Exercise 07 - An introduction to structural equation modeling (SEM)

Zoltan Kekecs

January 9, 2022

Contents

1 Abstract

This exercise serves as an introduction to the topic of structural equation modeling (SEM). We will start by an overview of basic concepts such as latent variables, and path models. We will also learn how to interpret and create a path diagrams. We will use this background to create some simple SEM models to assess path models and to do a confirmatory factor analysis.

This exercise uses materials from Dr. Erin M. Buchanan's wonderful SEM course that is available on her Statistics of Doom website, which has multiple great videos and links to other useful materials. Please, check it out if you want to get more deeply involved with SEM: https://statisticsofdoom.com/page/structural-equation-modeling/

2 Data management and descriptives

2.1 Load packages

We will work with the following packages in this exercise:

```r
library(lavaan) # for SEM fit and model functions
library(semPlot) # for semPaths()
library(semptools) # for set_sem_layout
library(tidyverse) # for tidy code
library(CompQuadForm) # for mvnorm.kur.test and mvnorm.skew.test (prerequisite)
library(ICS) # for mvnorm.kur.test and mvnorm.skew.test
```

2.2 Dataset

We are going to work with the Industrialization and Political Democracy dataset, which is a well known dataset for the demonstration of different SEM techniques. This dataset is a built-in dataset in the lavaan package. As described in its help text, the dataset contains various measures of political democracy and industrialization in developing countries. The variables are labelled y1-y8 and x1-x3. Here are the descriptions of the 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

2.3 Data management

We will start by renaming the variables so they have more informative names, making it easier to interpret the output and also less likely to make errors in model specification.

```
my_data = PoliticalDemocracy


my_data = my_data %>%
  rename(GNP60 = x1,
         energ60 = x2,
         laborforce60 = x3,
         freepress60 = y1,
         freeopp60 = y2,
         fairelect60 = y3,
         efflegis60 = y4,
         freepress65 = y5,
         freeopp65 = y6,
         fairelect65 = y7,
         efflegis65 = y8)
```

3 SEM basic concepts

3.1 Common threads in multiple types of analyses

Structural equation modeling is an umbrella term for multiple different analysis procedures. The common theme behind these procedures tend to be the use of **path models** and **latent variables**.

In general, during SEM we are using **more than one multiple regression analyses** at the same time to answer our research questions, which are most often **theory-based confirmatory research questions**, assessing how well the observed data **fit a pre-specified theoretical model**. Another common thread among the different SEM models is that we usually visualize the model(s) we are testing with **path diagrams**.

3.2 Advantages over multiple regression

- we can model multiple relationships between different variables, and have multiple dependent variables in the models
- we can model latent (unmeasured, or directly unmeasurable) variables
- we can distinguish in our models between theorized causal and correlational relationships
- we can model error terms specifically related to specific variables, and also model the relationship between different error terms if we think that the error of a specific item is related to the error of other items (e.g. if I have a questionnaire, and 5 questions are negatively worded in the questions, this might mean that the measurement error of these items are related)

3.3 General research questions that SEM is used for answering

- **Is my theoretical model a good fit to the actual observed data?**
- **Which theoretical models fit better to the observed data?**
- **How can the theoretical model be improved to be a better fit to the data?**

3.4 Doing research with SEM

- Specify model
- Make sure that the model is identified
- Collect data

- Check model fit
- Interpret estimates (only if model fit is good)
- Consider equvivalent models
- Report the findings

3.5 Concepts related to SEM

As mentioned above, SEM is a confirmatory procedure where we are the fit of the observed data to a theoretical model.

This study was testing a theoretical model about the relationship of political democracy and industrialization in countries all over the world.

In a nutshell, according to the theory of the authors, freedom of the press, freedom of political opposition, fairness of elections, and effectiveness of the elected legislature are determined by the overall political democracy in a country (a latent factor), and this political democracy is determined by how industrialized the country is (another latent factor), which in turn also affects (or manifests in) GNP, inanimate energy consumption, and percentage of the labor force in industry. Democracy-related observed variables are measured in 1960 and 1965, so the model also incorporates time: it proposes two separate latent factors: political democracy in 1960 and political democracy in 1965, where political democracy in 1965 is influenced by political democracy in 1960, and the same observed measures taken in 1960 and 1965 co-vary, incorporating common measurement error in these measures that are not related to political democracy, rather, to the way the data was collected and measured.

There are a couple of latent factors in this model, so first lets recap what this means.

3.5.1 Latent variables

We have already met latent factors in exploratory factor analysis (EFA). There, latent variables were common factors underlying observed responses, behaviors, or answers to questionnaire items.

Latent variables are not directly measured, so there is no variable corresponding to them in the raw data (although if we estimate these factors using factor analysis we can save these factor scores in our dataset later).

Nevertheless, latent variables are linked to observed variables. Sometimes they are underlying factors that cause related observed variables to change similarly, other times these are the variables that are causally affected by some observed variables. The important thing is that we do not have directly observed data about these variables, we infer them based on the observed variables. Because there is no observed data about these variables, we often call them abstract.

Most often, the latent variables are latent for a reason. They are either impossible or incredibly hard to directly measure, so it is better to collect data on the observable related variables, and infer their value from these observable data. A good example is IQ. Most of us would agree that there is something like intelligence, people are different in how intelligent they are, or how intelligently they think, reason, or behave. Nevertheless, there is no direct measure of intelligence that has been developed yet. It is not like the height of a person, which you can just directly measure with a tape measure. Instead, psychologists infer how intelligent a person is based on a collection of standardized test items or questions. We can directly observe the responses to these questions, and based on these we can infer how intelligent they are.

3.5.2 Manifest (observed) variables

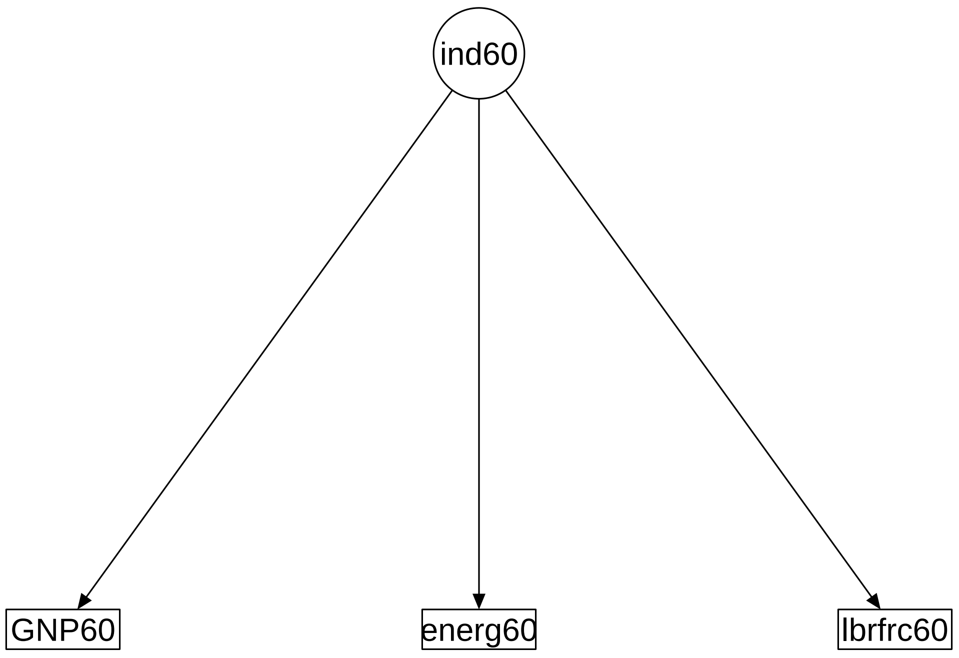
The manifest variables, or observed, variables are the ones we can directly measure. We call them “manifest” because in a common factor model these observed variables are the observable manifestations of the same underlying (latent) factor.

3.5.3 Path diagram

SEM models are often visualized using path diagrams. Lets look at a very simple model to begin with, where we theorize that the level of industrialization of the country is manifested in the gross national product (GNP), the inanimate energy consumption in the country, and the percentage of the labor force in the industry sector.

The following path diagram can represent this model.

```
model <- '
    ind60 =~ GNP60 + energ60 + laborforce60
'
fit <- sem(model, data = my_data)

plot = semPaths(fit, label.scale=F, nCharNodes = 8,
         sizeMan2=3.5, sizeMan=10, asize=3, edge.color="black", residuals = F, fixedStyle = 1)
```



On the path diagram, **observed (manifest) variables are represented by squares**, and **latent variables by circles**.

3.5.4 Directional relationships

In SEM we can model theorized directional relationships. In the above example we theorize that the industrialization of the country is responsible for (is causing) the variations in GNP, increased energy consumption, and larger labor force working in industry, and not the other way around. This is represented by the arrows leading from industrialization toward the other variables, so whether we have a one- or a two-headed arrow does matter in the path diagram. So this models causal relationship.

3.5.5 Exogenous variables

Exogenous variables are synonymous with **independent variables**. These are the variables that according to our theory **cause the change/variation** in the other (endogenous) variables. Since exogenous variables are causing change in some other variable, they have **single headed arrow(s)** pointing away from them, or in other words, the arrow is **EXITING** them. In the above example industrialization is an exogenous variable, since it has one-headed arrows starting from it and pointing toward the manifest variables.

3.5.6 Endogenous variables

Exogenous variables are synonymous with **dependent variables**. The change or variations in thes variables are caused by other variables in our SEM model, so they have single-headed arrow(s) pointing towards them, or in other words, the arrow is **ENTERING** them. In the above example GNP, inanimate energy consumption, and percentage of labor force in industry are endogenous variables, since variations in them are caused by variation in industrialization of the country, and thus, one-headed arrows starting from industrialization are entering them/pointing towards them.

In the example about industrialization all endogenous variables are manifest variables and the exogenous variable is a latent variable. This is common, but not necessarily always the case. Sometimes manifest variables can be exogenous (for example in path analysis and mediation models), and similarly, endogenous variables can be latent variables.

3.5.7 Endogenous AND Exogenous variables

In more complex models, variables can be exogenous and endogenous at the same time (i.e. they have both single-headed arrows exiting and entering them). This is the case if the model posits that the variable that causes change in some other variables is in turn also caused to be changed by yet another variable. For example variation in number of sexual intercourse per month in a marriage might be caused by variations in marital satisfaction, which in turn might be caused by changes in expression of intimacy between the partners. Here, marital satisfaction is both an exogenous and an endogenous variable, since it is both a proposed cause of change in frequency of intercourse, but its own variation is caused by another variable, expression of intimacy.

