Table of Contents

Introduction

1. Exploratory Factor Analysis (EFA)

EFA is a statistical technique used in data analysis to uncover the underlying structure of a set of observed variables. The main goal of EFA is to understand the relationships among a large number of variables by reducing them into a smaller set of underlying factors. These factors represent the shared variance among the observed variables and can provide insights into the underlying constructs or dimensions being measured.

2. Confirmatory Factor Analysis (CFA)

CFA is a subset of SEM that focuses on evaluating the measurement properties of latent constructs or factors. It is primarily used to assess the validity and reliability of a measurement instrument or questionnaire. CFA allows researchers to test whether the observed variables (indicators) in a dataset adequately reflect the underlying latent constructs they are intended to measure. CFA examines the relationship between the **observed variables** and the **latent factors** and estimates **factor loadings**, which indicates the strength of the relationship between the observed variables and the latent factors.

3. Structural Equation Modeling (SEM)

SEM is a more comprehensive statistical framework that encompasses CFA but goes beyond it. SEM includes not only the measurement model (CFA) but also a structural model that examines the relationships between latent variables (factors) and other observed variables. In other words, SEM allows researchers to simultaneously test the measurement model and the structural model, thereby examining the relationships and causal pathways between variables.

Models such as linear regression, multivariate regression, path analysis, confirmatory factor analysis, and structural regression can be thought of as special cases of SEM. The following relationships are possible in SEM:

- Observed to observed variables (e.g., regression)
- Latent to observed variables (e.g., confirmatory factor analysis)
- Latent to latent variables (e.g., structural regression)

SEM uniquely encompasses both measurement and structural models. The measurement model relates observed to latent variables and the structural model relates latent to latent variables. Various software programs currently handle SEM models including Mplus, EQS, SAS PROC CALIS, Stata's sem and more recently, R's lavaan. **The benefit of lavaan is that it is open source, freely available, and is relatively easy to use.**

Requirements

- Make sure you have *R* and *RStudio* installed.
- Make sure to have the following R packages installed, and if not, run these commands in RStudio: (only needs to be done once)

```
install.packages("readxl ")                          ## Reads excel files
install.packages("psych ")                           ## To perform EFA
install.packages("REdaS ")                           ## To perform KMO and Bartletts test
install.packages("GPArotation ")                     ## For Rotation
install.packages("paran ")                           ## For Parallel Analysis
install.packages("lavaan", dependencies = TRUE)      ## Tutorial for Lavaan
install.packages("lavaanPlot ")                      ## Tutorial for Plotting
install.packages("tidySEM")
```

- Once you've installed the packages, you can load them via the following:

```
library(readxl)
library(psych)
library(REdaS)
library(GPArotation)
library(paran)
library(lavaan)
library(lavaanPlot)
library(tidySEM)
```

- Try the following examples by running the following

```
example(fa)
example(paran)
example(cfa)
example(sem)
example(modindices)
example(lavaanPlot)
```

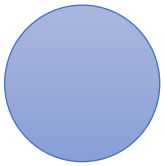
Loading the Data

- Load the file that contains he data directly into R with the following command:
  - dat <- read.csv("/Path") **or** dat <- read_excel("/Path")

Correlation Matrix & Sample Variance-Covariance Matrix

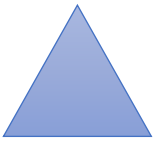
- The function **cor** specifies that we want to obtain the correlation matrix.
  - **cor(dat)**
- The function **cov** specifies that we want to obtain the covariance matrix S from the data.
  - **cov(dat)**

The Path Diagram



Latent Variable

Observed Variable



Intercept

Path

Variance or Covariance

Quick Reference of Lavaan Syntax

- **~ predict**: used for regression of observed outcome to observed predictors (e.g., y ~ x)
- **=~ indicator:** used for latent variable to observed indicator in factor analysis measurement models (e.g., f =~ q + r + s)
- **~~ covariance** (e.g., x ~~ x)
- **~1** intercept or mean (e.g., x ~ 1 estimates the mean of variable x)
- **1*** fixes parameter or loading to one or zero (e.g., f =~ 1*q, f1 ~~ 0*f2)
- **NA*** frees parameter or loading (useful to override default marker method, like: f =~ NA*q)
- **a* labels** the parameter ‘a’, used for model constraints

Exploratory Factor Analysis (EFA)

When conducting a survey, it is important to determine if the items in the survey exhibit similar response patterns and if they can be grouped together to form a larger construct. This is where Exploratory Factor Analysis (EFA) comes into play. EFA assumes that for a set of observed variables, there exist underlying variables known as factors that explain the relationships among these variables. By applying EFA, we aim to identify and understand these latent factors that drive the observed correlations among the variables. Ultimately, EFA allows us to uncover the hidden structure within the data, revealing the underlying dimensions or constructs that contribute to the observed responses.

```
## Load & View the Data

ATGC <- read_excel("Path to Data File/Data.xlsx")
View(ATGC)
```

