &= \mu_i + \beta_1, \end{aligned}$$

and so $\mu' - \mu = \beta_1$. Regardless of how many predictor variables there are in the linear regression, a change in predictor k by one unit, always leads to a change β_k in the average value of the outcome variable. By contrast, in a mediation model, whether full or partial, the effect of a change in the predictor x on the outcome y is not as simple. First, let us consider a pure mediation model. In this case, as we have seen, we can write each y_i and m_i as follows

$$\begin{aligned} y_i &= \beta_{y0} + \beta_{ym} m_i + \epsilon_i^y, \quad \epsilon_i^y \sim N(0, \sigma_y^2), \\ m_i &= \beta_{m0} + \beta_{mx} x_i + \epsilon_i^m, \quad \epsilon_i^m \sim N(0, \sigma_m^2). \end{aligned}$$

From this, we have

$$\begin{aligned} y_i &= \beta_{y0} + \beta_{ym} \left(\beta_{m0} + \beta_{mx} x_i + \epsilon_i^m\right) + \epsilon_i^y, \\ &= \beta_{y0} + \beta_{ym}\beta_{m0} + \beta_{ym}\beta_{mx} x_i + \beta_{ym}\epsilon_i^m + \epsilon_i^y, \end{aligned}$$

and this entails

$$y_i \sim N(\mu_i, \beta_{ym}^2 \sigma_m^2 + \sigma_y^2), \quad \mu_i = \beta_{y0} + \beta_{ym}\beta_{m0} + \beta_{ym}\beta_{mx} x_i.$$

Following the same reasoning as above for the case of standard linear regression, this entails that in a pure mediation model a unit change in x_i leads to a change of $\beta_{ym}\beta_{mx}$ in the expected value of y. In the case of the partial mediation model, we saw already that each y_i and m_i in the model can be defined as follows:

$$\begin{aligned} y_i &= \beta_{y0} + \beta_{ym} m_i + \beta_{yx} x_i + \epsilon_i^y, \quad \epsilon_i^y \sim N(0, \sigma_y^2), \\ m_i &= \beta_{m0} + \beta_{mx} x_i + \epsilon_i^m, \quad \epsilon_i^m \sim N(0, \sigma_m^2). \end{aligned}$$

From this, we have

$$\begin{aligned} y_i &= \beta_{y0} + \beta_{ym}(\beta_{m0} + \beta_{mx} x_i + \epsilon_i^m) + \beta_{yx} x_i + \epsilon_i^y, \\ y_i &= \beta_{y0} + \beta_{ym}\beta_{m0} + \beta_{ym}\beta_{mx} x_i + \beta_{yx} x_i + \beta_{ym}\epsilon_i^m + \epsilon_i^y, \end{aligned}$$

which entails

$$y_i \sim N(\mu_i, \beta_{ym}^2\sigma_m^2 + \sigma_y^2), \quad \mu_i = \beta_{y0} + \beta_{ym}\beta_{m0} + (\beta_{ym}\beta_{mx} + \beta_{yx})\, x_i,$$

and following the reasoning above, this entails that unit change in x_i leads to a change of $(\beta_{ym}\beta_{mx} + \beta_{yx})$ in the expected values of y_i. In general in a mediation model, we have following:

$$\underbrace{\overbrace{\beta_{ym}\beta_{mx}}^{\text{indirect effect}} + \overbrace{\beta_{yx}}^{\text{direct effect}}}_{\text{total effect}}.$$

If there is no direct effect, as would be the case in pure mediation model, then the total effect is equal to the indirect effect.

In a `lavaan` mediation model, we can create single variables that measure the direct, indirect and total effects. To do so, we must first use labels for our original parameters, i.e. the coefficients, and then use the `:=` operator to create new variables that are functions of the original parameters.

```
mediation_model_spec_1 <- '
y ~ 1 + b_ym * m + b_yx * x
m ~ 1 + b_mx * x

# Define effects
indirect := b_ym * b_mx
direct   := b_yx
total    := b_yx + (b_ym * b_mx)
'
```

We can fit this model as per usual.

```
mediation_model_1 <- sem(mediation_model_spec_1,
                         data = mediation_df)
```

In the usual parameter estimates output, we can use `dplyr::filter` to isolate these effects:

```
parameterEstimates(mediation_model_1) %>%
  filter(label %in% c('indirect', 'direct', 'total')) %>%
  select(label:ci.upper)
#>      label   est    se      z pvalue ci.lower ci.upper
#> 1 indirect 2.359 0.214 11.002      0    1.939    2.779
#> 2   direct 0.639 0.180  3.558      0    0.287    0.991
#> 3    total 2.998 0.181 16.566      0    2.643    3.353
```

As we can see, for example, the estimated effect for the total effect is 2.998, and the 95% confidence interval on this effect is $(2.643, 3.353)$.