3.5.8 Residual correlations

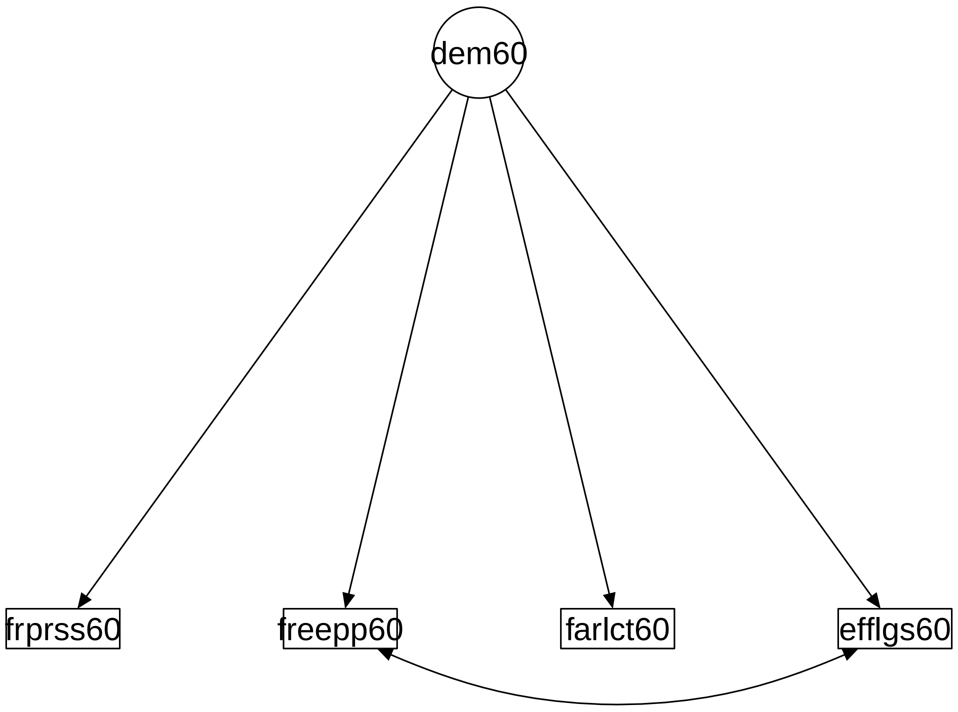
In SEM we can not only model causal relationships, we can also model correlations. If two variables are influenced by the same underlying third variable (cause), they will be naturally correlated with each other. So in a sense, when we have the same exogenous variable influencing two endogenous variables, we have already modeled their correlation. However, if we believe that after taking into account the variance explained by the influence of the exogenous variable on there variables the two endogenous variable would still be correlated in some other ways as well (maybe there are other third variables that we don't have in our model), we can model the correlation of the two variables. We call this "residual correlation" because we expect that after we build two regression models where the exogenous variable is the predictor and the endogenous variables are the predicted outcome variables, the residual error of these models would still significantly correlate.

In the example below, we are looking at another subset of the larger theoretical model discussed in the beginning. In the theoretical model we theorize that the level of political democracy of the country determines the freedom of the press, the freedom of political opposition, the fairness of elections, and the effectiveness of the elected legislature. In the original model the authors also indicate that there is a correlation between the the freedom of political opposition and the effectiveness of the elected legislature. The authors claim that this correlation remains even after taking into account the variance explained by democracy level. (Maybe non-intuitively, the more freedom the opposition has, the more effective the elected legislature is.)

This residual correlation between freeopp60 and efflegis60 is built into the model in the path diagram below. Note that by convention, residual correlations are represented by double-headed arrows between the variables. In most path diagrams arrows representing correlational relationships are curved.

Importantly, note that this does not mean that freedom of the press would not be related to freedom of the political opposition. These variables are of course correlated, since they are influenced by the same exigenous variable, political democracy level. The fact that freepress60 and freeopp60 do not have double-headed arrows between each other merely represents that after taking into account the co-variation of these variables caused by political democracy, there is no remaining (residual) correlation to explain.

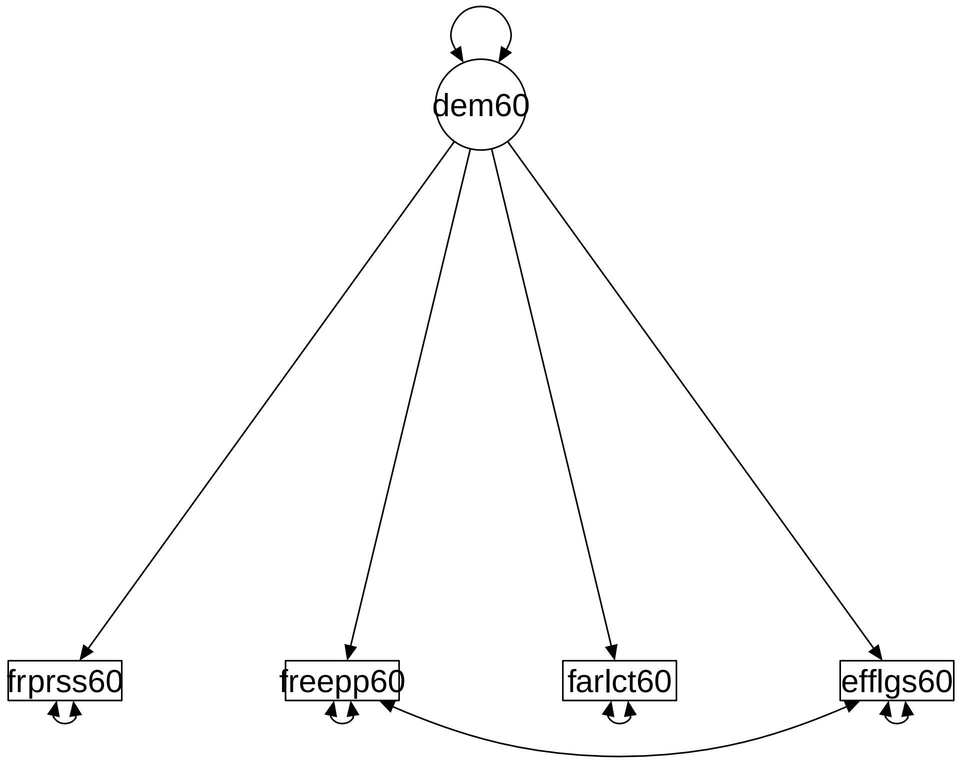
```
model2 <- '
    dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60
    freeopp60 ~~ efflegis60
'
fit2 <- sem(model2, data = my_data)

plot2 = semPaths(fit2, label.scale=F, nCharNodes = 8,
         sizeMan2=3.5, sizeMan=10, asize=3, edge.color="black", residuals = F, fixedStyle = 1)
```



3.5.9 Variance vs. residual error

In statistics there is no perfect model, there is always some variation in the variables that is unaccounted for by our model. The previous graphs were incomplete make make them simpler, but now we will add the unexplained variability to the graphs as well.

```
model3 <- '
    dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60
    freeopp60 ~~ efflegis60
'
fit3 <- sem(model3, data = my_data)

plot3 = semPaths(fit3, label.scale=F, nCharNodes = 8,
         sizeMan2=3.5, sizeMan=10, asize=3, edge.color="black", residuals = T, fixedStyle = 1)
```



This graph now also contains two-headed arrows pointing back to each variable itself. These represent the fact that the variables have variability left that is not explained by variables included in the model. You can think of these as residual (unexplained) variance of the variables in the model.

This residual variability of endogenous variables is called residual error (other names for this includes “error”, “residuals”, and “disturbance”). Note that the residual error is classically represented as a circle with a single-headed arrow pointing into the endogenous variable. However, this not commonly used nowadays, because it makes the path diagrams convoluted. Instead, residuals are commonly represented with horseshoe-like double-headed arrows pointing back to the endogenous variable, like in the plot above.

For exogenous-only variables (variables that only have arrows exiting them, and no single-headed arrows pointing toward them), the variability is called variance. It is important to distinguish between variance and residual error, because they are handled differently by the SEM models.

3.6 Different types of relationships in SEM models

In SEM modeling we usually distinguish between three types of relationships between variables:

Causal relationship where a latent variable is the underlying cause of a manifest variable

- This type of relationship is called **latent variable loading** in SEM
- On the path diagram this is denoted by a single-headed arrow pointing from the latent variable to the manifest variable
- In the code and output it is represented by [cause variable name] =~ [outcome variable name]
- The part of a SEM model which contains these types of relationships is often called the **measurement model**

Causal relationship where a manifest or a latent variable is the underlying cause of a latent variable, or a manifest variable is the underlying cause of another manifest variable

- This type of relationship is called **regression** in SEM
- On the path diagram this is denoted by a single-headed arrow pointing from the causing variable to the caused variable
- In the code and output it is represented by [outcome variable name] ~ [cause variable name]
- The part of a SEM model which contains these types of relationships is often called the **structural model**

Correlational relationship

- This type of relationship is called **residual correlation** or **covariance** in SEM
- In the path diagram this is denoted by a double-headed arrow (often curved) between the two correlated variables
- In the code and output it is represented by [variable name] ~~ [variable name]

3.7 Model specification

We can use the above operators to specify our SEM model using the sem() or cfa() functions in the lavaan package in R.

For example, let's say we want to specify the model shown in the path diagram above. That model proposes that political democracy (we call this dem60 in our model) is a latent factor that is a common underlying factor (cause) the manifest variables freedom of press (freepress60), freedom of the opposition (freeopp60), fairness of the elections (fairelect60), and the effectiveness of the elected legislature (efflegis60), and that after accounting for this factor, there is still some leftover/residual correlation, or covariance, between freedom of the opposition (freeopp60), and the effectiveness of the elected legislature.

Based on the above nomenclature we need to include in the model four latent factor loadings and one covariance. This model is usually written up as a separate string (text) object, which is later passed through the sem() or cfa() functions.

The model object needs to be a string vector (this means that it needs to be in quotes, either "..." or '...'), with each model relationship in separate lines.

The latent factor loading of dem60 on freepress60 can be written up as dem60 =~ freepress60, and the latent factor loading of dem60 on freeopp60 can be written up as dem60 =~ freeopp60 and so on. We could add these one by one in the model object, but if the relationship and the left side of the argument is the same, we can just add items on the right side of the equation with + signs. So this model specification:

" dem60 =~ freepress60 dem60 =~ freeopp60 dem60 =~ fairelect60 dem60 =~ efflegis60 "

is equvivalent to this model specification:

" dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60 "

The model also needs to include the covariance between freeopp60 and efflegis60, so the model specification would look like:

" dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60 freeopp60 ~~ efflegis60 "

Additionally, the model also contains residuals (error) for each endogenous variable (in this model these are the manifest variables freepress60, freeopp60, fairelect60, efflegis60), and variance for the exogenous variable (in this model it is the latent factor dem60). However, lavaan automatically adds these to the model so we do not have to write thise in the model object itself.

```
model3 <-
"
    dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60
    freeopp60 ~~ efflegis60
"
```

practical note: we are using the lavaan package for SEM in this exercise. Lavaan searches for the variable names that we specify in the model. If the variable name appears in our dataset, lavaan takes these as manifest (observed) variables, and if the variable name does not exist in our dataset, lavaan takes these as latent variables. So it is important to choose variable names correctly, and if you want to include a variable as latent variable in a model, be careful not to have a matching variable name in your dataset.

3.7.1 Fitting the model

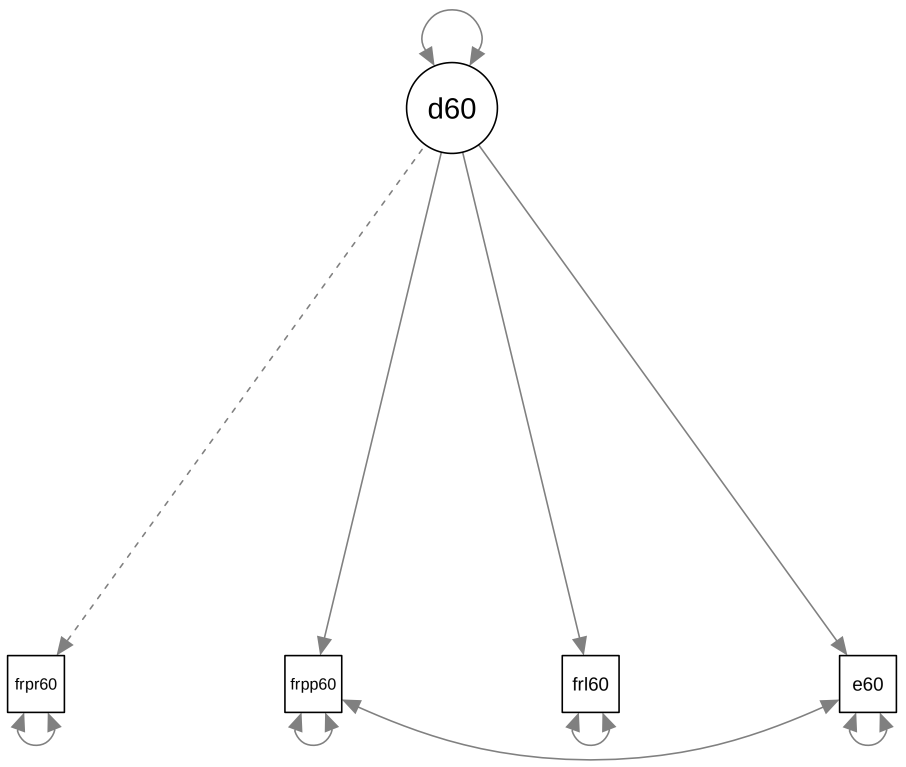
After saving the model in an object (usually named "model"), we can pass it through the sem() or cfa() function. In these functions there are a lot of parameters that can be used, but as a minimum we need to give the model object name and the name of the dataset. Instead of the 'data =' argument, we can also give these functions a correlation or covariance matrix using the 'sample.cov =' argument, similarly to an exploratory factor analysis. This is because the SEM analysis is not actually done on the raw data, but on the covariance matrix of the observed variables. So we do not necessarily need the raw data. But if we use the 'sample.cov =' argument, we also need to specify the number of observations using the 'sample.nobs =' argument. The sem() function fits to model, so we usually save the result of this function in a model object named "fit".

```
fit3 <- sem(model3, data = my_data)
```

3.7.2 Model visualization

The semPaths() function from the semPlot package can be used to visualize the SEM model. As with all SEM-related functions, it has a lot of parameters that can be tweeked in it if needed, but as a minimum it requires a fitted model object as an input, like this:

```
semPaths(fit3)
```



It is always a good idea to graph your model with semPaths() to verify that the model you specified and is fit to the data is actually the model you intended to run.

