Introduction to Structural Equation Modeling: Linear Regression

Dr. Henrik Kenneth Andersen

2022-10-06

Linear regression in SEM

Let us first look at a simple linear regression model in SEM. This will allow us to examine how SEM works in a relatively simple setting before moving on to more complex models.

We will use an example inspired by Reinecke and Pöge (2010), who looked at the effect of social class and feelings of anomia on xenophobia. They argue that those with a higher social class are less xenophobic, on average, perhaps due to their higher levels of education, which is associated with openness, liberalness, etc. Anomia, on the other hand, describes feelings of isolation and disorientation. Those experiencing anomia might distrust the unfamiliar and blame 'outsiders' for causing their 'ingroup' to be devalued.

However, we will modify the model by replacing anomia with populist sentiment, which could arguably be seen as a symptom of anomia. We do so because the anomia scale in the ALLBUS 2018 dataset is measured dichotomously. Dichotomous variables can easily be included in SEMs, but the theoretical motivation and method is too complex to focus on in this course.[1]

We could write the model as

$$\text{xeno} = \beta_0 + \beta_1 \text{class} + \beta_2 \text{populism} + \epsilon$$

where β_0 is the intercept, i.e., the expected value of xenophobia when the predictors both take on the value of zero. β_1 and β_2 are the regression coefficients that link social class and populism, respectively, to xenophobia. They give the expected change in xenophobia per unit increase in either social class or populism. ϵ is the error term that encompasses all of the other unobserved factors influencing xenophobia.

We will assume for now that the variables of interest $\{\text{xeno}, \text{class}, \text{populism}\}$ are *mean-centered*. In doing so, we can ignore the intercept. Further, we will write the model more abstractly as

$$y = \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

where we assume:

- linearity: the effects of the independent variables are constant and additive
- no autocorrelation: $\mathbb{E}(\epsilon_i \epsilon_j) = 0, \ i \neq j$, i.e., the error of one unit tells us nothing about another unit's errors on average, ensured by random sampling
- no perfect multicollinearity: the independent variables are not linear combinations of each other and are not constants
- zero conditional mean: the covariates are mean independent and thus also uncorrelated with the error, $\mathbb{E}(\epsilon | x_1, x_2) = \mathbb{E}(\epsilon) = 0$
- homoskedasticity: constant variance of the error, $\text{Var}(\epsilon | x_1, x_2) = \text{Var}(\epsilon) = \sigma^2_\epsilon$

[1] Roughly speaking, in an SEM, when we want to treat dichotomous or ordinal variables as dependent variables (e.g., as indicators of latent variables) we assume there are (continuous) normally distributed variables underlying these 'coarse' measures. We use the observed ordinal or dichotomous variable to infer the latent continuous variable and model the effects at the level of the latent variable. The link between the underlying latent variable and the ordinal or dichotomous 'indicator' is either a logic or probit function.

- normally distributed error: the unexplained portion of the dependent variable is normally distributed, $\epsilon \sim N(0, \sigma^2_\epsilon)$

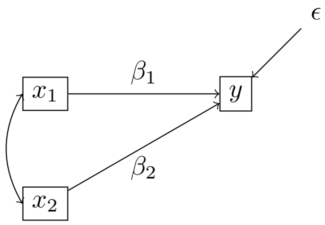
which can be succinctly summarized as

$$\begin{aligned} y|x_1, x_2 &\sim N(\beta_1 x_1 + \beta_2 x_2, \sigma^2_\epsilon), \\ \epsilon &\sim N(0, \sigma^2_\epsilon). \end{aligned}$$

Path diagrams

In SEM, we like to use *path diagrams* to represent our models. It may seem unnecessary here, because everyone knows what a multivariate regression model is and 'looks like', but it will help us to get into the habit of representing our models both as equations and path diagrams.

Figure 1: Path model, multivariate regression model



We use squares or rectangles to represent observed or 'manifest' variables and circles to represent unobserved or 'latent' variables (which we will introduce later). One-headed arrows represent an effect of one variable on another variable, and two-headed arrows represent covariances or correlations. The error, here ϵ is usually represented unenclosed by any circle or square. The implicit regression coefficient for the error is 1.0. Table 1 gives an overview of the path diagram notation.

Note that the two independent variables, x_1 and x_2, are shown to *covary.* This is because we of course want the partial effects of class on xenophobia, holding populism constant, and vice versa. If we say $y = \beta_1 x_1 + \beta_2 x_2 + \epsilon$, then by covariance algebra, we see that the effect of, say, β_1 is given by

$$\begin{aligned} \text{Cov}(y, x_1) &= \text{Cov}(\beta_1 x_1 + \beta_2 x_2 + \epsilon, x_1) \\ &= \beta_1 \text{Var}(x_1) + \beta_2 \text{Cov}(x_1, x_2) \\ \beta_1 &= \frac{\text{Cov}(y, x_1) - \beta_2 \text{Cov}(x_1, x_2)}{\text{Var}(x_1)} \end{aligned}$$

since $\text{Cov}(x_1, \epsilon) = 0$, by assumption. Obviously, the estimate for β_1 depends on whether $\text{Cov}(x_1, x_2)$ equals zero, or not. By allowing the independent variables to correlate, we get the partial effect of x_1 on y. If we were to assume the covariance was zero, we would have just $(\text{Cov}(y, x_1) - \beta_2 \cdot 0)/\text{Var}(x_1)$, which is the marginal effect.

There is some debate as to whether to display the covariance/correlation between model covariates, however, because in a regular regression model in SEM, the variances and covariances of the observed independent variables are set to the sample estimates. They are not 'estimated' like the error variance for ϵ is, but more on that later.

The logic of SEM

To discuss the logic of SEM, let us look at an even simpler model (we will come back to the xenophobia example shortly):

$$y = \beta_1 x + \epsilon$$

Table 1: Path diagram symbols and notation

Symbol	Explanation
$\boxed{x}$	Observed or "manifest" variable
$\textcircled{\eta}$	Unobserved or "latent" variable
$\longrightarrow$	Effect; regression or factor loading
$\longleftrightarrow$	Covariance or correlation

where both x and y have been mean-centered.[2] We could estimate this model by OLS, where from the so-called first order conditions,

$$\begin{aligned} \mathbb{E}(x\epsilon) &= \mathbb{E}(x(y - \beta_1 x)) = 0 \\ \mathbb{E}(\epsilon) &= \mathbb{E}(y - \beta_1 x) = 0 \end{aligned}$$

and for a given sample, we would could use the method of moments approach and solve to see

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n} x_i(y_i - \beta_1 x_i) &= 0 \\ \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \beta_1 \frac{1}{n}\sum_{i=1}^{n} x_i^2 &= 0 \\ \beta_1 \frac{1}{n}\sum_{i=1}^{n} x_i^2 &= \frac{1}{n}\sum_{i=1}^{n} x_i y_i \\ \hat{\beta}_1 &= \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n} x_i^2}. \end{aligned}$$

The logic of SEM is more complicated than this, and it will seem unnecessarily so for such a simple model. But we will see that the framework allows for a great deal of flexibility, making it possible to estimate complicated models with latent variables and numerous dependent variables.

In our simple model, we have two observed variables, x and y. We assume there is a *population covariance matrix*, which we call $\mathbf{\Sigma}$ (Sigma):

$$\mathbf{\Sigma} = \begin{bmatrix} \text{Var}(y) & \text{Cov}(y, x) \\ \text{Cov}(x, y) & \text{Var}(y) \end{bmatrix}.$$

We assume that our model, $y = \beta_1 x + \epsilon$ accurately describes the population data generating process. We translate our model into the so-called *model-implied covariance matrix*, which is a function of the unknown parameters, $\boldsymbol{\theta} = (\beta_1, \phi, \psi)$ where $\phi = \sigma_x^2$ is the variance of the independent variable x, and $\psi = \sigma_\epsilon^2$ is the unexplained variance of y.

[2] This means that $\frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i) = 0 \implies \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} = 0$, since by mean-centering, $\bar{y} = \bar{x} = 0$.

Based on our model, using covariance algebra, we have

$$\begin{aligned}
\operatorname{Var}(y) &= \operatorname{Var}(\beta_1 x + \epsilon) \\
&= \beta_1^2 \operatorname{Var}(x) + \operatorname{Var}(\epsilon) \\
&= \beta_1^2 \phi + \psi \\
\operatorname{Cov}(x, y) &= \operatorname{Cov}(x, \beta_1 x + \epsilon) \\
&= \beta_1 \operatorname{Cov}(x, x) + \operatorname{Cov}(x, \epsilon) \\
&= \beta_1 \phi \\
\operatorname{Var}(x) &= \operatorname{Var}(x) \\
&= \phi
\end{aligned}$$

since $\mathbb{E}(x\epsilon) = \operatorname{Cov}(x, \epsilon) = 0$.