Example 2: Modelling graduate school success

In the following `gre_df` data set, we have grade point average (GPA) scores for high school (`hs`), college (`col`), graduate school (`grad`), and Graduate Record Examination (GRE) scores (`gre`) from 200 individuals.

```
grad_df <- read_csv('data/grad.csv')
```

The scatterplot matrix, histograms, and intercorrelation matrix for these four variables are shown in Figure 6. As is clear from these plots, there is a high positive intercorrelation between all four variables.

To explore this data, we first perform a standard multiple linear regression predicting `grad` from all other variables.

```
coefs_summary <- function(model) summary(model)$coefficients

lm(grad ~ ., data = grad_df) %>%
```

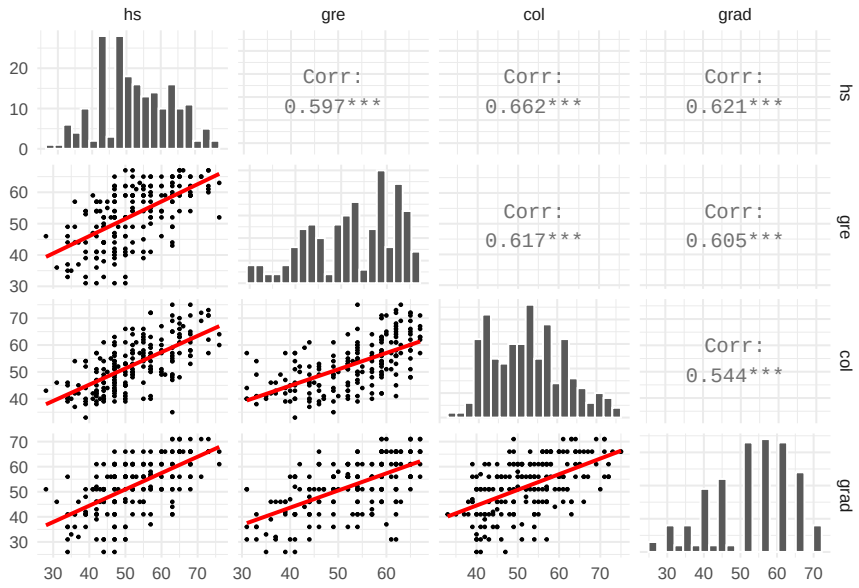


Figure 6: Scatterplot matrix, histograms, and intercorrelation matrix for the `hs`, `gre`, `col`, and `grad` scores.

```
  coefs_summary()
#>               Estimate Std. Error  t value     Pr(>|t|)
#> (Intercept) 6.9711123 3.54114285 1.968605 5.040816e-02
#> hs          0.3723267 0.07617483 4.887792 2.116731e-06
#> gre         0.3694099 0.07848502 4.706756 4.749912e-06
#> col         0.1233100 0.08504079 1.450010 1.486539e-01
```

From this, we see that when `hs` and `gre` are known, `col` does not tell us much about variability in `grad` scores, and if we drop `col`, as we do in the following code, there is no virtually no change in AIC scores.

```
lm(grad ~ ., data = grad_df) %>% drop1('col')
#> Single term deletions
#>
#> Model:
#> grad ~ hs + gre + col
#>        Df Sum of Sq   RSS    AIC
#> <none>              12000 826.86
#> col     1    128.72 12128 827.00
```

On the other hand, we see that `hs` and `col` together are predictors of `gre` scores.

```
lm(gre ~ hs + col, data = grad_df) %>%
  coefs_summary()
#>               Estimate Std. Error  t value     Pr(>|t|)
#> (Intercept) 15.5338947 3.01804716 5.147002 6.382242e-07
#> hs           0.3093503 0.06554338 4.719779 4.471315e-06
#> col          0.4004889 0.07173143 5.583172 7.756667e-08
```