3.7.3 Model results

We can get the results of the model using the summary() function. By default it only contains a small subset of all information that we can be interested in, here it only lists the estimator, information on degrees of freedom, the chi squared test statistic and related p-value, and the estimated parameter values (factor loadings, covariance, residuals and variance respectively). Depending on your research questions, the model, and the assumptions you have options to request more information. But before we can interpret the the results or request more results we need to check if our model is "identified" or not.

```
summary(fit3)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.104    0.195    5.659    0.000
##     fairelect60       1.065    0.154    6.923    0.000
##     efflegis60        1.119    0.159    7.044    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.376    0.591    2.329    0.020
##    .freeopp60         8.779    1.681    5.224    0.000
##    .fairelect60       4.487    0.959    4.676    0.000
##    .efflegis60        4.290    0.990    4.332    0.000
##     dem60             5.411    1.215    4.453    0.000
```

3.8 Identification of the model

SEM models need to be **"identified"** for the fit function to run.

The model is identified if the mathematical formulas it is comprised of has **only one solution**.

For example in the equation 2x = 4, x has only one possible solution, 2. So this would be called a "model" that is identified. On the other hand in the equation 2x + y = 4, x and y have multiple possible solutions, so this "model" would be not identified.

In SEM, whether a model is identified depends on the **degrees of freedom (df)** of the model. Unlike most statistical models, in SEM degrees of freedom **DO NOT depend on sample size**. Rather, degrees of freedom of the SEM model is calculated as the **difference between the total number of possible estimated parameters** of a "null model" with no latent variables, **and the number of free parameters** in our actual model.

To be identified the model needs to have 0 or higher degrees of freedom. You can think of this as a prerequisite or an assumption of the analysis.

3.8.1 Counting degrees of freedom

The number of all possible parameters of the manifest variables is calculated as p*(p+1)/2, where p is the number of manifest (observed) variables in the model.

In the example model above we have four manifest variables: freepress60, freeopp60, fairelect60, and efflegis60. So the total number of possible parameters is 4*(4+1)/2 = 10.

The number of **free parameters** depend on the number of relationships and residuals and variances we include in our model. In detail, we have the following estimated parameters:

- 3 latent factor loadings (between dem60 and the manifest varaibles freeopp60, fairelect60, efflegis60)
- 1 covariance (between freeopp60 and efflegis60)
- 4 residuals (one for each manifest variables)
- 1 variance (for our single latent factor)

We have a total of 9 solid grey arrows. So the df for our model is 10 - 9 = 1.

Advanced note: In most SEM models, where all exogenous variables are latent variables, the number of **"free parameters"** match the number of **"estimated parameters"** (in the model summary this is referred to as "Number of model parameters"). However, in some SEM models observed variables are exogenous. For **observed exogenous variables we do not need to estimate variance and covariance with another observed variable**, because we can directly observe these in the data. Nevertheless, these parameters are not "fixed parameters", since we are not setting them to a specific value, so **they are also considered "free parameters"**. You can think of them as they are free to depend on the data, but they do not need to be estimated, since they are directly observable in the data. This is the case in purely path models or mediation models, where we do not have latent variables in the model, and thus some observed variables are exogenous. A discussion about this can be found here: https://groups.google.com/g/lavaan/c/uAFUkfHoU4k?pli=1

This can be verified in the model summary above, which confirms that Degrees of freedom = 1, and lists these estimated parameters in the parameter estimates section. (The parameter dem60 =~ freepress60 is also included in that list, but it has a fixed value of 1, and has no p-value associated with it. We will talk about this in the Model constraints and scaling section below.)

A model with a df = 0 is called a **"just identified model"**, and since the df is often used in the denominator of formula of fit indexes, these indexes will be unreliable or un computable for just identified models, even though the model estimates would still be usable. To be able to use the fit indexes as well, the df needs to be at least 1, or ideally higher.

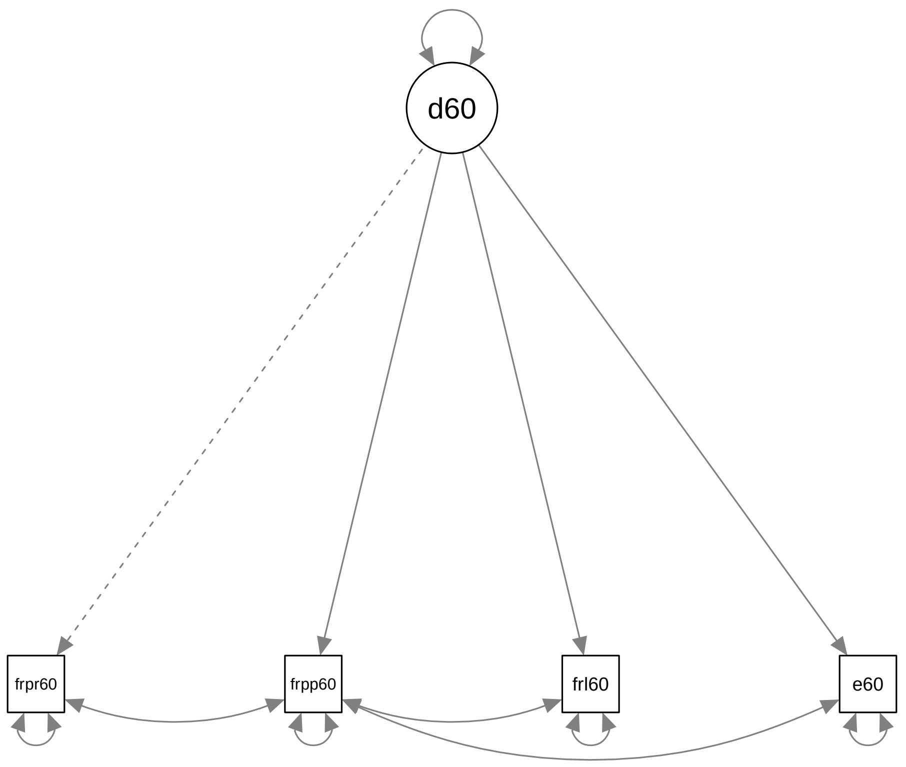
If we have more estimated (free) parameters in our model than the total number of possible parameters in the "null model" (the model without any latent factors but including all possible covariances), the degree of freedom drops below 0. this produces a model that is "not identified". This produces a warning message when running the sem() function, and the model fit and and estimates are not produced in the model summary.

In this example below we include the covariance of freeopp60 with every other manifest variable in the model. This pushes the degrees of freedom below 0.

```
model4 <-
  "
    dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60
    freeopp60 ~~ efflegis60 + fairelect60 + freepress60
"

fit4 <- sem(model4, data = my_data)
```

```
## Warning in lav_model_vcov(lavmodel = lavmodel, lavsamplestats = lavsamplestats, : lavaan WARNING:
##     Could not compute standard errors! The information matrix could
##     not be inverted. This may be a symptom that the model is not
##     identified.
```

```
semPaths(fit4)
```



```
summary(fit4)
```

```
## lavaan 0.6-9 ended normally after 38 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        11
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                    NA
##   Degrees of freedom                                -1
##   P-value (Unknown)                                 NA
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.300       NA
##     fairelect60       1.098       NA
##     efflegis60        1.145       NA
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        1.571       NA
##    .fairelect60      -1.730       NA
##  .freepress60 ~~
##    .freeopp60        -0.651       NA
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.542       NA
##    .freeopp60         6.506       NA
##    .fairelect60       4.293       NA
##    .efflegis60        4.188       NA
##     dem60             5.245       NA
```

3.8.2 Model constraints and scaling the model

Why did we only count solid arrows? What about the dashed arrow pointing from dem60 to freepress60? Isn't the part of our model? Yes, it is, but that parameter is fixed to a value of 1 instead of being estimated. The reason for this is that ideally we would like to have **as high degrees of freedom as we can**, and we also need to **scale the model**.

The sem models generally need a scaling parameter, which is done by fixing one of the parameters to 1 instead of estimating it. If we don't tell it otherwise, the sem() function automatically fixes the **first path (latent variable loading) in each measurement model to 1**. This is a common practice in the field, so do not deviate from this unless you know what you are doing. Generally **it does not matter which latent factor loading you fix** to one in terms of model fit. Since we only have a single measurement model in our model above, we only have a single fixed parameter. Fixing the first path of the measurement models also helps with keeping the degrees of freedom above 0. In fact, since the df = 1 now, if we did not have this parameter fixed, the model degree of freedom would be already 0. Because these **fixed parameters** are also used in the scaling of the model, they are also called **"reference parameters"** or **"marker parameters"**.

In some cases we might have more estimated parameters than the model can handle. In these cases, we should consider **removing some of the free parameters**. Another option is to **merging some parameters**, to increase the degree of freedom, or put some other constraint on the parameters so they become a linear transformation of each other, this way we reduce the number of parameters we need to estimate, and thus, increase the degree of freedom. For example we can **constrain two or more parameters to be equal to each other**, or we can constrain a parameter to be the inverse or exactly double the size of another parameter.

So in summary:

- The parameters that are being estimated based on the data are called **free parameters**.
- In SEM we also often have parameters that are set to a certain value to aid with model identification. These are called **fixed parameters**, and their value is traditionally set to 1.
- We can also apply specific constraints to the parameters that are being estimated, to decrease the number of parameters that are being estimated, for example, parameters can be set to be equal ("equality constraint"), or to be a certain linear transformation of each other. These are called **constrained parameters**.

3.9 Model estimation

SEM models do not have a closed form solution that can be easily computed, like the solution for simple linear models. So the model solution is estimated. When an estimator is used, an iterative process takes place which starts from a likely solution and with each iteration the solution is fine-tuned to get closer and closer to the actual/best solution.

There are multiple estimators that can be used, such a Maximum likelihood, Asimptotically distribution free estimator, Generalized least squares, Unweighted least squares, Weighted least squares, Two staged least squares, etc.. By far, the most popular method is the **Maximum likelihood (ML) estimator**.

The estimators can be changed by using the estimator = argument. On this link you can find the different options and a short description: https://lavaan.ugent.be/tutorial/est.html. The default is estimator = "ML" referring for the maximum likelihood estimator, so if you are OK with the ML estimator, you do not have to specify the estimator argument.