Short excurse on covariance algebra

Covariance and variance are defined as (Bollen 1989, 21):

$$\begin{aligned}
\operatorname{Cov}(x_1, x_2) &= \mathbb{E}[(x_1 - \mathbb{E}(x_1))(x_2 - \mathbb{E}(x_2))] \\
&= \mathbb{E}(x_1 x_2) - \mathbb{E}(x_1)\,\mathbb{E}(x_2), \\
\operatorname{Var}(x_1) &= \operatorname{Cov}(x_1, x_1) \\
&= \mathbb{E}[(x_1 - \mathbb{E}(x_1))^2] \\
&= \mathbb{E}(x_1^2) - \mathbb{E}(x_1)^2.
\end{aligned}$$

The sample analogs replace the expectation with the mean, so if we call the sample covariance and variance $\operatorname{cov}(\cdot)$ and $\operatorname{var}(\cdot)$, and the sample means $\bar{x}_1, \bar{x}_2$, etc., then

$$\begin{aligned}
\operatorname{cov}(x_1, x_2) &= \frac{1}{n-1} \sum_{i=1}^{n} (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2), \\
\operatorname{var}(x_1) &= \frac{1}{n-1} \sum_{i=1}^{n} (x_{1i} - \bar{x}_{1i})^2
\end{aligned}$$

For mean-centered variables, these reduce to

$$\begin{aligned}
\operatorname{Cov}(z_1, z_2) &= \mathbb{E}(z_1 z_2) \\
\operatorname{cov}(z_1, z_2) &= \frac{1}{n-1} \sum_{i=1}^{n} z_{1i} z_{2i}, \\
\operatorname{Var}(z_1) &= \mathbb{E}(z_1^2) \\
\operatorname{var}(z_1) &= \frac{1}{n-1} \sum_{i=1}^{n} z_{1i}^2
\end{aligned}$$

If c is a constant, then

$$\begin{aligned}
\operatorname{Cov}(c, x_1) &= 0 \\
\operatorname{Cov}(c x_1, x_2) &= c \operatorname{Cov}(x_1, x_2) \\
\operatorname{Var}(c x_1) &= c^2 \operatorname{Var}(x_1) \\
\operatorname{Cov}(x_1 + x_2, x_3) &= \operatorname{Cov}(x_1, x_3) + \operatorname{Cov}(x_2, x_3).
\end{aligned}$$

We put these expressions into the model-implied coariance matrix, which we call $\mathbf{\Sigma}(\boldsymbol{\theta})$,

$$\mathbf{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \beta_1^2\phi + \psi & \beta_1\phi \\ \beta_1\phi & \phi \end{bmatrix}$$

and we set the population and the model-implied covariance matrix to be equal, since we assume our model accurately describes the population data generating process:

$$\begin{gathered} \mathbf{\Sigma} = \mathbf{\Sigma}(\boldsymbol{\theta}) \\ \begin{bmatrix} \mathrm{Var}(y) & \mathrm{Cov}(y,x) \\ \mathrm{Cov}(x,y) & \mathrm{Var}(y) \end{bmatrix} = \begin{bmatrix} \beta_1^2\phi + \psi & \beta_1\phi \\ \beta_1\phi & \phi \end{bmatrix} \end{gathered}$$

Note, however, that we do not observe the population covariance matrix, but rather just a sample of it, so we switch it out for the sample covariance matrix, $\boldsymbol{S}$

$$\begin{gathered} \boldsymbol{S} = \mathbf{\Sigma}(\boldsymbol{\theta}) \\ \begin{bmatrix} \mathrm{var}(y) & \mathrm{cov}(y,x) \\ \mathrm{cov}(x,y) & \mathrm{var}(y) \end{bmatrix} = \begin{bmatrix} \beta_1^2\phi + \psi & \beta_1\phi \\ \beta_1\phi & \phi \end{bmatrix} \end{gathered}$$

where we can use var($\cdot$) and cov($\cdot$) (lower-case) to represent the sample analogs of the population parameters, Var($\cdot$), Cov($\cdot$).

Now, estimation means applying some rule to select the the values for β_1, ϕ and ψ. We call these the *estimates*, denoted $\hat{\beta}_1$, $\hat{\phi}$ and $\hat{\psi}$ and replace $\boldsymbol{\theta}$ with $\hat{\boldsymbol{\theta}}$:

$$\begin{gathered} \boldsymbol{S} = \mathbf{\Sigma}(\hat{\boldsymbol{\theta}}) \\ \begin{bmatrix} \mathrm{var}(y) & \mathrm{cov}(y,x) \\ \mathrm{cov}(x,y) & \mathrm{var}(y) \end{bmatrix} = \begin{bmatrix} \hat{\beta}_1^2\hat{\phi} + \hat{\psi} & \hat{\beta}_1\hat{\phi} \\ \hat{\beta}_1\hat{\phi} & \hat{\phi} \end{bmatrix} \end{gathered}$$

From here, we define the *residual matrix*, which we could call $\boldsymbol{R} = \boldsymbol{S} - \mathbf{\Sigma}(\hat{\boldsymbol{\theta}})$

$$\begin{aligned} \boldsymbol{R} &= \boldsymbol{S} - \mathbf{\Sigma}(\hat{\boldsymbol{\theta}}) \\ &= \begin{bmatrix} \mathrm{var}(y) & \mathrm{cov}(y,x) \\ \mathrm{cov}(x,y) & \mathrm{var}(y) \end{bmatrix} - \begin{bmatrix} \hat{\beta}_1^2\hat{\phi} + \hat{\psi} & \hat{\beta}_1\hat{\phi} \\ \hat{\beta}_1\hat{\phi} & \hat{\phi} \end{bmatrix} \end{aligned}$$

and choose the unknown values as those that *minimize the discrepancy* between the observed and model-implied matrices.

Estimation

There are a number of *estimators* available for SEM analyses. They can be seen as different approaches to achieving the goal of minimizing the discrepancy of the observed and model-implied matrices.

We say that for each estimator, there is a *fitting function*, F, that we would like to optimize (usually minimize). For example, the Unweighted Least Squares (ULS) estimator with its fitting function F_{ULS} works on the logic of minimizing one-half the squared differences between the observed and model-implied matrices:

$$F_{ULS} = \frac{1}{2}\,\mathrm{tr}[(\boldsymbol{S} - \mathbf{\Sigma}(\boldsymbol{\theta}))^2].$$

F_{ULS} is an intuitive estimator because of the *obvious parallels to ordinary least squares*-based regression (only here we are minimizing the squared residuals defined as the elements of the residual matrix rather than the individual deviations from the predicted value). It also *does not require the normality assumption* on the observed variables for consistent estimates of $\boldsymbol{\theta}$.

The so-called *Generalized Least Squares* (GLS) estimator weights the elements of $\boldsymbol{S} - \boldsymbol{\Sigma}(\boldsymbol{\theta})$ according to their variances and covariances to correct for heteroskedasticity or autocorrelation. Its fitting function is given by

$$F_{GLS} = \frac{1}{2}\,\mathrm{tr}[([\boldsymbol{S} - \boldsymbol{\Sigma}(\boldsymbol{\theta})]\boldsymbol{W}^{-1})^2]$$

where $\boldsymbol{W}^{-1}$ is usually set to $\boldsymbol{S}^{-1}$, i.e., we are essentially dividing the elements of the residual matrix by the sample (co)variances before performing least squares. GLS reduces to ULS when $\boldsymbol{W}^{-1} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix with ones down the diagonal and zeros everywhere else.

However, *Maximum Likelihood* (ML) is the most widely used estimator. Its fitting function is

$$F_{ML} = \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \mathrm{tr}[\boldsymbol{S}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta})] - \log|\boldsymbol{S}| - (p+q)$$

where log is the natural logarithm, $|\cdot|$ is the determinant, $\mathrm{tr}(\cdot)$ is the trace and p is the number of dependent and q is the number of independent observed variables.[3]

Once we have chosen the estimator, we plug in the sample and model-implied matrices and take the first derivative with respect to each of the unknowns in $\boldsymbol{\theta}$. We set the derivative to zero and solve for the unknowns. Sometimes, like in the case of simple linear regression models, there are explicit solutions to the parameters that can be worked out algebraically.

A numeric example

Let us assume we have the simple model from above, $y = \beta_1 x + \epsilon$ and we observe the following covariance matrix:

$$\boldsymbol{S} = \begin{bmatrix} 11.88 & 6.13 \\ 6.13 & 4.05 \end{bmatrix}$$

where $\mathrm{var}(y) = 11.88$, $\mathrm{cov}(y, x) = \mathrm{cov}(x, y) = 6.13$ and $\mathrm{var}(x) = 4.05$.

The unknown parameters are, again, $\boldsymbol{\theta} = (\beta_1, \phi, \psi)$, i.e., the regression coefficient, the variance of x and the error variance. Normally, the variance of the observed independent variables, here x, is not estimated but rather set to the sample value. So, we would say $\hat{\phi} = 4.05$.

We find the respective estimates (ULS, GLS, ML, etc.) by taking the first partial derivative of the fitting functions with respect to each of the unknown parameters, setting the results to zero and solving. Taking the partial derivative means we are finding a function for the slope of the tangent line (or tangent plane in the multivariate case), the line that touches the function at just one point. When we set the slope of the tangent line to zero, we are looking for a horizontal line which indicates either a minimum or a maximum (because if the slope was not zero, it would imply the function was still increasing or decreasing).