Droping either `hs` or `col` from this model would lead to a substantial increase in AIC scores.

```
lm(gre ~ hs + col, data = grad_df) %>%
  drop1()
#> Single term deletions
#>
```

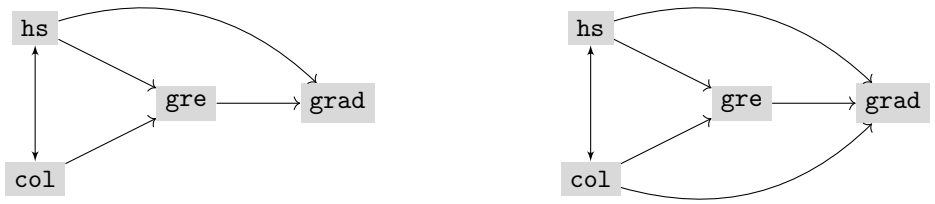
```
#> Model:
#> gre ~ hs + col
#>        Df Sum of Sq     RSS    AIC
#> <none>               9938.8 787.18
#> hs      1    1123.9 11062.7 806.60
#> col     1    1572.6 11511.5 814.56
```

From this, we may propose two hypothetical models that involve mediation of the effect of `col` on `grad` via its effect on `gre`. The two variants of this model are a pure (`model_0`) and a partial (`model_1`) mediation model.

```
grad_mediation_models_specs <- within(list(),{
  model_0 <- '
      grad ~ hs + gre
      gre ~ hs + col
  '

  model_1 <- '
      grad ~ hs + b_grad_gre*gre + b_grad_col*col
      gre ~ hs + b_gre_col*col

      # labels for indirect, direct, and total
      direct := b_grad_col
      indirect := b_gre_col*b_grad_gre
      total := b_grad_col + (b_gre_col*b_grad_gre)
'
})
```



(a) (b)

Figure 7: A full (a) and partial (b) mediation model of the effect of `col` on `grad` via its effect on `gre`.

These models are depicted in Figure 7. Note that in these diagrams, we have a doubled ended arrow from `col` to `hs`. This indicates that `col` and `hs` are correlated, or more precisely, that `col` and `hs` are assumed to be drawn from a 2d normal distribution with a full covariance matrix, i.e. allowing for non-independence of `col` and `hs`.

```
grad_mediation_models <- map(grad_mediation_models_specs,
                             ~sem(., data = grad_df)
)
```

In terms of AIC, these two models are practically identical.

```
map_dbl(grad_mediation_models, AIC)
#>  model_1  model_0
#> 2749.189 2749.323
```

Likewise, by performing a log likelihood ratio test, we find there is no significant difference between the two models.

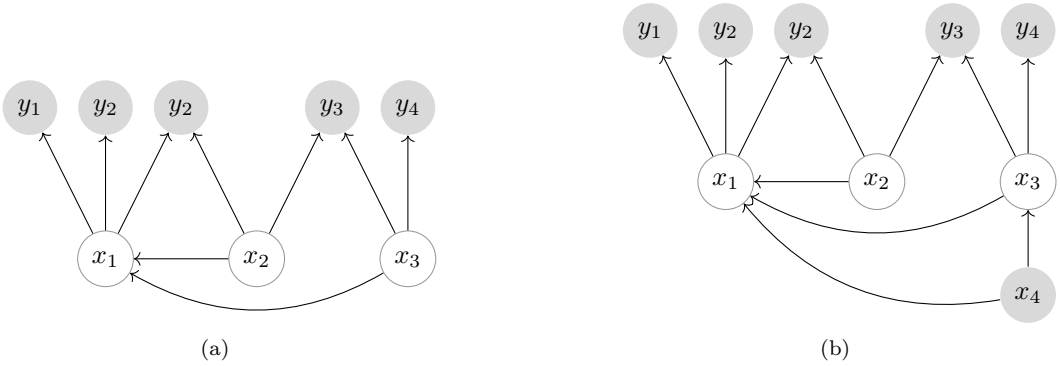
```
grad_mediation_models %$%
  anova(model_0, model_1)
#> Chi-Squared Difference Test
#>
#>         Df    AIC    BIC Chisq Chisq diff Df diff Pr(>Chisq)
#> model_1  0 2749.2 2772.3 0.000
#> model_0  1 2749.3 2769.1 2.134      2.134       1     0.1441
```