3.9.1 Maximum likelihood estimator

Assumptions of the ML estimator

- Assumes that the observed data entered into the model comes from a **multivariate normal distribution**. This can be checked for example using the mvnorm.kur.test and mvnorm.skew.test as discussed in the exercise about factor analysis (significant p-value indicates significant non-normality). If the mutlivariate normaliy assumption does not hold true, it is still generally considered OK to enter binary categorical variables and skewed continuous variables into the models as exogenous variables, but if they are endogenous variables in the model, another estimator is advised. If the assumption of normality is violated, the standard errors are usually too large, and the p-values are not reliable. Furthermore, the model fit tends to be overestimated (the fit indexis are unreliable). In this case, normality-adjusted **robust standard errors** are used and correction to the model fit indices. The model fit statistics aslo have robust variants. For example the Chi-squared fit statistic can be corrected by the Satorra-Bentler statistic. We can get these robust estimates using the estimator = "MLM" argument. You can also use bootstrapping instead by using the se = "bootstrap" and test = "bootstrap" arguments, to get bootsrapped standard errors and p-values. However, be aware that this can run for a few minutes or even hours depending on the amount of data and the complexity of the model (that is why this line is commented in the code below).
- Assumes unstandardized manifest variables (so do not scale or Z-transform variables before entering data to the sem() function)
- Unlike some other estimators, the ML estimator is **scale-free**, so it does not assume that the variables in the model have the same scale.
- Requires a positive definite matrix to run. When this is violated, the sem() function returns a warning. It tends to be violated when manifest variables are highly correlated with each other, or latent variables are higly correlated with each other (r > 0.9).

```
mvnorm.kur.test(my_data[,c("freepress60", "freeopp60", "fairelect60", "efflegis60")])
```

```
##
##  Multivariate Normality Test Based on Kurtosis
##
## data:  my_data[, c("freepress60", "freeopp60", "fairelect60", "efflegis60")]
## W = 14.611, w1 = 0.88889, df1 = 9.00000, w2 = 1.33333, df2 = 1.00000,
## p-value = 0.1113
```

```r
mvnorm.skew.test(my_data[,c("freepress60", "freeopp60", "fairelect60", "efflegis60")])
```

```
##
##  Multivariate Normality Test Based on Skewness
##
## data:  my_data[, c("freepress60", "freeopp60", "fairelect60", "efflegis60")]
## U = 7.8762, df = 4, p-value = 0.09622
```

```r
# solution of the ML estimator
summary(fit3)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.104    0.195    5.659    0.000
##     fairelect60       1.065    0.154    6.923    0.000
##     efflegis60        1.119    0.159    7.044    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.376    0.591    2.329    0.020
##    .freeopp60         8.779    1.681    5.224    0.000
##    .fairelect60       4.487    0.959    4.676    0.000
##    .efflegis60        4.290    0.990    4.332    0.000
##     dem60             5.411    1.215    4.453    0.000
```

```r
# solution of the ML estimator with robust SE and test statistics
fit3_MLM <- sem(model3, data = my_data, estimator = "MLM")
summary(fit3_MLM, fit.measures = T)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##                                               Standard      Robust
##   Test Statistic                                 1.663       1.307
##   Degrees of freedom                                 1           1
##   P-value (Chi-square)                           0.197       0.253
##   Scaling correction factor                                  1.273
##        Satorra-Bentler correction
##
## Model Test Baseline Model:
##
##   Test statistic                               159.183     201.252
##   Degrees of freedom                                 6           6
##   P-value                                        0.000       0.000
##   Scaling correction factor                                  0.791
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.996       0.998
##   Tucker-Lewis Index (TLI)                       0.974       0.991
##
##   Robust Comparative Fit Index (CFI)                         0.997
##   Robust Tucker-Lewis Index (TLI)                            0.985
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -699.966    -699.966
##   Loglikelihood unrestricted model (H1)       -699.135    -699.135
##
##   Akaike (AIC)                                1417.933    1417.933
##   Bayesian (BIC)                              1438.790    1438.790
##   Sample-size adjusted Bayesian (BIC)         1410.424    1410.424
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.094       0.064
##   90 Percent confidence interval - lower         0.000       0.000
##   90 Percent confidence interval - upper         0.339       0.293
##   P-value RMSEA <= 0.05                          0.238       0.312
##
##   Robust RMSEA                                               0.072
##   90 Percent confidence interval - lower                     0.000
##   90 Percent confidence interval - upper                     0.363
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                         0.017       0.017
##
## Parameter Estimates:
##
##   Standard errors                           Robust.sem
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.104    0.168    6.566    0.000
##     fairelect60       1.065    0.157    6.802    0.000
##     efflegis60        1.119    0.154    7.259    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        2.696    1.350    1.997    0.046
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.376    0.575    2.396    0.017
##    .freeopp60         8.779    1.683    5.218    0.000
##    .fairelect60       4.487    0.964    4.656    0.000
##    .efflegis60        4.290    1.170    3.668    0.000
##     dem60             5.411    0.957    5.656    0.000
```

```
# bootstrapped ML solution for robust SE and test statistics
# fit3_boot <- sem(model3, data = my_data, se = "bootstrap", test = "bootstrap")
# summary(fit3_boot)
```

3.9.2 Unweighted least squares estimator

- does not assume a matrix that is not positive definite
- it does assume that the manifest variables are on the same scale. This means that before entering into the sem() function, manifest variables need to be re-scaled if they are not on the same scale already.

3.10 Interpreting model fit

We have met model fit indexes before when we talked about model comparison. These indicate how well the data and the model fit with each other.

- **Model fit and model significance are not the same!** The full model R^2 is one example for a model fit index that is used in linear regression. Even if you have a significant linear regression model (a model predicting the outcome significantly better than the null model), the R^2 might be still very low, indicating that your model is practically not super useful, even if it is slightly better than a model without the predictors.

- **There are many fit indexes** for SEM models, and there is no clear consensus on which one is the best, so usually the recommendation is to report multiple fit indexes and interpret them together. Due to no clear consensus on which ones to use, people often abuse them, cherry-picking the ones that indicate good model fit for their models. Do not do this, instead, report always the same fit indexes.
- A limitation of fit indexes is that because they provide a **single value for the whole model**, they don't tell you which part of the model causes the problem. For the same reason, in complex models, they might **hide bad model sections** if the rest of the model has a very good fit.
- Model fit also **does not inform you about predictive efficiency**. Other measures like the squared multiple correlations (SMCs) can be used in combination with them to get a better picture about how well the model explains the variance of different outcome variables.
- The fit index **does not tell you if the model is theoretically meaningful**. A theoretically inplausible model can still have a good fit index.

3.10.1 Types of model fit

The traditional goodness of fit test for SEM models uses a Chi-squared test, which compares the covariance matrix of the observed variables with the covariance matrix estimated by our model. However, unlike the regression model significance test, we want this to be non-significant, because significant difference would indicate that our model was not able to reproduce the observed covariance matrix. Using a traditional null-hypothesis significance test (NHST) to "support the null" is problematic, because the larger the sample size the larger the chance that the test will be significant even for trivially small deviations. This created a biased incentive for small samples, in which this test was rarely significant. Nowadays the Chi-squared test is not really used as an important model fit criteria, even though most people still report this test statistic by convention.

There are more modern model fit indexes, generally falling into two categories: **goodness of fit** indexes and **badness of fit** indexes. When these model fit indices were invented, arbitrary cutoff points were used to indicate good and bad fit. The most influential publication in this area was Hu and Bentler's (1999) paper investigating the influence on type I and II error rate of different cutoff points for rejecting models based on model fit using a simulation study. In this paper they recommend the following cutoffs for rejecting models for bad fit:

Goodness of fit statistics:

- Tucker-Lewis Index (TLI) < 0.95
- Bollen's Fit Index (BL89) < 0.95
- Comparative Fit Index (CFI) < 0.95
- Relative Noncentrality Index (RNI) < 0.95
- Gamma hat < 0.95
- McDonald's Centrality Index (Mc) < 0.90

Badness of fit statistics:

- standardized root mean squared residual (SRMR) > 0.08
- root mean squared error of approximation (RMSEA) > 0.06

Out of these model fit indexes the TLI, CFI, the RMSEA are the most commonly used fit indexes. You can extract them from the model fit object by specifying the fit.measures = T argument in the summary() function.

```r
# unstandardized estimates
summary(fit3, fit.measures = T)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Model Test Baseline Model:
##
##   Test statistic                               159.183
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.996
##   Tucker-Lewis Index (TLI)                       0.974
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -699.966
##   Loglikelihood unrestricted model (H1)       -699.135
##
##   Akaike (AIC)                                1417.933
##   Bayesian (BIC)                              1438.790
##   Sample-size adjusted Bayesian (BIC)         1410.424
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.094
##   90 Percent confidence interval - lower         0.000
##   90 Percent confidence interval - upper         0.339
##   P-value RMSEA <= 0.05                          0.238
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.017
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.104    0.195    5.659    0.000
##     fairelect60       1.065    0.154    6.923    0.000
##     efflegis60        1.119    0.159    7.044    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.376    0.591    2.329    0.020
##    .freeopp60         8.779    1.681    5.224    0.000
##    .fairelect60       4.487    0.959    4.676    0.000
##    .efflegis60        4.290    0.990    4.332    0.000
##     dem60             5.411    1.215    4.453    0.000
```

3.11 Interpreting the estimates

Let's see how do we interpret the parameter estimates part of the summary output.

By default, in the Estimate column the **Unstandardized estimates** are listed. The interpretation of these are as follows:

- In the **Latent variable loadings and Regressions** sections: the unstandardized estimates are interpreted as unstandardized **regression coefficients** (b) in linear models. So with 1 unit increase in the value of the independent variable (predictor), we can expect the Estimate unit change in the value of the outcome or dependent variable.
- In the **Covariances** section: the estimate is covariance of the two variables. Covariance is a measure of the joint variability of two variables. It is similar to correlation in the sense that if the two variables are correlated positively, the covariance is also positive. This unstandardized estimate is usually not interpreted directly beyond its sign (positive or negative), instead, the standardized estimate is interpreted (see below).
- The **Variances** section: contains the estimated **variances** and **residuals**. Residuals are marked with a "." before them, while variances (error in exogenous-only variables) do not have a . in front of their variable name in this section. The unstandardized residuals are interpreted as the variance of the residual error in the dependent variables in linear regression. The smaller this variance is the better the model fit, and the more accurate the model is at predicting the given variable. Variances show the variance of the exogenous-only variable.