The following code generates an example function and shows that the slope of the tangent line is given by the first derivative of the function. Setting the slope of the tangent line to zero tells us which unknown value produces an optimum.

```r
library(ggplot2)

# Some arbitrary function of x
exfunc <- function(x) {
  y <- 15 + 2 * x - 0.75 * x^2
```

[3] How the fitting functions are derived is not especially interesting nor is it important for this course. The ML fitting function comes from the multivariate normal distribution and the assumption of independent and identically distributed (i.i.d.) observations, see Bollen (1989), p. 107–111 and p. 131–134.

```
  return(y)
}

# First derivative w.r.t. x is 2 - 1.5 * x
D(expression(15 + 2 * x - 0.75 * x^2 ), "x")
```

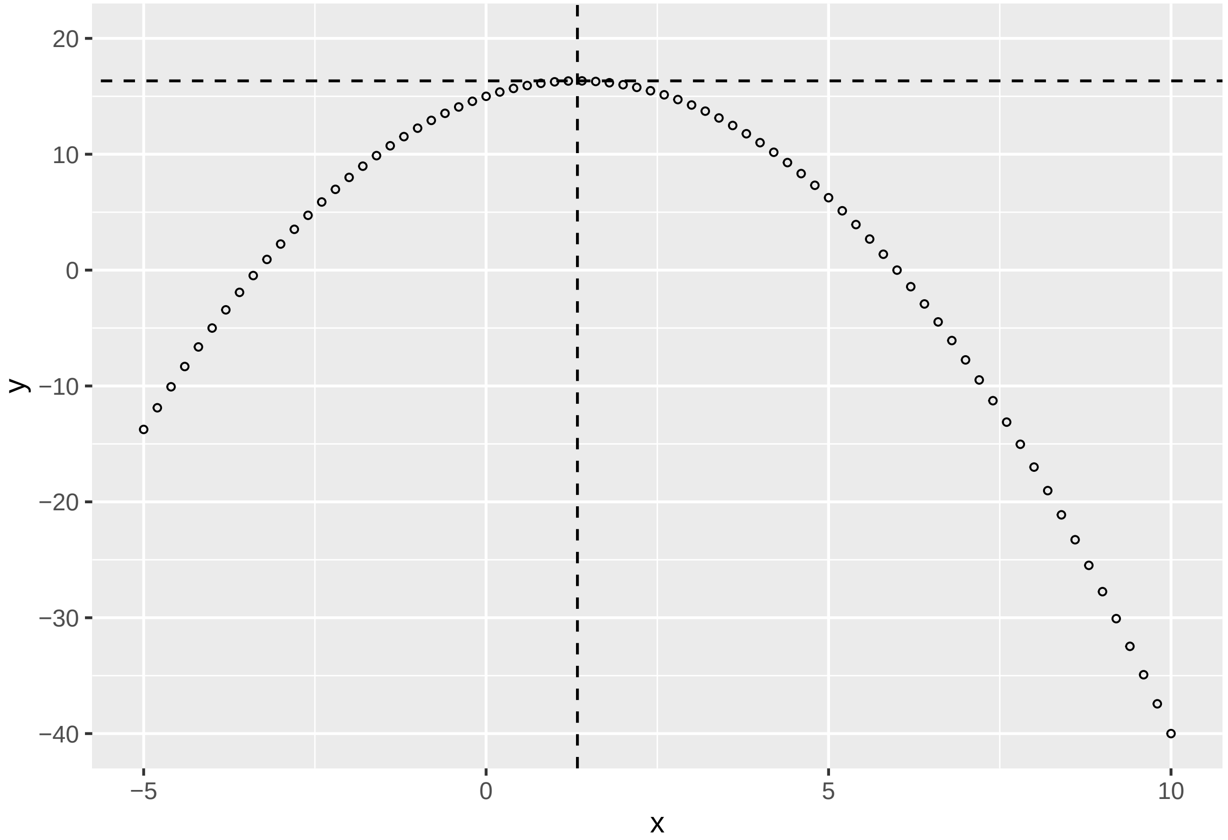
```
## 2 - 0.75 * (2 * x)
```

```
# Solve for x
# 2 - 1.5x = 0
# 1.5x     = 2
# x        = 2 / 1.5
# x        = 1.33
# x = 1.33 maximizes the function

# Make some data consistent with function, inputs chosen mostly arbitrarily
x <- seq(from = -5, to = 10, by = 0.2)
# Outputs
y <- 15 + 2 * x - 0.75 * x^2

# Put the inputs and outputs in a temp dataframe
dftemp <- data.frame(x, y)

# Plot the function, show the tangent line with slope of zero
ggplot(dftemp, aes(x = x, y = y)) +
  geom_point(shape = 1, size = 1) +
  geom_abline(intercept = exfunc(2 / 1.5), slope = 0, linetype = "dashed") +
  geom_vline(xintercept = 2 / 1.5, linetype = "dashed") +
  scale_y_continuous(limits = c(-40, 20), breaks = seq(-40, 20, 10))
```



For ULS, we have

$$
\begin{aligned}
F_{ULS} &= \frac{1}{2}\,\mathrm{tr}\left[\left(\begin{bmatrix}11.88 & 6.13\\ 6.13 & 4.05\end{bmatrix} - \begin{bmatrix}\beta_1^2\phi+\psi & \beta_1\phi\\ \beta_1\phi & \phi\end{bmatrix}\right)^2\right]\\
&= \frac{1}{2}\,\mathrm{tr}\left[\left(\begin{bmatrix}11.88-\beta_1^2 4.05-\psi & 6.13-\beta_1 4.05\\ 6.13-\beta_1 4.05 & 4.05-4.05\end{bmatrix}\right)^2\right]
\end{aligned}
$$

since we are setting the model-implied variance of x to the sample value, $\hat{\phi} = 4.05$. From here, we can take the first partial derivatives with respect to ψ and β and solve for each to get the ULS estimates. We could do this by hand, but it is much easier to use a symbolic algebra program, like `sympy` in `Python`:

```
# Use symbolic algebra library sympy
from sympy import *
# import sympy

# Symbols
psi, beta = symbols('psi, beta')

# Observed sample covariance matrix S
S = Matrix([[11.88, 6.13],
            [6.13,  4.05]])

# Model-implied covariance matrix Sigma(theta)
Sig = Matrix([[beta ** 2 * 4.05 + psi, beta * 4.05],
              [beta * 4.05,            4.05]])
```

```
# ULS fitting function
Fuls = 1 / 2 * trace((S - Sig) ** 2)

# First partial derivative for beta
dFulsdbeta = diff(Fuls, beta)
print('$' + latex(dFulsdbeta) + '$')
```

$$32.805\beta^3 + 8.1\beta\psi - 63 - 423\beta - 49.653$$

```
# First partial derivative for psi
dFulsdpsi = diff(Fuls, psi)
print('$' + latex(dFulsdpsi) + '$')
```

$$4.05\beta^2 + 1.0\psi - 11.88$$

```
# Solve for psi
print('$' + latex(solve(dFulsdpsi, psi)) + '$')
```

$$[11.88 - 4.05\beta^2]$$

```
# Substitute solution for psi into derivative for beta
dFulsdbeta = dFulsdbeta.subs(Symbol('psi'), 11.88 - beta ** 2 * 4.05)

# Solve for beta
print('$' + latex(solve(dFulsdbeta, beta)) + '$')
```

$$[1.51358024691358]$$

```
# Solve for psi = 11.88 - 4.05 * beta ** 2
11.88 - 4.05 * 1.51 ** 2
```

$$2.645595000000002$$

From this, we see that $\hat{\boldsymbol{\theta}}_{ULS} = (\hat{\beta}_{ULS} = 1.51, \hat{\psi}_{ULS} = 2.65)$.

The ML approach is actually rather easy to work out by hand. We have

$$F_{ML} = \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \text{tr}[\boldsymbol{S}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta})] - \log|\boldsymbol{S}| - (p+q)$$

where we input the matrices from above,

$$\begin{aligned} F_{ML} &= \log\left|\begin{bmatrix} \beta^2\phi + \psi & \beta\phi \\ \beta\phi & \phi \end{bmatrix}\right| + \text{tr}\left[\begin{bmatrix} 11.88 & 6.13 \\ 6.13 & 4.05 \end{bmatrix}\begin{bmatrix} \beta^2\phi + \psi & \beta\phi \\ \beta\phi & \phi \end{bmatrix}^{-1}\right] - \log\left|\begin{bmatrix} 11.88 & 6.13 \\ 6.13 & 4.05 \end{bmatrix}\right| - (2) \\ &= \log(\phi\psi) + \frac{11.88}{\psi} + \frac{4.05\beta^2}{\psi} + \frac{4.05}{\phi} - 2\frac{6.13\beta}{\psi} - 10.54 - 2 \end{aligned}$$

and take the partial derivatives with respect to each and solve. For β, we have

$$\begin{aligned}\frac{\partial F_{ML}}{\partial \beta} &= \frac{8.1\beta}{\psi} - \frac{12.26}{\psi} \\ 0 &= \frac{8.1\beta}{\psi} - \frac{12.26}{\psi} \\ \hat{\beta}_{ML} &= \frac{12.26}{8.1} = 1.51\end{aligned}$$

and for ψ, we have

$$\begin{aligned}\frac{\partial F_{ML}}{\partial \psi} &= \frac{12.26\beta}{\psi^2} + \frac{1}{\psi} - \frac{4.05\beta^2}{\psi^2} - \frac{11.88}{\psi^2} \\ 0 &= \frac{12.26(1.51)}{\psi^2}\psi^2 + \frac{1}{\psi}\psi^2 - \frac{4.05(1.51)^2}{\psi^2}\psi^2 - \frac{11.88}{\psi^2}\psi^2 \\ \hat{\psi}_{ML} &= 2.65\end{aligned}$$

so we could write $\hat{\boldsymbol{\theta}}_{ML} = (\hat{\beta}_{ML} = 1.51, \hat{\psi}_{ML} = 2.65)$, which, in this case, are the same as the ULS estimates.

Numerical methods

For more complex models, however, there may not be explicit solutions. In those cases, SEM uses so-called *'numerical methods'* to find the minimum of the functions. These are iterative methods, i.e., we take some 'starting values' for the unknown parameters, call those $\boldsymbol{\theta}^{(1)}$, and then change them to some other values, $\boldsymbol{\theta}^{(2)}$, and then re-evaluate the fitting function. We do this until a minimum is reached; essentially the point after which the fitting function begins to no longer improve but become worse. Bollen (1989), p. 136–144 and Ferron and Hess (2007) discuss numerical solutions in more detail, but we can briefly look at it here.