From these results, it implies that we can not decide between the full and partial mediation models. However, similarly to the situation in an earlier example, the coefficient from `col` to `grad` in the `model_1` is relatively close to zero and the 95% confidence interval is between -0.042 and 0.288.

```
grad_mediation_models[['model_1']] %>%
  parameterEstimates() %>%
  filter(label == 'direct') %>%
  select(est:ci.upper)
#>     est    se     z pvalue ci.lower ci.upper
#> 1 0.123 0.084 1.465  0.143   -0.042    0.288
```

In conclusion then, the effect of college GPA on graduate school GPA is largely, or completely, mediated by the effect of college GPA on GRE scores.

Structural Equation Modelling



(a) (b)

Figure 8: Two SEM models. In (a), a set of observed variables are functions of a set of latent variables, which are functions of one another. In (b), a set observed variables are functions of both observed and latent variables, which are also functions on one another. We use the convention here of shading the nodes representing observed variables.

The term SEM can be used as an umbrella term for a set of related techniques including factor analysis, path analysis, causal modelling, and even latent variable modelling generally. It is also used in a more narrow sense that is a kind of combination of factor analysis and path analysis models. In this more specific sense of the term, a SEM model consists a set of observed variables that are linear regression functions of latent variables, just as in factor analysis. Then, optionally, the latent variables themselves may be linear regression functions of one another, assuming that these relationships can be described by a directed acyclic graph, just as in a path analysis model. It may also be the case, however, that in addition to the latent variables, we may have further observed variables. For example, we may have observed variables that are predictors, or explanatory variables, of the latent variables. Two simple examples of these SEM model scenarios are depicted in Figure 8.

In general, a SEM model can be defined by a system of regression models some of whose outcome or predictor variables may be latent variables. This definition of a SEM model treat observed variable path analysis models, and confirmatory factor analysis as special cases. In the traditional SEM model, all the regression model are

normal linear ones. This was certainly the case in all the factor analysis and path analysis models we have looked at thus far. While there are no in principle restriction to the regression models being normal and linear, these are still the default case and there are easy to use SEM software packages, including in R, that specifically assume these types of models.

In this section, we will explore SEM using `lavaan` R package. We will explore some relatively simple models, though ones that illustrate the major features of SEM models. For the models we consider, we will use the `PoliticalDemocracy` data set provided by `lavaan`. This data set is regarded as a classic data set for the illustration of SEM models (Bollen 1979, 1989). In it, there are the following variables.

y1 Expert ratings of the freedom of the press in 1960
y2 The freedom of political opposition in 1960
y3 The fairness of elections in 1960
y4 The effectiveness of the elected legislature in 1960
y5 Expert ratings of the freedom of the press in 1965
y6 The freedom of political opposition in 1965
y7 The fairness of elections in 1965
y8 The effectiveness of the elected legislature in 1965
x1 The gross national product (GNP) per capita in 1960
x2 The inanimate energy consumption per capita in 1960
x3 The percentage of the labor force in industry in 1960

The variables beginning with `y_` are all measures of the democracy in a country at two points in time, 1960 and 1965. The variables `y_1`, `y_2`, `y_3` and `y_4` measures democracy variables in 1960. The variables `y_5`, `y_6`, `y_7`, and `y_8` measure the same democracy variables but in 1965. The variables `x_1`, `x_2`, and `x_3` are all measures of the economy in the represented countries in 1960.