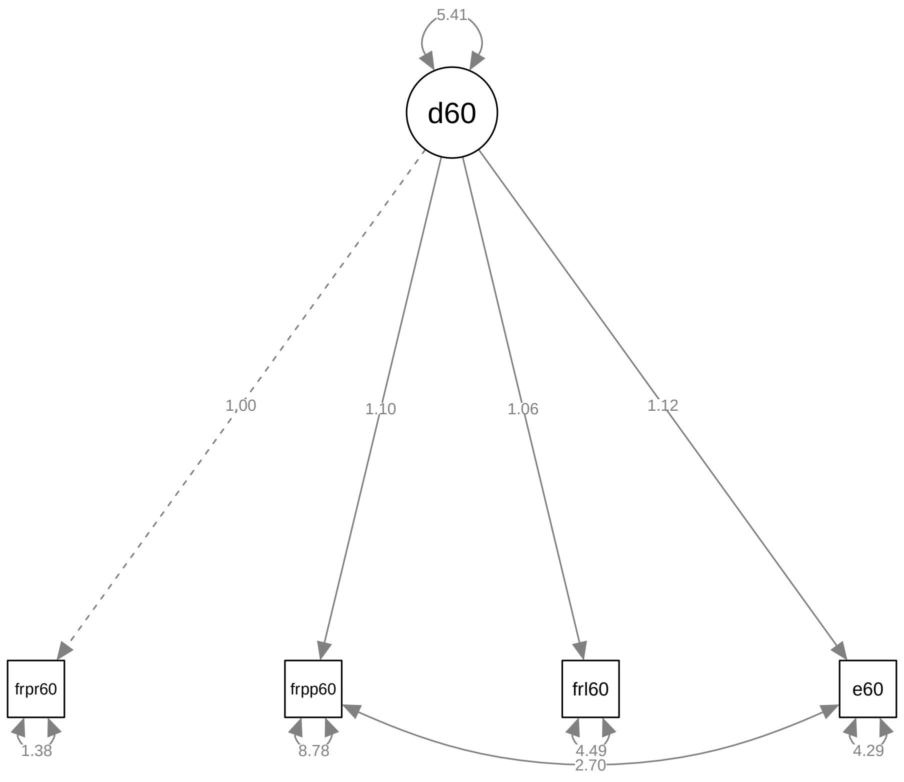
The unstandardized estimates can be printed on the **semPath diagram** as well using the whatLabels = "est" argument.

```
# unstandardized estimates
summary(fit3)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
```

```
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.104    0.195    5.659    0.000
##     fairelect60       1.065    0.154    6.923    0.000
##     efflegis60        1.119    0.159    7.044    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.376    0.591    2.329    0.020
##    .freeopp60         8.779    1.681    5.224    0.000
##    .fairelect60       4.487    0.959    4.676    0.000
##    .efflegis60        4.290    0.990    4.332    0.000
##     dem60             5.411    1.215    4.453    0.000
```

```
semPaths(fit3, whatLabels = "est")
```



We can ask for the **Standardized estimates** as well by using the standardized = T argument in the summary() function, and they will be listed in the Std.lv Std.all columns. We usually interpret the Std.all section, because it indicates the results when all variables (both latent and manifest) are standardized.

Alternatively, function **standardizedsolution()** can be used with the fit object, together with the type = "std.all" or type = "std.lv" arguments. The advantage of the standardizedsolution() function is that it also gives confidence intervals for the standardized estimates. Also, in purely path models (with no latent variables) this will provide the observed standardized variances and covariances of exogenous-only variables as well, which are hidden in the simple summary() output.

Note that in the standardized solution the **reference/marker/scaling path** is changed to the **variance of the exogenous variables**.

The interpretation of the numbers when using Std.all are as follows:

- In the **Latent variables** section: the standardized estimates are interpreted as factor loadings, which in most cases are correlation coefficients, so can range from -1 to +1.
- In the **Regressions** section: the standardized estimates show the standardized beta coefficients. This can be interpreted these are interpreted just like standardized beta coefficients in linear regression. Their values should be between -1 to +1.
- In the **Covariances** section: the standardized estimates are correlation coefficients. Their values should be between -1 to +1.
- In the **Variances** section: we can find the standardized version of the residual variances and the exogenous-only variances.
- It is often useful to also include the **rsquare = T** argument in the summary() function when wanting to interpret the standardized estimates. This will provide the so called squared multiple correlations (SMCs), which can be interpreted like the R^2 coefficient for endogenous variables. That is, we can see the percentage of variance of that particular endogenous variable explained in the model. This

can supplement our interpretation of the standardized residual variances. We do not get these for exogenous-only variables, since we are not trying to estimate them, so they not dependent variables in the model.

The standardized estimates can be printed on the **semPath diagram** as well using the whatLabels = "std" argument. One important role of the standardized estimates is that the strength of the paths can be compared to each other. So in our example we can tell that the factor loading of dem60 is strongest for freedom of press (0.89), and weakest for freedom of the political opposition (0.66).

```
# unstandardized estimates
summary(fit3, standardized = T, rsquare = T)
```

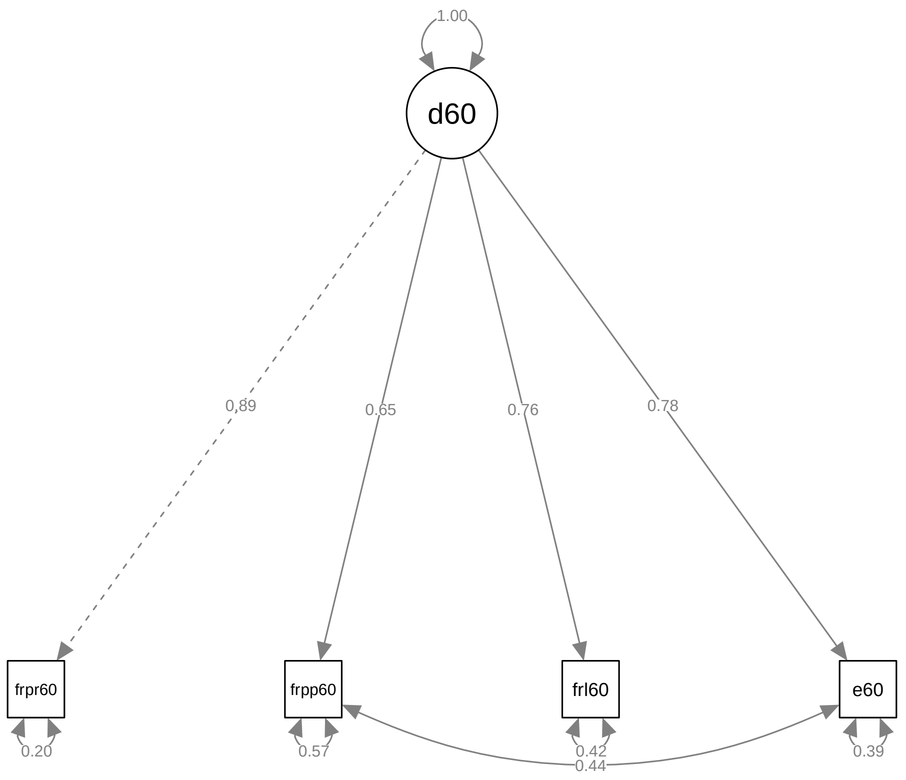
```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   dem60 =~
##     freepress60       1.000                               2.326    0.893
##     freeopp60         1.104    0.195    5.659    0.000    2.568    0.655
##     fairelect60       1.065    0.154    6.923    0.000    2.477    0.760
##     efflegis60        1.119    0.159    7.044    0.000    2.604    0.783
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010    2.696    0.439
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .freepress60       1.376    0.591    2.329    0.020    1.376    0.203
##    .freeopp60         8.779    1.681    5.224    0.000    8.779    0.571
##    .fairelect60       4.487    0.959    4.676    0.000    4.487    0.422
##    .efflegis60        4.290    0.990    4.332    0.000    4.290    0.388
##     dem60             5.411    1.215    4.453    0.000    1.000    1.000
##
```

```
## R-Square:
##                    Estimate
##     freepress60       0.797
##     freeopp60         0.429
##     fairelect60       0.578
##     efflegis60        0.612
```

```
standardizedsolution(fit3)
```

```
##            lhs op         rhs est.std    se      z pvalue ci.lower ci.upper
## 1        dem60 =~ freepress60   0.893 0.051 17.574  0.000    0.793    0.992
## 2        dem60 =~   freeopp60   0.655 0.079  8.286  0.000    0.500    0.810
## 3        dem60 =~ fairelect60   0.760 0.062 12.245  0.000    0.638    0.882
## 4        dem60 =~  efflegis60   0.783 0.060 12.981  0.000    0.664    0.901
## 5    freeopp60 ~~  efflegis60   0.439 0.114  3.868  0.000    0.217    0.662
## 6  freepress60 ~~ freepress60   0.203 0.091  2.235  0.025    0.025    0.381
## 7    freeopp60 ~~   freeopp60   0.571 0.104  5.517  0.000    0.368    0.774
## 8  fairelect60 ~~ fairelect60   0.422 0.094  4.478  0.000    0.238    0.607
## 9   efflegis60 ~~  efflegis60   0.388 0.094  4.107  0.000    0.203    0.572
## 10       dem60 ~~       dem60   1.000 0.000     NA     NA    1.000    1.000
```

```
semPaths(fit3, whatLabels = "std")
```



3.11.1 Getting estimates for fixed parameters (reference/marker/scaling variable)

One issue with fixed parameters is that the model summary does not produce the regular **parameter estimates** for them, and we do not get a p-value for their path either. There are a couple of solutions for this issue.

One solution is to simply **set another path as the reference parameter**. This way, we can look at the estimates of all parameters in the different models. For example in this model we set dem60 =~ fairelect60 as the reference path (fixed parameter) instead of dem60 =~ freepress60. This way we can get an estimate and p-value for the dem60 =~ freepress60, that we previously set to 1.

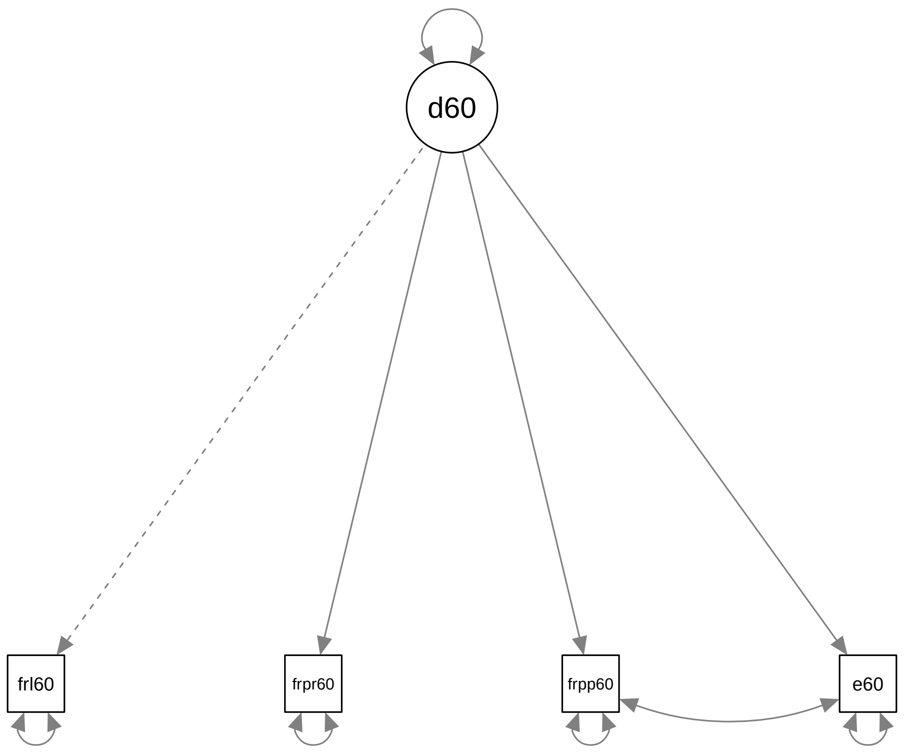
Note that in this solution the full-model chi-squared test statistics have not changed, however, the estimates are slightly changed for all parameters, because the scaling is changed.

```
model5 <-
"
    dem60 =~ fairelect60 + freepress60 + freeopp60 +  efflegis60
    freeopp60 ~~ efflegis60
"
fit5 <- sem(model5, data = my_data)

summary(fit5)
```

```
## lavaan 0.6-9 ended normally after 34 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     fairelect60       1.000
##     freepress60       0.939    0.136    6.923    0.000
##     freeopp60         1.037    0.191    5.435    0.000
##     efflegis60        1.051    0.159    6.626    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010
```

```
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .fairelect60       4.487    0.959    4.676    0.000
##    .freepress60       1.376    0.591    2.329    0.020
##    .freeopp60         8.779    1.681    5.224    0.000
##    .efflegis60        4.290    0.990    4.332    0.000
##     dem60             6.134    1.690    3.630    0.000
```

```
semPaths(fit5)
```



Another solution is to request the **standardized estimates** for the paths using the standardized = T argument in the summary function. In this solution lavaan automatically shifts the fixed parameter to the variance of the exogenous factors. This way, via the standardized estimates, we can get comparable estimates for all latent factor loadings in the same model. The standardized estimates are reported in two columns, in the Std.lv column the estimates are reported from a solution where only the latent variables are standardized, while in the Std.all column the estimates are reported rom a solution where all variables (latent and manifest) are standardized.

```
model3 <-
"
    dem60 =~  freepress60 + freeopp60 + fairelect60 + efflegis60
    freeopp60 ~~ efflegis60
"
fit3 <- sem(model3, data = my_data)
```

```
summary(fit3, standardized = T)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   dem60 =~
##     freepress60       1.000                               2.326    0.893
##     freeopp60         1.104    0.195    5.659    0.000    2.568    0.655
##     fairelect60       1.065    0.154    6.923    0.000    2.477    0.760
##     efflegis60        1.119    0.159    7.044    0.000    2.604    0.783
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010    2.696    0.439
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .freepress60       1.376    0.591    2.329    0.020    1.376    0.203
##    .freeopp60         8.779    1.681    5.224    0.000    8.779    0.571
##    .fairelect60       4.487    0.959    4.676    0.000    4.487    0.422
##    .efflegis60        4.290    0.990    4.332    0.000    4.290    0.388
##     dem60             5.411    1.215    4.453    0.000    1.000    1.000
```

3.12 Considering alternative models

SEM models can be compared to each other. As always when it comes to model selection, we should rely on theory and previous findings first. But in some cases multiple models might be theoretically plausible, so we may want to pit them against each other to see which one fits better with observed data.