Take the example used in Bollen (1989), p. 138 f.: say we had the model $y = x + \epsilon$ where $\phi = 1$, $\text{Cov}(x, \epsilon) = 0$ and $\mathbb{E}(\epsilon) = 0$. Essentially, we are fixing the regression coefficient to 1.0 and letting the variance of the independent variable be set to the sample value, which in this example is also 1.

The observed covariance matrix is

$$\boldsymbol{S} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

and the model-implied covariance matrix is

$$\begin{bmatrix} 1+\psi & 1 \\ 1 & 1 \end{bmatrix}$$

When we plug these into F_{ML}, and after some simplification, we get

$$F_{ML} = \log(\hat{\psi}) + \hat{\psi}^{-1} - 1.$$

The goal is to find a value for $\hat{\psi}$ that minimizes the fitting function. In this case, it is easy to see that plugging in 1 gives us exactly the observed covariance matrix, but say we wanted to use a numerical method instead.

Note that F_{ML} is a *function*, we put in an input, $\hat{\psi}^{(i)}$, and we get an output, $F_{ML}^{(i)}$, as shown in Figure 2.

```r
# Make the Fml fitting function
Fml <- function(psi) {
  Fmli <- log(psi) + psi^-1 - 1
  return(Fmli)
}

# Values for psi from 0.1 to 10
psi <- seq(from = 0.1, to = 10, by = 0.01)

# Calculate Fml for each value of psi
Fmli <- Fml(psi)

# Put both into a dataframe for ggplot
dftemp <- data.frame(Fmli, psi)

# Plot the fitting function
library(ggplot2)
ggplot(dftemp, aes(x = psi, y = Fmli)) +
  geom_point(shape = 1) +
  scale_x_continuous(limits = c(0, 10), breaks = seq(from = 0, to = 10, by = 0.5)) +
  scale_y_continuous(name = "Fml", limits = c(-1, 7), breaks = -1:7)
```

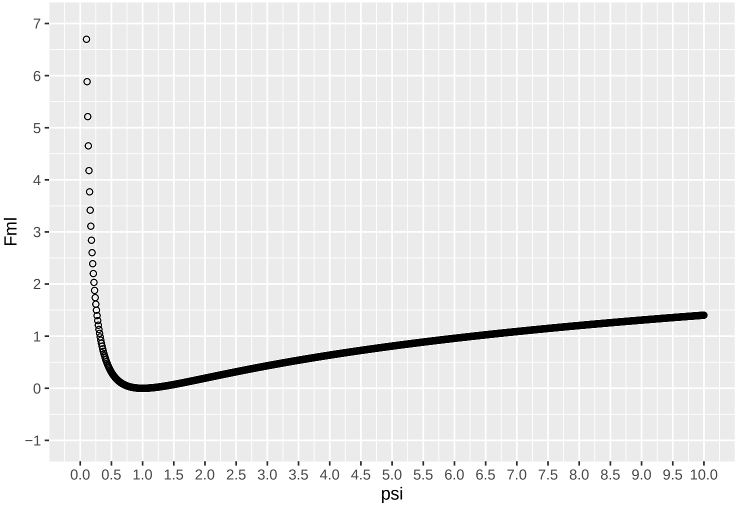


Figure 2: Fitting function

All this shows is that for every possible value of ψ we can calculate the value of the fitting function, and we would take the value for $\hat{\psi}$ that minimizes the function. This is $\hat{\psi} = 1$, which is what we knew already from just looking at the covariance matrices.

But the interesting thing about numerical methods is that we can basically start from any value, $\hat{\psi}^{(1)}$, note the fitting function, then plug in a second value, $\hat{\psi}^{(2)}$, again note the value of the fitting function and compare. We would then continue to do this until the fitting function was no longer improving but getting worse.

This raises at least three questions, however:

1. what values to start with?
2. how to know which direction to begin searching in?
3. how much to increase/decrease the next value to plug in?

As for the first question, if we are estimating a complex model, we could start by breaking the model up into parts and estimating simple linear regressions. Then we could use those values as the starting values. Or, we could turn to the literature for plausible starting values. Often, the choice of starting values will not make or break the model, however.

Concerning the second question, notice that the first derivative of the fitting function gives us the *tangent line*, whose slope, or *gradient* tells us whether the function is increasing or decreasing at a given point. The first derivative of the function above is $\frac{1}{\psi} - \psi^{-2}$, which we can verify using R:

```
# Write Fml as an expression and use D() to get the derivative
D(expression(log(psi) + psi^-1 - 1), "psi")
```

```
## 1/psi - psi^-(1 + 1)
```

So, at, say, $\psi = 0.5$, the slope of the tangent line is $\frac{1}{0.5} - 0.5^{-2} = -2$. We can add this to the plot[4]

```
# Add the gradient at psi = 0.5 to the plot
ggplot(dftemp, aes(x = psi, y = Fmli)) +
  geom_point(shape = 1, size = 0.5) +
  scale_x_continuous(limits = c(0, 2), breaks = seq(from = 0, to = 2, by = 0.1)) +
  scale_y_continuous(name = "Fml", limits = c(-1, 7), breaks = -1:7) +
  geom_abline(intercept = 1.31, slope = -2)
```

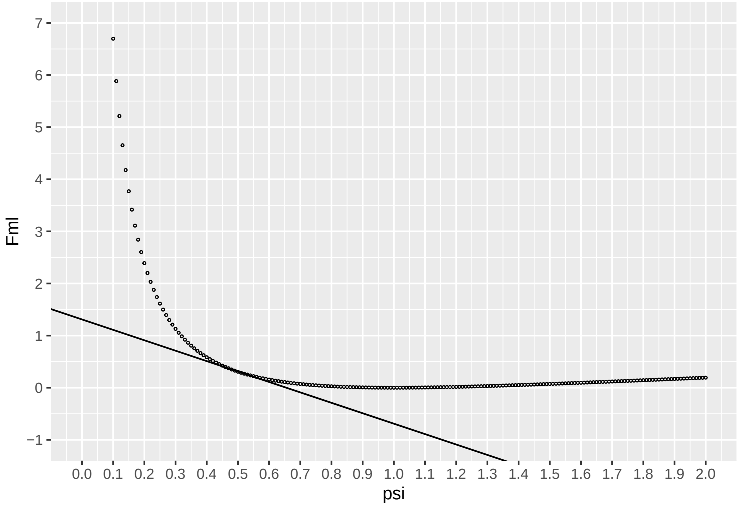


Figure 3: Fitting function (zoomed in) with gradient at $\psi = 0.5$

Our goal is to minimize F_{ML}, so if the gradient at the starting value is negative, like it is in this case, then it tells us we need to *increase* the value of $\hat{\psi}$. This will bring us closer to the minimum. If we had a positive gradient for our starting value and we were trying to minimize the function, it would tell us we need to go back by decreasing $\hat{\psi}$, because the minimum (or a minimum) is behind us, so to speak. The same goes in reverse for maximizing a function.

As for the third question, how much to increase the value we plug in from iteration to iteration, note that the slope of the gradient tells us how far away we are from the minimum. For a value of $\hat{\psi}$ that is far away from the minimum, the slope is very steep, see for example the red gradient in Figure 4. The slope gets less steep the closer we get to the minimum, e.g., the blue gradient. The green gradient is not very steep either, but it is positive, so we know that we have gone too far.

```
# Model the function again over range of psi values
psi_vals <- seq(from = 0.1, to = 10, by = 0.1)
fml_vals <- Fml(psi_vals)

# Put the inputs and outputs into a dataframe
```

[4] For those wondering, we know the line passes through $\psi = 0.5$, $F_{ML} = \log(\psi) + \psi^{-1} - 1 = 0.31$ and the slope is $\partial F_{ML}/\partial\psi = 1/0.5 - 0.5^{-2} = -2$, so the intercept follows the pattern of $-\beta x^1 + y^1 = -(-2)0.5 + 0.31 = 1.31$.

```r
dftemp1 <- data.frame(psi_vals, fml_vals)

# Now select just three values for psi to plot the gradients
psi_vals_select <- c(0.4, 0.9, 1.2)
fml_vals_select <- Fml(psi_vals_select)

# Get the slopes based on derivative, 1 / psi - psi^-2
slope_vals_select <- 1 / psi_vals_select - psi_vals_select^-2
# Work out the intercepts
int_vals_select <- -slope_vals_select * psi_vals_select + fml_vals_select

# Put these slopes and intercepts into second temp df
dftemp2 <- data.frame(slope_vals_select, int_vals_select)

# Plot the function with the three gradients
ggplot(dftemp1, aes(x = psi_vals, y = fml_vals)) +
  geom_point(size = 1, shape = 1) +
  geom_line() +
  scale_y_continuous(name = "Fml", limits = c(-1, 7), breaks = seq(-1, 7, 1)) +
  scale_x_continuous(name = "phi", limits = c(0, 10), breaks = 0:10) +
  geom_abline(dftemp2, slope = slope_vals_select, intercept = int_vals_select,
              size = 1, color = c("red", "blue", "green"))
```