A reasonable SEM model is that variables `y_1`, `y_2`, `y_3` and `y_4` are all functions of a single underlying latent variable. This latent variable represents democracy in a country in 1960. Likewise, `y_5`, `y_6`, `y_7`, and `y_8` are all functions of a single latent variable that represents democracy in 1965. Finally, `x_1`, `x_2`, and `x_3` are all measures of a single latent variable representing industrialization in 1960. If we leave the model as such, this leads to a confirmatory factor analysis model. This can be specified using `lavaan` model syntax as follows.

```
sem_model_1_spec <- '
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8
'
```

A note on syntax. We have seen in the previous section that we can define regression and path analysis models easily in `lavaan` using the familiar *R formula* syntax. For example, to specify a regression model such as $y_i \sim \beta_0 + \beta_x x_i + \beta_z z_i + \epsilon_i$, we would write the following.

```
y ~ x + z
```

On the other hand, if we wish to specify a model where, for each i, a set of observed variables y_{1i}, y_{2i}, y_{3i} are functions of a latent variable x_i, we must code this as follows.

```
x =~ y_1 + y_2 + y_3
```

Here, the outcome variables are to the right, rather than tothe left, of the formula operator, which in this case is `=~` rather than `~`.

We may now fit this model as follows.

```
sem_model_1 <- sem(sem_model_1_spec,
                   data = PoliticalDemocracy)
```

As we did in the previous section, we may look at the parameters estimates of this model with the command `parameterEstimates`, and we will filter out the individual variance estimates for simplicity.

```
parameterestimates(sem_model_1) %>%
  filter(op == '=~') %>%
  select(lhs, op, rhs, est, ci.lower, ci.upper)
#>      lhs op rhs   est ci.lower ci.upper
#> 1  ind60 =~  x1 1.000    1.000    1.000
#> 2  ind60 =~  x2 2.182    1.910    2.454
#> 3  ind60 =~  x3 1.819    1.521    2.117
#> 4  dem60 =~  y1 1.000    1.000    1.000
#> 5  dem60 =~  y2 1.354    1.012    1.696
#> 6  dem60 =~  y3 1.044    0.750    1.338
#> 7  dem60 =~  y4 1.300    1.029    1.570
#> 8  dem65 =~  y5 1.000    1.000    1.000
#> 9  dem65 =~  y6 1.258    0.936    1.581
#> 10 dem65 =~  y7 1.282    0.974    1.591
#> 11 dem65 =~  y8 1.310    1.009    1.611
```

Note that one of each of the factor loadings for each latent variable is exactly `1.000000`. This is because, by default, the latent variable variance are not set to be equal to 1 and so it is necessary to constrain the loading matrix values. This is done by setting one arbitrarily chosen value to 1. We may, however, set the variances of the latent variables to 1 as follows.

```
sem_model_1 <- sem(sem_model_1_spec, data = PoliticalDemocracy, std.lv = T)
```

In addition, we may force the latent factors to be orthogonal as follows.

```
sem_model_1 <- sem(sem_model_1_spec,
                   orthogonal = T,
                   std.lv = T,
                   data = PoliticalDemocracy)
```

We now see that the factor loadings are no longer constrained.

```
parameterestimates(sem_model_1) %>%
  filter(op == '=~') %>%
  select(lhs, op, rhs, est, ci.lower, ci.upper)
#>      lhs op rhs   est ci.lower ci.upper
#> 1  ind60 =~  x1 0.667    0.540    0.795
#> 2  ind60 =~  x2 1.464    1.213    1.715
#> 3  ind60 =~  x3 1.217    0.965    1.469
#> 4  dem60 =~  y1 2.133    1.624    2.641
#> 5  dem60 =~  y2 2.993    2.206    3.780
#> 6  dem60 =~  y3 2.322    1.651    2.993
#> 7  dem60 =~  y4 2.922    2.294    3.551
#> 8  dem65 =~  y5 1.907    1.382    2.431
#> 9  dem65 =~  y6 2.703    2.052    3.354
#> 10 dem65 =~  y7 2.627    1.992    3.262
#> 11 dem65 =~  y8 2.891    2.296    3.486
```

We may assess the fit of this model using any of the many fit indices calculated by `sem`. For now, we will just look at those that we have defined above, namely AIC, χ^2_M, and RMSEA.

```
fitmeasures(sem_model_1,
            c("chisq", "df", "pvalue", "aic", "rmsea")
)
#>    chisq       df   pvalue      aic    rmsea
```

```
#>  197.210   44.000    0.000 3298.667    0.215
```

First, we see that the χ^2_M is well above its expected value and hence the corresponding p-value is very low. Hence, we confidently reject the hypothesis that this model provides a perfect fit to the data. The AIC value is 3298.667, which is essentially meaningless in itself, but will be valuable when we compare this model to comparable models later. The RMSEA value is 0.215, which is not low by conventional standards.