As previously, the Akaike Information Criterion (AIC) can be used to compare SEM models, using the same rule as before: smaller AIC means better fit, and a difference in AIC of 2 or more means that the models can be considered to be significantly different in their model fit. Alternatively, the Bayesian Information Criteria

(BIC) and the Sample-size adjusted Bayesian Information Criteria (SABIC) is also used for such model comparison, with similar rules of comparison. These fit indexes can be optained in the model summary with the fit.measures = T argument.

ALternatively, if the models are nested, the Chi-squared-change-test and the CFI-change-test can be used to compare SEM models.

(It is less obvious, what can be called nested models among SEM models, but generally, the same rule applies as with model comparison of linear regression models: a simpler model is nested within a more complex model if all variables in the simpler model are also included in the more complex model. In CFA a one-factor solution is thought to be nested within a two-factor solution if the manifest variables are the same.)

The Chi-squared-change-test is produced by the anova() function, where if it is significant the model with the lower Chi-squared is preferred. The CFI change test simply means that if the difference in CFI is at least 0.01, the models are significantly different in model fit and the one with the higher CFI is preferred.

```
summary(fit3, fit.measures = T)
```

```
## lavaan 0.6-9 ended normally after 33 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 1.663
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.197
##
## Model Test Baseline Model:
##
##   Test statistic                               159.183
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.996
##   Tucker-Lewis Index (TLI)                       0.974
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -699.966
##   Loglikelihood unrestricted model (H1)       -699.135
##
##   Akaike (AIC)                                1417.933
##   Bayesian (BIC)                              1438.790
##   Sample-size adjusted Bayesian (BIC)         1410.424
##
## Root Mean Square Error of Approximation:
##
```

```
##   RMSEA                                          0.094
##   90 Percent confidence interval - lower         0.000
##   90 Percent confidence interval - upper         0.339
##   P-value RMSEA <= 0.05                          0.238
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.017
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.104    0.195    5.659    0.000
##     fairelect60       1.065    0.154    6.923    0.000
##     efflegis60        1.119    0.159    7.044    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##  .freeopp60 ~~
##    .efflegis60        2.696    1.050    2.568    0.010
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       1.376    0.591    2.329    0.020
##    .freeopp60         8.779    1.681    5.224    0.000
##    .fairelect60       4.487    0.959    4.676    0.000
##    .efflegis60        4.290    0.990    4.332    0.000
##     dem60             5.411    1.215    4.453    0.000
```

```r
model6 <-
"
    dem60 =~  freepress60 + freeopp60 + fairelect60 + efflegis60
"
fit6 <- sem(model6, data = my_data)

summary(fit6, fit.measures = T)
```

```
## lavaan 0.6-9 ended normally after 26 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         8
##
##   Number of observations                            75
##
## Model Test User Model:
##
```

```
##   Test statistic                                10.006
##   Degrees of freedom                                 2
##   P-value (Chi-square)                           0.007
##
## Model Test Baseline Model:
##
##   Test statistic                               159.183
##   Degrees of freedom                                 6
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.948
##   Tucker-Lewis Index (TLI)                       0.843
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -704.138
##   Loglikelihood unrestricted model (H1)       -699.135
##
##   Akaike (AIC)                                1424.275
##   Bayesian (BIC)                              1442.815
##   Sample-size adjusted Bayesian (BIC)         1417.601
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.231
##   90 Percent confidence interval - lower         0.103
##   90 Percent confidence interval - upper         0.382
##   P-value RMSEA <= 0.05                          0.014
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.046
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   dem60 =~
##     freepress60       1.000
##     freeopp60         1.404    0.197    7.119    0.000
##     fairelect60       1.089    0.167    6.529    0.000
##     efflegis60        1.370    0.167    8.228    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .freepress60       2.239    0.512    4.371    0.000
##    .freeopp60         6.412    1.293    4.960    0.000
##    .fairelect60       5.229    0.990    5.281    0.000
##    .efflegis60        2.530    0.765    3.306    0.001
##     dem60             4.548    1.106    4.112    0.000
```

```
anova(fit6, fit3)
```

```
## Chi-Squared Difference Test
##
##      Df    AIC    BIC   Chisq Chisq diff Df diff Pr(>Chisq)
## fit3  1 1417.9 1438.8  1.6632
## fit6  2 1424.3 1442.8 10.0059     8.3427       1   0.003872 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4 An example of a path analysis

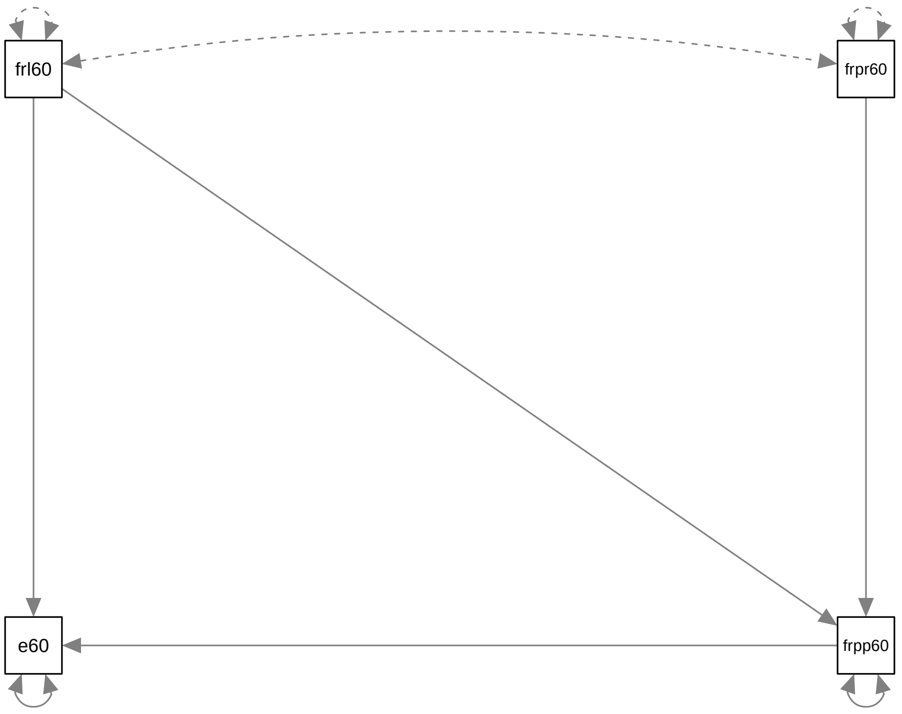
So far, we have been working with a simple CFA example. In this new example we will see an example for a path analysis, where no latent variables are involved.

In this example we will estimate the model parameters for a theoretical model explaining the effectiveness of the elected legislature. According to this model, the effectiveness of the elected legislature (efflegis60) is determined by the fairness of the elections (fairelect60) and the freedom of the political opposition (freeopp60) in the country. Furthermore, the model states that the freedom of the political opposition (freeopp60) is determined by the fairness of the elections (fairelect60) and the freedom of the press (freepress60), which in turn are correlated with each other. We can specify the model as follows:

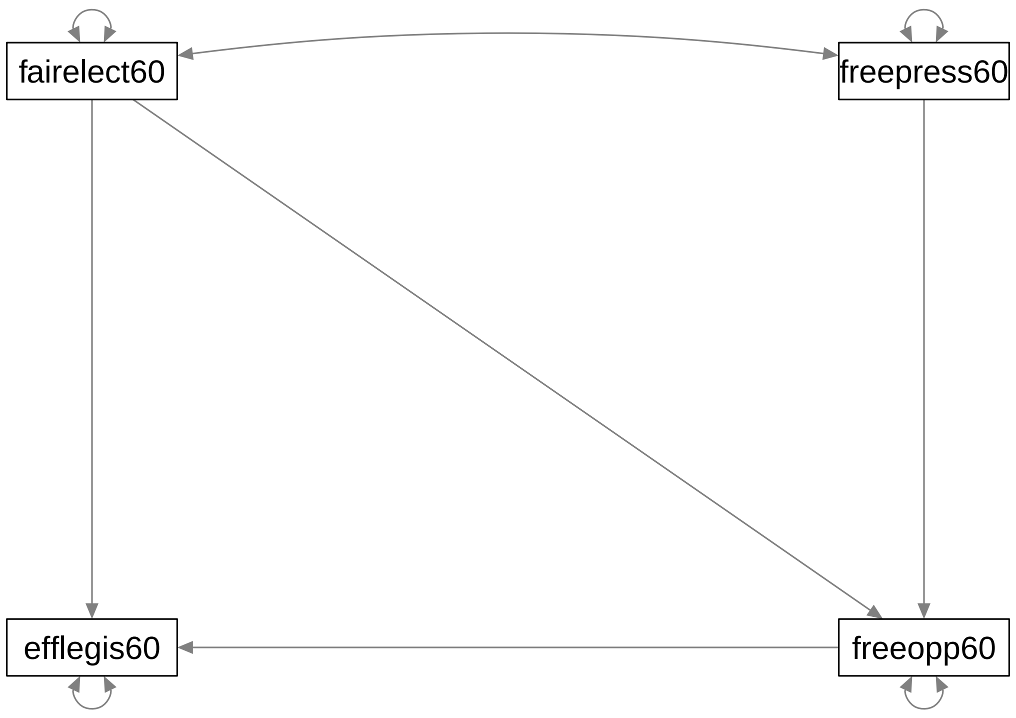
```
model_pathexample =
  "
  efflegis60 ~ fairelect60 + freeopp60
  freeopp60 ~ freepress60 + fairelect60
  "

fit_path_example = sem(model_pathexample, data = my_data)
```

As before, it is a good idea to visualize the model to make sure that the specification is correct. We can use the simple semPaths() function with our fit object, but it could look better. The variable names are small and abbreviated, and the paths do not really represent the true estimated paths. Below is an example if making the plot a bit more fancy and easier to interpret.

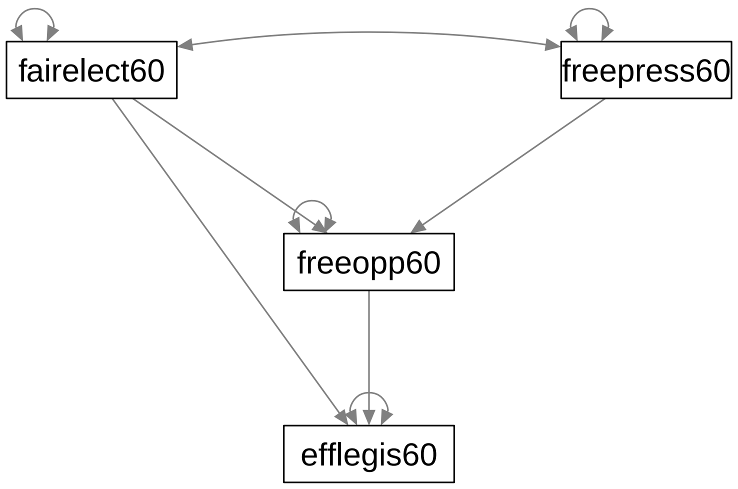
```
semPaths(fit_path_example)
```