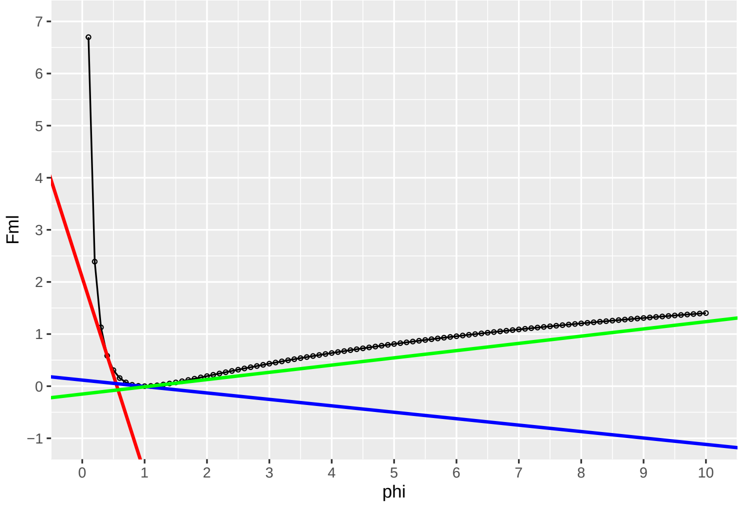


Figure 4: Fitting function with multiple gradients

The widely used *Newton-Raphson* algorithm for choosing the next value $\hat{\psi}^{(i+1)}$ works by dividing the gradient of the current value, $\hat{\psi}^{(i)}$, by the second partial derivative, the *Hessian* and then subtracting that from the current value. More generally, when $\boldsymbol{\theta}$ contains more than one unknown parameter, we choose $\boldsymbol{\theta}^{(i+1)}$ by premultiplying the vector of gradients by the inverse of the Hessian matrix

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} - \left[\frac{\partial^2 F_{ML}}{\partial\hat{\boldsymbol{\theta}}\partial\hat{\boldsymbol{\theta}}^{\intercal}}\right]^{-1}\left[\frac{\partial F_{ML}}{\partial\hat{\boldsymbol{\theta}}}\right]$$

In our case, we have the function itself,

$$F_{ML} = \log(\hat{\psi}) + \hat{\psi}^{-1} - 1$$

we have the first derivative

$$\frac{\partial F_{ML}}{\partial \psi} = \psi^{-1}(1 - \psi^{-1})$$

and the second partial derivative

$$\frac{\partial^2 F_{ML}}{\partial \psi^2} = \frac{2 - \psi}{\psi^3}$$

and so when we put these together, we get

$$\hat{\psi}^{(i+1)} = \hat{\psi}^{(i)} - \left[\frac{2 - \hat{\psi}^{(i)}}{(\hat{\psi}^{(i)})^3}\right]^{-1} [(\hat{\psi}^{(i)})^{-1}(1 - (\hat{\psi}^{(i)})^{-1})]$$

this looks ugly, but it is not that complicated. We just plug in our starting value and evaluate. Then use that result as the new starting value, so to speak, and continue until we no longer see any improvement.

```r
# Newton-Raphson fuction
nr_func <- function(psi) {
  psi_plus1 <- psi - ((2 - psi) / (psi^3))^-1 * (psi^-1 * (1 - psi^-1))
  return(psi_plus1)
}
```

Say we started at 0.5:

```r
nr_func(0.5)
```

```
## [1] 0.6666667
```

after the first step, our new value would be 0.667. We plug this into the function

```r
nr_func(0.667)
```

```
## [1] 0.8336249
```

We would continue this until we saw no further improvement:

```r
nr_func(0.834)
```

```
## [1] 0.9527341
```

```r
nr_func(0.953)
```

```
## [1] 0.9957803
```

```r
nr_func(0.996)
```

```
## [1] 0.9999681
```

which, we can see *converges* to the value we know to be correct based on the algebraic solution, i.e., $\hat{\psi} = 1$. SEM software set the break-off point in terms of the change in subsequent estimates. Moving from the first to the second iteration, the change was $0.834 - 0.667 = 0.167$, from the second to the third it was $0.953 - 0.834 = 0.119$, from the third to the fourth just $0.996 - 0.953 = 0.043$ and from the fourth to the fifth $1 - 0.996 = 0.004$. In the move from the second last to the last iteration shown here, we see basically no further improvement and we would stop the search.

The Newton-Raphson algorithm may seem abstract. See the Notes on Numerical Methods for more details.

Maximum likelihood

Maximum likelihood is by far the most popular estimator used in SEM. One criticism you might have heard of ML is that it requires that the model variables are normally distributed. In this section, I will describe on a basic level why the assumption of normality is so important for SEM (compared to, say, OLS which also tends to assume normality but with a different motivation), what it pertains to and how one can treat the assumption in practice.

When a continuous random variable is normally distributed, the probability of observing, say, y_i is

$$\mathcal{P}(y_i) \sim \mathcal{N}(\mu, \sigma^2), \text{ or}$$
$$\mathcal{P}(y_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{y_i - \mu}{\sigma}\right)^2\right].$$

In other words, the distribution can be fully characterized by its mean, μ, and variance, σ^2. As such, for every possible value, we can calculate the exact probability of y_i.

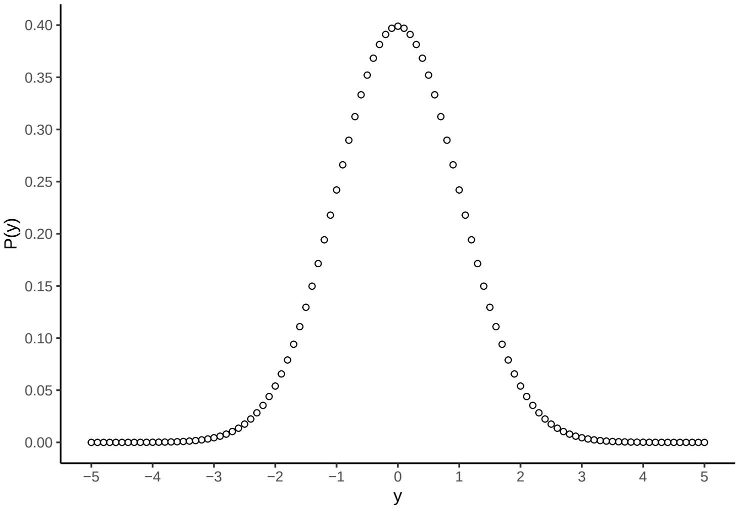


Figure 5: Probability distribution $y \sim \mathcal{N}(\mu = 0, \sigma = 1)$

For example, assuming the standard normal distribution, which is a special case of the normal distribution where $\mu = 0$ and $\sigma^2 = 1$, shown in Figure 5, the probability of observing the value 0.75 is

$$\begin{aligned}\mathcal{P}(y_i; \mu, \sigma^2) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{y_i - \mu}{\sigma}\right)^2\right]\\ \mathcal{P}(0.75; 0, 1) &= \frac{1}{\sqrt{2\pi(1)}} \exp\left[-\frac{1}{2}\left(\frac{0.75 - 0}{1}\right)^2\right]\\ &= 0.3.\end{aligned}$$

This is highlighted in Figure 6 with the vertical line showing $y_i = 0.75$ and the horizontal line showing $\mathcal{P}(y_i = 0.75; \mu = 0, \sigma^2 = 1)$.

We say the *likelihood* of some set of parameters, $\boldsymbol{\theta}$, given some data is the same as the probability of having observed the data given those parameters.

However, when we use SEM, we are typically not interested in the marginal distribution of y, but rather the *conditional distribution*, given the model covariates. So, in our example from above, we have $y = \beta_1 x_1 + \beta_2 x_2 + \epsilon$ and we note that

$$\mathbb{E}(y|x_1, x_2) = \mathbb{E}(\beta_1 x_1 + \beta_2 x_2 + \epsilon | x_1, x_2) = \beta_1 x_1 + \beta_2 x_2$$

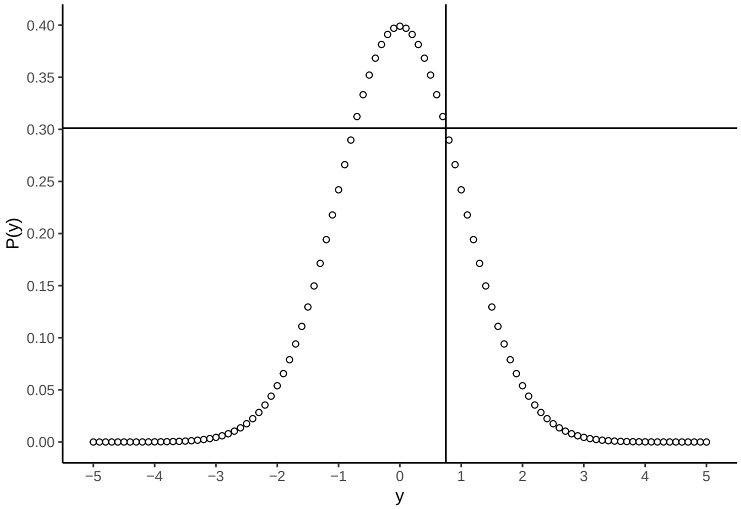


Figure 6: Probability distribution $y \sim \mathcal{N}(\mu = 0, \sigma = 1)$, $p(y_i = 0.75; \mu = 0, \sigma^2 = 1)$

since $\mathbb{E}(\epsilon|x_1, x_2) = \mathbb{E}(\epsilon) = 0$ and

$$\begin{aligned}\text{Var}(y|x_1, x_2) &= \text{Var}(\beta_1 x_1 + \beta_2 x_2 + \epsilon|x_1, x_2) \\ &= \text{Var}(\epsilon|x_1, x_2) = \text{Var}(\epsilon) = \sigma_\epsilon^2\end{aligned}$$

since in conditioning on themselves, the random variables x_1 and x_2 become constants with a variance of zero. This means that y, conditional on x_1 and x_2 is assumed distributed as $y \sim N(\beta_1 x_1 + \beta_2 x_2, \sigma_\epsilon^2)$. With this in mind, we can write

$$\mathcal{L}(\boldsymbol{\theta}; y_1, y_2, \ldots, y_n) = \mathcal{P}(y_1, y_2, \ldots, y_n; \boldsymbol{\theta})$$

where $i = 1, \ldots, n$ is the sample. Here and for the remainder of this section, we are conditioning on x_1, x_2 but we will leave this implicit so as to not clutter the notation.