Let us now expand this model. The pairs of variables (`y1`, `y5`), (`y2`, `y6`), (`y3`, `y7`), (`y4`, `y8`), given that they each measure the same variable but in different years, ought to be correlated. We can implement this as follows.

```
sem_model_2_spec <- '
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  y1 ~~ y5
  y2 ~~ y6
  y3 ~~ y7
  y4 ~~ y8

'
sem_model_2 <- sem(sem_model_2_spec,
                   orthogonal = T,
                   std.lv = T,
                   data = PoliticalDemocracy)
```

Let us now consider the fit indices of this new model.

```
fitmeasures(sem_model_2,
            c("chisq", "df", "pvalue", "aic", "rmsea")
)
#>    chisq       df   pvalue      aic    rmsea
#>  171.295   40.000    0.000 3280.751    0.209
```

By comparison to `sem_model_1`, these indicate the modelling the residual covariances improves the model's fit, though clearly the fit is still not satisfactory.

As another example of how we can expand this model, we can model the covariances between some of the latent variables. For example, `dem65` and `dem60` are highly likely to be intercorrelated. We can specify this model as follows.

```
sem_model_3_spec <- '
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  y1 ~~ y5
  y2 ~~ y6
  y3 ~~ y7
  y4 ~~ y8

  dem60 ~~ dem65

'
sem_model_3 <- sem(sem_model_3_spec,
                   orthogonal = T,
```

```
                  std.lv = T,
                  data = PoliticalDemocracy)
```

We may again assess the model fit.

```
fitmeasures(sem_model_3,
            c("chisq", "df", "pvalue", "aic", "rmsea")
)
#>    chisq       df   pvalue      aic    rmsea
#>   74.218   39.000    0.001 3185.675    0.110
```

Clearly, this has improved the model fit.

As a final example, let us model `dem65` as a function of `dem60` and `ind60`, `dem60` as a function of `ind60`. We can specify this model as follows.

```
sem_model_3_spec <- '
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + y2 + y3 + y4
  dem65 =~ y5 + y6 + y7 + y8

  dem65 ~ dem60 + ind60
  dem60 ~ ind60

  y1 ~~ y5
  y2 ~~ y6
  y3 ~~ y7
  y4 ~~ y8
'
sem_model_3 <- sem(sem_model_3_spec,
                  orthogonal = T,
                  std.lv = T,
                  data = PoliticalDemocracy)
```

First, let us consider the fit indices of this new model.

```
fitmeasures(sem_model_3,
            c("chisq", "df", "pvalue", "aic", "rmsea")
)
#>    chisq       df   pvalue      aic    rmsea
#>   50.835   37.000    0.064 3166.292    0.071
```

Again, this extension has further improved the fit of the model.

References

Bollen, K. A. 1989. *Structural Equations with Latent Variables*. Wiley.

Bollen, Kenneth A. 1979. "Political Democracy and the Timing of Development." *American Sociological Review*, 572–87.

Horn, John L. 1965. "A Rationale and Test for the Number of Factors in Factor Analysis." *Psychometrika* 30 (2): 179–85.

Joreskog, Karl G, and Marielle Van Thillo. 1972. "LISREL: A General Computer Program for Estimating a Linear Structural Equation System Involving Multiple Indicators of Unmeasured Variables."

Jöreskog, Karl G. 1967. "Some Contributions to Maximum Likelihood Factor Analysis." *Psychometrika* 32 (4): 443–82.

Pearl, Judea. 2012. "The Causal Foundations of Structural Equation Modeling." In *Handbook of Structural Equation Modeling*, edited by Rick H Hoyle. Guilford press.

Spearman, C. 1904. ""General Intelligence," Objectively Determined and Measured." *The American Journal of Psychology* 15 (2): 201–92.

Wright, S. 1921. "Correlation and Causation." *Journal of Agricultural Research* 20 (7): 557–85.

Wright, Sewall. 1934. "The Method of Path Coefficients." *The Annals of Mathematical Statistics* 5 (3): 161–215.