```
plot_pathexample_fancy = semPaths(fit_path_example, fixedStyle = 1, label.scale=F, nCharNodes = 0,
          sizeMan2=5, sizeMan=15, asize=3)
```



```
indicator_order_pathexample  <- c("freepress60", "freeopp60", "fairelect60", "efflegis60")
indicator_factor_pathexample <- c("freepress60","freeopp60","fairelect60","efflegis60")

factor_layout_pathexample <- layout_matrix(freepress60 = c(1,3),
                               fairelect60 = c(1,1),
                               freeopp60 = c(2,2),
                               efflegis60 = c(3,2))

factor_point_to_pathexample <- layout_matrix(down = c(1,3),
                                 down = c(1,1),
                                 down = c(2,2),
                                 down = c(3,2))

p2_pathexample <- set_sem_layout(plot_pathexample_fancy,
                     indicator_order = indicator_order_pathexample,
                     indicator_factor = indicator_factor_pathexample,
                     factor_layout = factor_layout_pathexample,
                     factor_point_to = factor_point_to_pathexample)
plot(p2_pathexample)
```



Before we look at the summary, lets count the degrees of freedom of this model:

4 manifest variables mean that the null model has 4*(4+1)/2 = 10 possible parameters.

The free parameters in the model are (9 in total):

- 4 regression paths (single headed arrows)
- 1 covariance (double headed arrow between fairelect60 and freepress60)
- 2 residuals (of the endogenous variables freeopp60 and efflegis60)
- 2 variances (of the exogenous variables fairelect60 and freepress60)

df = 10 - 9 = 1

After we made sure that the model is really what we wanted to specify, we can get the model estimates and fit statistics using the summary() function. First we can verify that the df is 1 as we expected, however, the list of estimates only contain the estimates for 6 paths: the regressions and the errors. So the estimates of the covariance and the variances are missing from the summary.

The reason for this is as stated above in the advanced note in the Counting degrees of freedom section. For **observed exogenous variables** we do not need to estimate **variance and covariance** with another observed variable, because **we can directly observe** these in the data. So they are not estimated by the model, but they are counted as "free parameteres" when counting degrees of freedom, because they do depend on the data and are not set to a fixed value by us.

We can see the **full list of free parameters** in the parTable() output, and the estimates (and observed variance and covariance) can be found in the semPaths() if we specify the , whatLabels = "est" argument. We can also get the full list of parameters (both estimated and observed) by using the parameterEstimates() function. Note that the variances and the covariance have no standard error, because they are not estimated, they are just included in the model as observed in the raw data.

Based on the estimates it seems that all of the theorized relationships are meaningful in this model except that fairelect60 does not seem to have a substantial influence on the freedom of the political opposition.

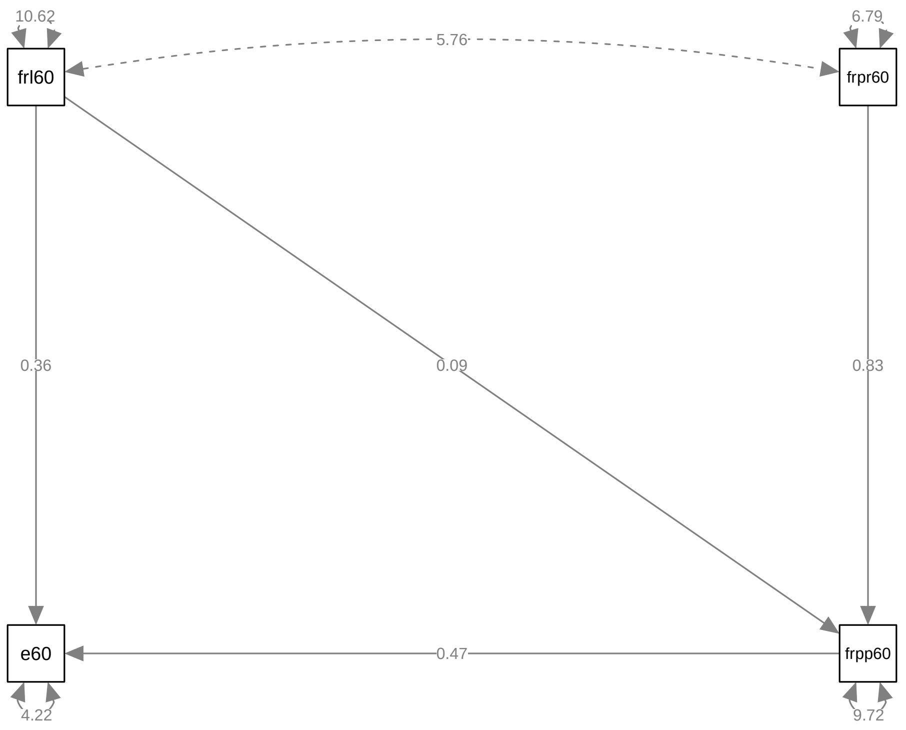
```r
summary(fit_path_example, fit.measures = T, standardized = T)
```

```
## lavaan 0.6-9 ended normally after 17 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         6
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 6.199
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.013
##
## Model Test Baseline Model:
##
##   Test statistic                               112.908
##   Degrees of freedom                                 5
##   P-value                                        0.000
##
## User Model versus Baseline Model:
##
##   Comparative Fit Index (CFI)                    0.952
##   Tucker-Lewis Index (TLI)                       0.759
##
## Loglikelihood and Information Criteria:
##
##   Loglikelihood user model (H0)               -352.113
##   Loglikelihood unrestricted model (H1)       -349.014
##
##   Akaike (AIC)                                 716.227
##   Bayesian (BIC)                               730.132
##   Sample-size adjusted Bayesian (BIC)          711.221
##
## Root Mean Square Error of Approximation:
##
##   RMSEA                                          0.263
##   90 Percent confidence interval - lower         0.097
##   90 Percent confidence interval - upper         0.477
##   P-value RMSEA <= 0.05                          0.022
##
## Standardized Root Mean Square Residual:
##
##   SRMR                                           0.036
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   efflegis60 ~
##     fairelect60       0.364    0.082    4.468    0.000    0.364    0.357
##     freeopp60         0.474    0.068    6.990    0.000    0.474    0.558
##   freeopp60 ~
##     freepress60       0.831    0.188    4.418    0.000    0.831    0.552
##     fairelect60       0.092    0.150    0.609    0.542    0.092    0.076
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .efflegis60        4.220    0.689    6.124    0.000    4.220    0.381
##    .freeopp60         9.718    1.587    6.124    0.000    9.718    0.632
```

```
parTable(fit_path_example)
```

```
##   id         lhs op         rhs user block group free ustart exo label plabel
## 1  1  efflegis60  ~ fairelect60    1     1     1    1     NA   0         .p1.
## 2  2  efflegis60  ~   freeopp60    1     1     1    2     NA   0         .p2.
## 3  3   freeopp60  ~ freepress60    1     1     1    3     NA   0         .p3.
## 4  4   freeopp60  ~ fairelect60    1     1     1    4     NA   0         .p4.
## 5  5  efflegis60 ~~  efflegis60    0     1     1    5     NA   0         .p5.
## 6  6   freeopp60 ~~   freeopp60    0     1     1    6     NA   0         .p6.
## 7  7 fairelect60 ~~ fairelect60    0     1     1    0     NA   1         .p7.
## 8  8 fairelect60 ~~ freepress60    0     1     1    0     NA   1         .p8.
## 9  9 freepress60 ~~ freepress60    0     1     1    0     NA   1         .p9.
##    start    est    se
## 1  0.000  0.364 0.082
## 2  0.000  0.474 0.068
## 3  0.000  0.831 0.188
## 4  0.000  0.092 0.150
## 5  5.535  4.220 0.689
## 6  7.686  9.718 1.587
## 7 10.621 10.621 0.000
## 8  5.761  5.761 0.000
## 9  6.787  6.787 0.000
```

```
semPaths(fit_path_example, whatLabels = "est")
```



```r
parameterEstimates(fit_path_example)
```

```
##           lhs op         rhs    est    se     z pvalue ci.lower ci.upper
## 1  efflegis60  ~ fairelect60  0.364 0.082 4.468  0.000    0.204    0.524
## 2  efflegis60  ~   freeopp60  0.474 0.068 6.990  0.000    0.341    0.607
## 3   freeopp60  ~ freepress60  0.831 0.188 4.418  0.000    0.462    1.200
## 4   freeopp60  ~ fairelect60  0.092 0.150 0.609  0.542   -0.203    0.386
## 5  efflegis60 ~~  efflegis60  4.220 0.689 6.124  0.000    2.870    5.571
## 6   freeopp60 ~~   freeopp60  9.718 1.587 6.124  0.000    6.608   12.829
## 7 fairelect60 ~~ fairelect60 10.621 0.000    NA     NA   10.621   10.621
## 8 fairelect60 ~~ freepress60  5.761 0.000    NA     NA    5.761    5.761
## 9 freepress60 ~~ freepress60  6.787 0.000    NA     NA    6.787    6.787
```

```r
standardizedsolution(fit_path_example, type = "std.all")
```

```
##           lhs op         rhs est.std    se     z pvalue ci.lower ci.upper
## 1  efflegis60  ~ fairelect60   0.357 0.076 4.693  0.000    0.208    0.506
## 2  efflegis60  ~   freeopp60   0.558 0.072 7.799  0.000    0.418    0.699
## 3   freeopp60  ~ freepress60   0.552 0.111 4.991  0.000    0.335    0.769
## 4   freeopp60  ~ fairelect60   0.076 0.125 0.611  0.541   -0.168    0.321
## 5  efflegis60 ~~  efflegis60   0.381 0.064 5.949  0.000    0.256    0.507
## 6   freeopp60 ~~   freeopp60   0.632 0.080 7.904  0.000    0.475    0.789
## 7 fairelect60 ~~ fairelect60   1.000 0.000    NA     NA    1.000    1.000
## 8 fairelect60 ~~ freepress60   0.679 0.000    NA     NA    0.679    0.679
## 9 freepress60 ~~ freepress60   1.000 0.000    NA     NA    1.000    1.000
```

5 Example of Mediation analysis

You can also conduct a classical mediation analysis using lavaan, where you calculate the direct, the indirect, and the total effect of an independent variable on an outcome variable, where part of its effect is mediated by another variable.

The trick is that we can give "labels" to paths by sung the "*" in the model specification, and we can ask lavaan to compute specific derived parameters, such as axb, wich refers to the mediated (indirect) effect of the fairness of the elections on the effectiveness of the elected legislature mediated through the freedom of the opposition, such as c + (axb), which stands for the total effect of the fairness of the elections on the effectiveness of the elected legislature (direct and indirect effects combined).