Because of random sampling, we assume the observations are independent of each other, we can say that the joint probability $\mathcal{P}(y_1, y_2, \ldots, y_n; \boldsymbol{\theta})$ is the product of the individual probabilities, so the likelihood is the product, as well:

$$\mathcal{L}(\boldsymbol{\theta}; y_1)\mathcal{L}(\boldsymbol{\theta}; y_2) \cdots \mathcal{L}(\boldsymbol{\theta}; y_n) = \mathcal{P}(y_1; \boldsymbol{\theta})\mathcal{P}(y_2; \boldsymbol{\theta}) \cdots \mathcal{P}(y_n; \boldsymbol{\theta}).$$

When the errors are normally distributed, then we can write the likelihood of parameters $\boldsymbol{\theta}$ as

$$\mathcal{L}(\boldsymbol{\theta}; y_i) = \left(\frac{1}{\sqrt{2\pi\sigma_\epsilon^2}}\right)^n \prod_{i=1}^n \exp\left[-\frac{1}{2}\left(\frac{y_i - (\beta_1 x_{1i} + \beta_2 x_{2i})}{\sigma_\epsilon}\right)^2\right].$$

Notice that in modeling the conditional probability/likelihood, we are replacing the mean μ from above with the conditional mean, $\beta_1 x_1 + \beta_2 x_2$ in the numerator of the final expression.

From here, we see that the unknown parameters are $\boldsymbol{\theta} = (\beta_1, \beta_2, \sigma_\epsilon^2)$. We want to find the values of these that make the observed data, $\{(y_i|x_{1i}, x_{2i}) : i = 1, \ldots, n\}$ the most likely.

Because multiplying probabilities together is computationally troublesome, we instead use the *log likelihood*, which is

$$\ell(\boldsymbol{\theta}; y_i) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma_\epsilon^2) - \frac{1}{2}\sum_{i=1}^n \left(\frac{y_i - \beta_1 x_{1i} - \beta_2 x_{2i}}{\sigma_\epsilon}\right)^2.$$

By replacing the product by the sum, this expression is much more manageable. The values $\boldsymbol{\theta}$ that maximize the likelihood are the same that maximize the log likelihood, so there is no downside.

We find the values that maximize the log likelihood by taking the first derivative of the function with respect to $\boldsymbol{\theta}$ and setting this to zero (this is where the slope of the tangent line is zero; a horizontal line indicating either a maximum or a minimum), and solving for $\boldsymbol{\theta}$.

For σ^2, we have

$$\begin{aligned}
\frac{\partial \ell}{\partial \sigma^2_\epsilon} &= 0 - \frac{n}{2\sigma^2_\epsilon} + \frac{1}{2\sigma^4_\epsilon} \sum_{i=1}^{n} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 \\
0 &= -\frac{n}{2\sigma^2_\epsilon} + \frac{1}{2\sigma^4_\epsilon} \sum_{i=1}^{n} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 \\
2\sigma^2 \frac{n}{2\sigma^2_\epsilon} &= 2\sigma^2 \frac{1}{2\sigma^4_\epsilon} \sum_{i=1}^{n} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 \\
n &= \frac{1}{\sigma^2_\epsilon} \sum_{i=1}^{n} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 \\
\sigma^2_\epsilon n &= \sum_{i=1}^{n} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 \\
\hat{\sigma}^2_\epsilon &= \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{\beta}_1 x_{1i} - \hat{\beta}_2 x_{2i})^2 \\
&= \frac{1}{n} \sum_{i=1}^{n} \hat{\epsilon}_i^2
\end{aligned}$$

where $\hat{\epsilon}_i = y_i - \hat{\beta}_1 x_{1i} - \hat{\beta}_2 x_{2i}$ are the residuals (not the errors). Thus, the ML estimate of σ^2_ϵ is biased, as it does not use the sample corrected $(n-1)^{-1}$. But in large samples, the bias is negligable.

Similarly, for β_1, we have

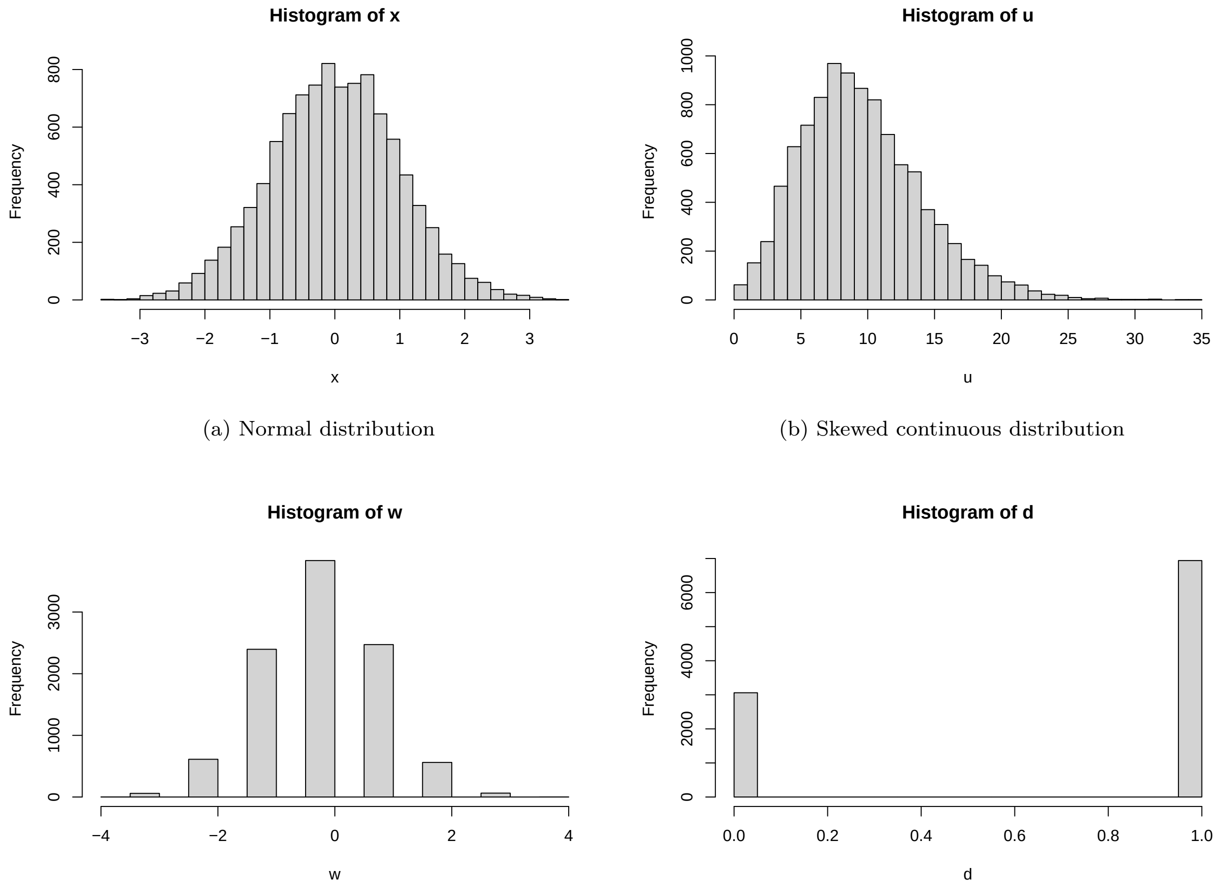
$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_1} &= 0 - 0 - 2 \sum_{i=1}^{n} x_{1i} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i}) \\
0 &= -2 \sum_{i=1}^{n} x_{1i} (y_{1i} - \beta_1 x_{1i} - \beta_2 x_{2i}) \\
&= \sum_{i=1}^{n} (x_{i1} y_i - \beta_1 x_{1i}^2 - \beta_2 x_{1i} x_{2i}) \\
\hat{\beta}_1 &= \frac{\sum_{i=1}^{n} x_{1i} y_i - \hat{\beta}_2 \sum_{i=1}^{n} x_{1i} x_{2i}}{\sum_{i=1}^{n} x_{1i}^2}
\end{aligned}$$

which is the ML estimate of β_1 and is identical to the ordinary least squares (OLS) estimate. $\hat{\beta}_2$ is arrived at analogously.

Normality

From the previous section, it should be clear that the assumption of normally distributed errors is vitally important for maximum likelihood estimation. This is because only when an outcome variable (conditional on the covariates, i.e., the errors) is normally distributed can it be summarized completely by the mean and variance, and only then can we calculate exact probabilities (and thus also likelihoods) based on those pieces of information alone.

The normal distribution is *continuous* and characterized by a concentration of the observations around the mean with symmetrical tails, see Figure 7a. But many social science constructs are not normally distributed and may be skewed or have excess kurtosis, see Figure 7b. Others may not be measured continuously, but rather using Likert-style scales leading to ordinal data, shown in Figure 7c. Still others may be nominal, i.e., dichotomous (yes/no) or multinomial (postal codes, outcomes of a die roll, etc.) in nature. Figure 7d shows an example of a variable that can take on one of two values, which is characterized by the binomial distribution.



(a) Normal distribution

(b) Skewed continuous distribution

(c) Coarsely measured normal distribution

(d) Binomial distribution

Figure 7: Normal distribution assumption

The consequences of nevertheless performing ML on nonnormal data has been widely investigated, however, and most analyses come to the conclusion that a violation of the normality assumption is not catastrophic. We generally differentiate between nonnormal continuous variables, and coarsely categorized variables, which can encompass ordinal as well as nominal variables.

Nonnormal continuous variables are arguably the less problematic of the two. They can lead to a χ^2 model fit test statistic that is too often significant when it should not be (incorrect rejection of the true model) and standard errors that tend to be too low, yielding too many significant parameter estimates (West, Finch, and Curran 1995). In `lavaan`, estimators like `MLM` (for complete data) and `MLR` (for complete and incomplete data) offer standard errors and χ^2 test statistics that are robust to nonnormality, see the `lavaan` tutorial website (https://lavaan.ugent.be/tutorial/est.html) and the 'brief user's guide' (https://users.ugent.be/~yrosseel/lavaan/lavaan2.pdf).

Ordered categorical variables display lower correlations between variables than continuous counterparts (West, Finch, and Curran 1995). This leads to a type of attenuation bias where the estimated effects in ML-based models with categorical outcomes are lower than they should be. The amount of bias depends on how coarse the variable was categorized, so to speak. Generally, the fewer the categories, the weaker the observed correlations and thus the more downwards biased the parameter estimates will be (downwards in the sense of towards zero). The same that goes for nonnormal continuous variables in terms of χ^2 tests of model fit and standard errors apply.

The solution to the problem of categorical (or even dichotomous) outcomes in SEM is to simply declare them as such. This depends on the software, but in `lavaan` it can be done using the shortcut `ordered = c("y1", "y2")` in the `sem()` function call, assuming there are two dependent variables are called `y1` and `y2` and they are ordered categorical. In the model syntax, the user can includes $M_k - 1$ thresholds (where M_k is the number of categories for the kth variable) per categorical endogenous variable (Rosseel and Loh 2021). For example, if each dependent variable were measured on a Likert style scale with five categories, four thresholds could be estimated (rather than means) using

```
y1 | t1 + t2 + t3 + t4
y2 | t1 + t2 + t3 + t4
```

where `t1`, `t2` etc. are used by convention to specify the thresholds (Rosseel et al. 2020). The `ordered` option turns on the diagonal weighted least squares (DWLS) estimator along with corrections for standard errors and test statistics. Many know this as the *weighted least squares means and variance adjusted* (WLSMV) 'estimator' in `Mplus`.[5] Those who are familiar with the latent variable derivation of the logistic and probit regression model (see Best and Wolf 2015) will be familiar with the logic of these estimators. Essentially, it is assumed that a normally distributed underlying variable is responsible for the crude measures on the categorical scale. Thresholds are calculated based on the proportion of responses per category, which are used to estimate the continuous latent variable distribution. In a `lavaan` model using the DWLS estimator, the estimated coefficients are those of a probit regression model, but can be interpreted as the linear effect of the covariate on the latent continuous underlying response variable, see also this post on the `lavaan` Google Groups forum: https://groups.google.com/g/lavaan/c/mG5Mjrf2jgo.

An empirical example

Let us now look at some actual data to investigate the model described above.

We will use the ALLBUS 2018 dataset, which includes data on xenophobic attitudes, anomia/populism sentiments,[6] as well as social class. We will start as a traditional regression model does, with observed variables and only one dependent variable. In the coming sessions, we will expand the model to look at mediation effects and move from observed variables to latent variables to deal with measurement error.

Preparing the data

For now, let us look at the variables:

- social class (x_1): subjective social class (`id02`)
- populism (x_2): "Politicians talk too much instead of acting" (`pa30`)
- xenophobia (y): "There is a dangerous amount of foreigners living in Germany" (`px06`)

Populism was recoded to match the direction of the xenophobia item, where higher values indicate agreement with the xenophobic/populist statement. They are both measured on 5-point a Likert scale. Social class is measured on a five point scale, from 1: "underclass" to 5: "upper-class". The following displays the variables:

[5] Estimator is placed in quotations because it is somewhat of a misnomer. Turning on WLSMV in `Mplus` employs the DWLS estimator plus robust standard error and test statistics, see this post on the `lavaan` Google Groups forum: https://groups.google.com/g/lavaan/c/Nymu7jmVUk8.

[6] Again, we focus instead on populism because it can arguably be treated as a continuous variable.

```
# Recode pa30 to match px06
df$pa30r <- abs(df$pa30 - 6)

# Re-label pa30
df$pa30r <- labelled(df$pa30r,
                     labels = c("LEHNE VOLL UND GANZ AB" = 1,
                                "LEHNE EHER AB" = 2,
                                "TEILS/TEILS" = 3,
                                "STIMME EHER ZU" = 4,
                                "STIMME VOLL UND GANZ ZU" = 5),
                     label = "POLITIKER REDEN ZU VIEL, HANDELN ZU WENIG (REKODIERT)")

# Inspect variables
head(df$id02); table(df$id02)
```

```
## <labelled<double>[6]>: SUBJEKTIVE SCHICHTEINSTUFUNG, BEFR.
## [1] 3 2 2 4 3 3
##
## Labels:
##  value                label
##    -50 KEINER DER SCHICHTEN
##     -9         KEINE ANGABE
##     -8          WEISS NICHT
##     -7           VERWEIGERT
##      1         UNTERSCHICHT
##      2      ARBEITERSCHICHT
##      3        MITTELSCHICHT
##      4  OBERE MITTELSCHICHT
##      5          OBERSCHICHT

##
##    1    2    3    4    5
##   87  916 2013  373   16
```

```
head(df$pa30r); table(df$pa30r)
```

```
## <labelled<double>[6]>: POLITIKER REDEN ZU VIEL, HANDELN ZU WENIG (REKODIERT)
## [1] 4 4 4 4 5 5
##
## Labels:
##  value                   label
##      1  LEHNE VOLL UND GANZ AB
##      2           LEHNE EHER AB
##      3             TEILS/TEILS
##      4          STIMME EHER ZU
##      5 STIMME VOLL UND GANZ ZU

##
##    1    2    3    4    5
##   41  234  764 1079 1331
```

```
head(df$px06); table(df$px06)
```

```
## <labelled<double>[6]>: ZUSTIMMUNG: BRD GEFAEHRLICH UEBERFREMDET
## [1] 2 5 4 3 5 4
##
```

```
## Labels:
##  value                label
##    -10          TNZ: FILTER
##     -9         KEINE ANGABE
##     -8          WEISS NICHT
##      1  STIMME GAR NICHT ZU
##      2 STIMME EHER NICHT ZU
##      3           WEDER NOCH
##      4       STIMME EHER ZU
##      5       STIMME VOLL ZU
```

```
##
##   1   2   3   4   5
## 725 656 520 793 534
```

The empirical or sample covariance matrix is

```r
cov(df[, c("px06", "id02", "pa30r")], use = "complete.obs")
```

```
##             px06       id02      pa30r
## px06   2.0016754 -0.2616604  0.5199974
## id02  -0.2616604  0.4532541 -0.1755239
## pa30r  0.5199974 -0.1755239  0.9737980
```

Fitting the model in lavaan

The package lavaan needs to be installed once with install.packages("lavaan"). This can be entered directly into the console or at the top of the script. To be able to use the package, we need to load it for every new R session:

```r
# install.packages("lavaan") # Uncomment to install
library(lavaan)
```

For users unfamiliar with R, SEM analyses can be carried out with almost no knowledge of the language. Typically, someone unfamiliar with R would prepare their data using some other statistical software, and then save the intended dataset as a .csv, .xlsx, .dta, .sav, etc. file. The user must then import the data, preferably as a dataframe, and the rest occurs using the lavaan syntax.

The ALLBUS 2018 file we will be using is an SPSS (.sav) file and we will use the read_sav() function from the haven package to load it.

```r
# install.packages("haven") # Uncomment to install
library(haven)

# Set working directory - change to your directory
setwd("F:/github-projects/intro-sem")

# Import ALLBUS 2018
df <- read_sav("04_data/allbus2018.sav")
```

To use lavaan, we create an R object using the assignment operator <-, see the model syntax example below. Here, the object has been called rm1 for 'regression model 1'. The object can be named anything that complies with naming conventions in R (e.g., the object name must start with a letter or dot, underscores and dots can be used to separate words, etc.). The model syntax is enclosed in quotes, either single '' or double "". This means that the model syntax is essentially a string that the lavaan package interprets in a second step. Once the model has been specified, we use the sem() function to 'fit' or 'estimate' the model. Notice a second object is made out of the fitted lavaan object. Here the fitted lavaan object has been named rm1.fit.

The default estimator is ML. We can switch the estimator in the fitting function call, e.g., `sem(model = rm1, data = df, estimator = "ULS")` for ULS. If we do not include the `estimator` argument, ML is used.