```
model_mediation_example =
  "
  efflegis60 ~ c*fairelect60 + b*freeopp60
  freeopp60 ~ freepress60 + a*fairelect60

  # indirect effect (a*b)
             indirect := a*b

  # total effect
             total := c + (a*b)

  "

fit_mediation_example = sem(model_mediation_example, data = my_data)
```

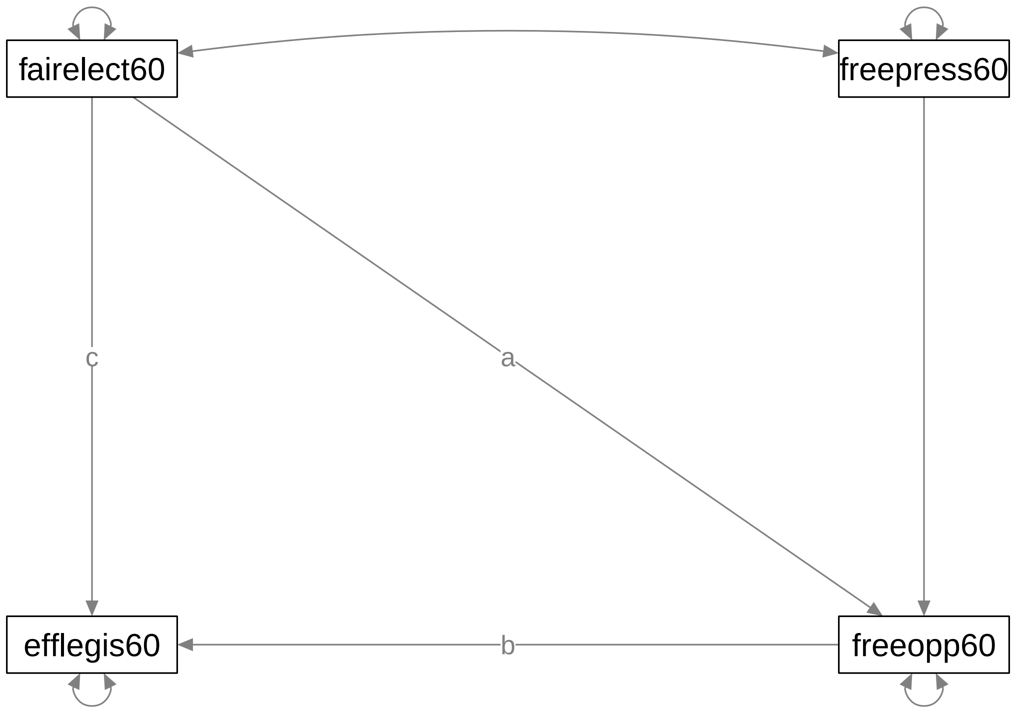
The results of the analysis indicate that indirect effect of fairelect60 on efflegis60 mediated through freeopp60 is not significant, so the model does not support a mediation effect.

```
summary(fit_mediation_example)
```

```
## lavaan 0.6-9 ended normally after 17 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         6
##
##   Number of observations                            75
##
## Model Test User Model:
##
##   Test statistic                                 6.199
##   Degrees of freedom                                 1
##   P-value (Chi-square)                           0.013
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
```

```
##                       Estimate  Std.Err  z-value  P(>|z|)
##   efflegis60 ~
##     fairelct60 (c)     0.364    0.082    4.468    0.000
##     freeopp60  (b)     0.474    0.068    6.990    0.000
##   freeopp60 ~
##     freeprss60         0.831    0.188    4.418    0.000
##     fairelct60 (a)     0.092    0.150    0.609    0.542
##
## Variances:
##                       Estimate  Std.Err  z-value  P(>|z|)
##    .efflegis60         4.220    0.689    6.124    0.000
##    .freeopp60          9.718    1.587    6.124    0.000
##
## Defined Parameters:
##                       Estimate  Std.Err  z-value  P(>|z|)
##     indirect           0.043    0.072    0.607    0.544
##     total              0.408    0.106    3.834    0.000
```

```
semPaths(fit_mediation_example, fixedStyle = 1, label.scale=F, nCharNodes = 0,
         sizeMan2=5, sizeMan=15, asize=3, edge.label.cex = 1)
```



5.0.1 What to report

As SEM analysis is used for many different goals, it is hard to give exact guidance about what to report. Nevertheless here are some general guidlines about reporting the results:

When reporting the model specification, report the model in full detail, specifying not only all paths in the model, but also which parameters were fixed. This might be aided with a path diagram.

When reporting model fit, report multiple fit indexes including Chi-squared test statistics, CFI, TLI, and RMSEA point (for RMSEA not only point estimate but also confidence interval).

Report the path diagram including parameter values and estimates. Furthermore, report in a table format the parameter estimates, both unstandardized and standardized, with standard errors, p-values and/or confidence intervals.

Report the correlation matrix or covariance matrix across observed variables along the analysis code, which will allow others to reporoduce the model.

6 Example of full structural SEM

Practice

Now that we know all of the above, we can see how to specify, fit, and assess the full theoretical model of the authors in the original paper.

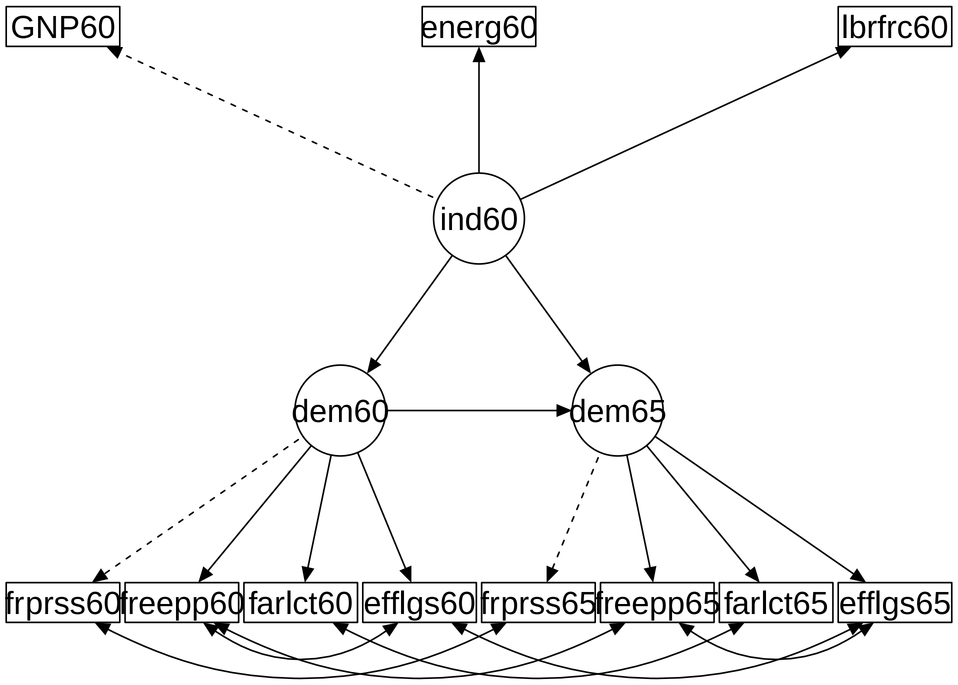
To revisit, the theory of the authors proposes that freedom of the press, freedom of political opposition, fairness of elections, and effectiveness of the elected legislature are determined by the overall political democracy in a country (a latent factor), and this political democracy is determined by how industrialized the country is (another latent factor), which in turn also affects (or manifests in) GNP, inanimate energy consumption, and percentage of the labor force in industry. Adding some complexity is that democracy-related variables are measured in 1960 and 1965, so the model also incorporates time: it proposes two separate latent factors: political democracy in 1960 and political democracy in 1965, where political democracy in 1965 is influenced by political democracy in 1960, and the same observed measures taken in 1960 and 1965 co-vary, incorporating common measurement error in these measures that are not related to political democracy, rather, to the way the data was collected and measured.

We can specify this model as follows:

```r
model_full <- '
  # measurement model
    ind60 =~ GNP60 + energ60 + laborforce60
    dem60 =~ freepress60 + freeopp60 + fairelect60 + efflegis60
    dem65 =~freepress65 + freeopp65 + fairelect65 + efflegis65
  # regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60
  # residual correlations
    freepress60 ~~freepress65
    freeopp60 ~~ efflegis60 + freeopp65
    fairelect60 ~~ fairelect65
    efflegis60 ~~ efflegis65
    freeopp65 ~~ efflegis65
'


fit_full <- sem(model_full, data = my_data)
```

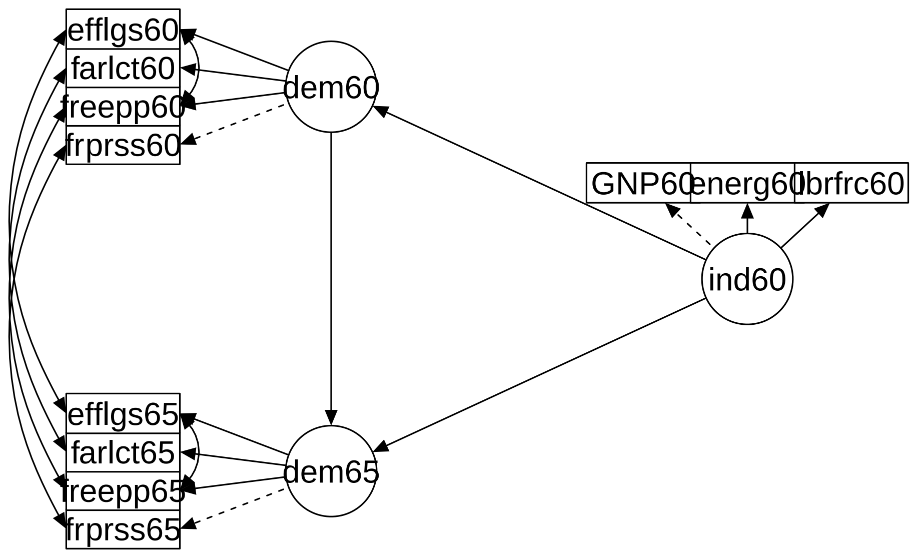
Here are two possible plots of the same path diagram:

```
plot_full = semPaths(fit_full, label.scale=F, nCharNodes = 8,
         sizeMan2=3.5, sizeMan=10, asize=3, edge.color="black", residuals = F)
```



```
plot_full$graphAttributes$Edges$edgeConnectPoints[c(4:11, 16, 20),2] <- 0.5 * pi
plot_full$graphAttributes$Edges$edgeConnectPoints[c(16, 20),1] <- 0.5 * pi
plot_full$graphAttributes$Edges$curve[c(16, 20)] <- -2
plot_full$graphAttributes$Edges$curve[c(15, 17:19, 21)] <- -3
plot_full$graphAttributes$Edges$edgeConnectPoints[c(15, 17:19, 21:26),2] <- 1.5 * pi
plot_full$graphAttributes$Edges$edgeConnectPoints[c(15, 17:19, 21:26),1] <- 1.5 * pi


indicator_order  <- c("GNP60", "energ60", "laborforce60",
                      "freepress60", "freeopp60", "fairelect60", "efflegis60",
                      "freepress65", "freeopp65", "fairelect65", "efflegis65")

indicator_factor <- c( "ind60",  "ind60",  "ind60",
                       "dem60",  "dem60",  "dem60", "dem60",
                       "dem65",  "dem65",  "dem65",  "dem65")


factor_layout <- layout_matrix(dem60 = c(1,1),
                               ind60 = c(2,2),
                               dem65 = c(3,1))

factor_point_to <- layout_matrix(left = c(1,1),
```

```r
                               up = c(2,2),
                               left = c(3,1))


p2_full <- set_sem_layout(plot_full,
                          indicator_order = indicator_order,
                          indicator_factor = indicator_factor,
                          factor_layout = factor_layout,
                          factor_point_to = factor_point_to)
plot(p2_full)
```



Practice tasks:

- Calculate the degrees of freedom of the model based on the path diagram and what we learned above.
- Test the assumption of multivariate normality, and decide whether to use the robust or the regular estimates.
- Look at model fit indexes CFI, TLI, and RMSEA, and make a decision about whether this model is a good fit with the data
- Run the model summary in a way so that that you can see both he unstandardized and the standardized estimates.
- Based on the estimates, what has more influence on the level of political democracy in 1965, the level of industrialization in 1960 or the level of political democracy in in 1960?
- Create a plot with the standardized model parameters highlighted on the diagram.