To reiterate, we specify the SEM by writing the model syntax as a string and saving it as an object. Then, in a second step, we run the `sem()` function on that object. The `sem()` function requires at least two arguments: `model`, i.e., the model object (here: `rm1`), and `data`, i.e., the dataframe or covariance matrix (along with the mean vector, if desired). That is, at a bare minimum, we must tell `lavaan` how the model is specified and where the data is. There are a number of other optional arguments that can be included. If they are not, the defaults of the `sem()` wrapper are used.[7]

By default, the model is essentially fit to centered data so that each variable has a mean of zero. When we turn the mean structure on, the default behaviour is to estimate an intercept or mean (depending on whether the variable is exogenous or endogenous) per equation.

We use the ~ or 'regressed on' operator to regress `px06` on `id02` and `pa30r`. We can further get into the habit of using *labels* to make the output easier to read, and to be able to extract pieces of the fitted object more easily. Labels work by premultiplying the variable by some string, as shown below. We will call the coefficients `b1` and `b2` for 'beta 1' and 'beta 2', respectively.

```
# Specify the model
rm1 <- '
# Regression
  px06 ~ b1*id02 + b2*pa30r
'
rm1.fit <- sem(model = rm1, data = df, estimator = "ML")
```

Once we have specified and estimated the model, we can inspect the results using `summary()`:

```
summary(rm1.fit)
```

```
## lavaan 0.6-12 ended normally after 1 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         3
##
##                                                   Used       Total
##   Number of observations                          3169        3477
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
```

[7] The main defaults of the `sem()` wrapper are: intercepts of the latent variables set to zero, the first factor loading in factor models is set to one, the residual variances and variances of exogenous latent variables are freely estimated, exogenous latent variables are set to covary. Further details can be found at https://rdrr.io/cran/lavaan/man/sem.html and https://cran.r-project.org/web/packages/lavaan/lavaan.pdf, or by entering `lavOptions()` into the console to get a full list of defaults. An explanation of the optional arguments can be found by entering `?lavOptions` in the console. There are other 'wrappers' with slightly different default options, like `cfa()` for example, see the `lavaan` tutorial website at https://lavaan.ugent.be/tutorial/cfa.html.

```
##   px06 ~
##     id02      (b1)   -0.398    0.035  -11.310    0.000
##     pa30r     (b2)    0.462    0.024   19.237    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .px06              1.657    0.042   39.806    0.000
```

The summary is broken up into roughly five parts. First, there is a short summary with some general information: the estimator, number of model parameters (β_1, β_2, ψ),[8] and the number of observations used and in total. By default, listwise deletion is used.

```
summary(rm1.fit)
```

```
## lavaan 0.6-12 ended normally after 1 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         3
##
##                                                   Used       Total
##   Number of observations                          3169        3477
...
```

Next, we have some information on model fit. Shown here is the chi square test of the hypothesis $\boldsymbol{S} = \boldsymbol{\Sigma}(\boldsymbol{\theta})$, that the model-implied covariance matrix is consistent with the observed data. We will discuss the chi square test and model fit when we discuss confirmatory factor analysis. Since the model is *just-identified*, i.e., the number of unique pieces of information $(\text{var}(y), \text{cov}(y, x), \text{var}(x))$ equals the number of parameters to estimate $\hat{\beta}_1, \hat{\beta}_2, \hat{\sigma}^2_\epsilon$, there is always a solution that satisfies $\boldsymbol{S} - \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}}) = \boldsymbol{0}$.

By setting the model-implied variances and covariances to the sample values, there are three equations, i.e., $\phi_1 = \text{var}(x_1), \phi_2 = \text{var}(x_2), \phi_{21} = \text{cov}(x_1, x_2)$. By covariance algebra, we know

$$
\begin{aligned}
\text{var}(y) = 2.00 &= \beta_1^2\phi_1 + \beta_2^2\phi_2 + 2\beta_1\beta_2\phi_{21} + \psi = \beta_1^2 0.45 + \beta_2^2 0.97 - 2\beta_1\beta_2 0.18 + \psi \\
\text{cov}(y, x_1) = -0.26 &= \beta_1\phi_1 + \beta_2\phi_{21} = \beta_1 0.45 - \beta_2 0.18 \\
\text{cov}(y, x_2) = 0.52 &= \beta_1\phi_{21} + \beta_2\phi_2 = -\beta_1 0.18 + \beta_2 0.97
\end{aligned}
$$

where we can start with either covariance and express one coefficient in terms of the other. For example,

$$
\begin{aligned}
\text{cov}(y, x_2) &= -\beta_1 0.18 + \beta_2 0.97 \\
0.52 &= -\beta_1 0.18 + \beta_2 0.97 \\
\hat{\beta}_1 &= \frac{\beta_2 0.97 - 0.52}{0.18} \\
&= \beta_2 5.39 - 2.89
\end{aligned}
$$

[8] Note, again, that the variances of the independent variables are not included as parameters. They are set to the sample values.

which we can substitute into $\text{cov}(y, x_1)$

$$\begin{aligned}
\text{cov}(y, x_1) &= \beta_1 0.45 - \beta_2 0.18 \\
-0.26 &= \beta_1 0.45 - \beta_2 0.18 \\
-0.26 &= (\beta_2 5.39 - 2.89)0.45 - \beta_2 0.18 \\
-0.26 &= \beta_2 2.42 - 1.30 - \beta_2 0.18 \\
\beta_2(2.42 - 0.18) &= 1.30 - 0.26 \\
\hat{\beta}_2 &= \frac{1.04}{2.24} \\
&= 0.46.
\end{aligned}$$

This we can now re-enter into $\text{cov}(y, x_2)$ for

$$\begin{aligned}
0.52 &= -\beta_1 0.18 + \beta_2 0.97 \\
&= -\beta_1 0.18 + 0.46(0.97) \\
\hat{\beta}_1 &= \frac{0.45 - 0.52}{0.18} \\
&= -0.39
\end{aligned}$$

which can both finally be entered into $\text{var}(y)$ to solve for ψ, the error variance

$$\begin{aligned}
2.00 &= \beta_1^2 0.45 + \beta_2^2 0.97 - 2\beta_1\beta_2 0.18 \\
&= -0.39^2(0.45) + 0.46^2(0.97) - 2(-0.39 \cdot 0.46)(0.18) + \psi \\
\hat{\psi} &= 2.00 - (-0.39)^2(0.45) - (0.46)^2(0.97) + 2(-0.39 \cdot 0.46)(0.18) \\
&= 1.66
\end{aligned}$$

which are, indeed, the *closed-form* solutions for the unknown parameters. These are the estimates of the SEM shown in the model output. When we plug these values back in to the model-implied matrix, we see that it results in an exact match of the sample covariance matrix:

```
# Observed covariance matrix
lavInspect(rm1.fit, "obs")
```

```
## $cov
##        px06   id02   pa30r
## px06   2.001
## id02  -0.262  0.453
## pa30r  0.520 -0.175  0.973
```

```
# Model-implied covariance matrix
lavInspect(rm1.fit, "implied")
```

```
## $cov
##        px06   id02   pa30r
## px06   2.001
## id02  -0.262  0.453
## pa30r  0.520 -0.175  0.973
```

```
# Residual matrix
lavInspect(rm1.fit, "resid")
```

```
## $cov
##        px06 id02 pa30r
## px06   0
## id02  0     0
## pa30r 0     0     0
```

This does not mean the model accurately describes the data, but rather that the hypothesis $\boldsymbol{S} - \boldsymbol{\Sigma}(\boldsymbol{\theta}) = \mathbf{0}$ *cannot be tested* with a just-identified model.

The information about the parameter estimates we can ignore because they are all just the default. Next, the regression coefficients are shown. These are the unstandardized coefficients of the independent variables, `id02` and `pa30r`, on the dependent variable, `px06`. The labels we used (`b1`, `b2`) are listed to the right of the the variable names, along with the standard errors, z- and p-values. The estimates are the same as we derived above.

```
summary(rm1.fit)
```

```
...
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   px06 ~
##     id02      (b1)   -0.398    0.035  -11.310    0.000
##     pa30r     (b2)    0.462    0.024   19.237    0.000
##
...
```

Finally, the only other parameter in the model, the error variance, is shown.

```
summary(rm1.fit)
```

```
...
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .px06              1.657    0.042   39.806    0.000
```

References

Best, Henning, and Christof Wolf. 2015. "Logistic Regression." In *The Sage Handbook of Regression Analysis and Causal Inference*, edited by Henning Best and Christof Wolf, 153–71. London, Thousand Oaks: Sage Publications.

Bollen, Kenneth. 1989. *Structural Equations with Latent Variables.* New York: John Wiley & Sons.

Ferron, John, and Melinda Hess. 2007. "Estimation in SEM: A Concrete Example." *Journal of Educational and Behavioral Statistics* 32 (1): 110–20.

Reinecke, Jost, and Andreas Pöge. 2010. "Strukturgleichungsmodelle." In *Handbuch der sozialwissenschaftlichen Datenanalyse*, edited by Christof Wolf and Henning Best. Wiesbaden: VS Verlag für Sozialwissenschaften.

Rosseel, Yves, Terrence D. Jorgensen, Daniel Oberski, Jarrett Byrnes, Leonard Vanbrabant, Victoria Savalei, Ed Merkle, et al. 2020. *lavaan: Latent Variable Analysis.* https://CRAN.R-project.org/package=lavaan.

Rosseel, Yves, and Wen Wei Loh. 2021. "A Structural After Measurement (SAM) Approach to SEM." https://osf.io/xme9g/.

West, Stephen, John Finch, and Patrick Curran. 1995. "Structural Equation Models with Nonnormal Variables: Problems and Remedies." In *Structural Equation Modeling: Concepts, Issues, and Applications*, edited by Rick Hoyle, 56–75. Thousand Oaks: SAGE Publications.