The **paran()** function is used to conduct parallel analysis to determine the number of factors to retain. Parallel analysis is a statistical technique that compares the observed eigenvalues of the correlation matrix with the randomly generated eigenvalues to determine the number of meaningful factors.

```
## When cfa = F, the paran() function performs a principal component parallel analysis (PCA)

paran(ATGC, cfa = T)

## Optional Plotting

paran(ATGC, cfa = T, graph = T, color=T, col = c("black", "red", "blue"))

## According to parallel analysis, we need 3 factors

fa(ATGC, nfactors = 3, rotate = "oblimin" )

## Save the analysis as the object m1

M1<-fa(ATGC, nfactors = 3, rotate = "oblimin" )

## Produce a figure with a Title. Note: fa.diagram still works for PCA

fa.diagram(M1, main="Title")
```

Confirmatory Factor Analysis (CFA) / Measurement Model

Confirmatory factor analysis borrows many of the same concepts from exploratory factor analysis except that instead of letting the data tell us the factor structure, we pre-determine the factor structure and verify the psychometric structure of a previously developed scale.

In order to identify a factor in a CFA model with three or more items, there are two options known respectively as the marker method and the variance standardization method.

- **Marker method** fixes the ***first*** loading of each factor to 1.
- **Variance standardization method** fixes the variance of each factor to 1 but freely estimates all loadings.

1. One Factor Confirmatory Factor Analysis

Technically a three item CFA is the minimum number of items for a one factor CFA as this results in a ***saturated*** model where the number of free parameters equals to number of elements in the variance-covariance matrix (i.e., the degrees of freedom is zero).

```
## One factor three items, default marker method
m1a  <- ' f  =~ q03 + q04 + q05'
onefac3items_a <- cfa(m1a, data=dat)
summary(onefac3items_a)
```

By default, **lavaan** chooses the marker method if nothing else is specified. In order to free a parameter, put NA* in front of the parameter to be freed, to fix a parameter to 1, put 1* in front of the parameter to be fixed. The syntax NA*q03 frees the loading of the first item because by default marker method fixes it to one, and f ~~ 1*f means to fix the variance of the factor to one.

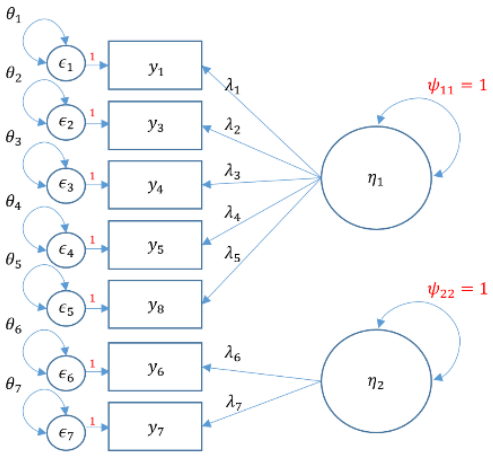
```
## One factor three items, variance std
m1b  <- ' f =~ NA*q03 + q04 + q05
        f ~~ 1*f '
onefac3items_b <- cfa(m1b, data=dat)
summary(onefac3items_b)

## Alternatively you can use std.lv=TRUE and obtain the same results
onefac3items_a <- cfa(m1a, data=dat, std.lv=TRUE)
summary(onefac3items_a)
```

To better interpret the factor loadings, often times you would request the standardized solutions. Going back to our orginal marker method object onefac3items_a we request the summary but also specify that **standardized=T**.

```
summary(onefac3items_a, standardized = T)
```

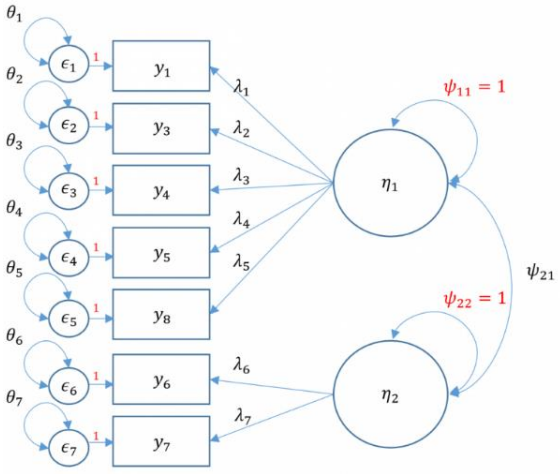
2. Two Factor Confirmatory Factor Analysis

a. Uncorrelated Factors




```
## Uncorrelated two factor solution, var std method

m4a <- 'f1 =~ q01+ q03 + q04 + q05 + q08
      f2 =~ a*q06 + a*q07
      f1 ~~ 0*f2 '
twofac7items_a <- cfa(m4a, data = dat, std.lv = T, standardized = T, ci = T, fit.measures = T)

## Standardised = T gives beta values (std.all column), ci = T gives 95% CIs
```

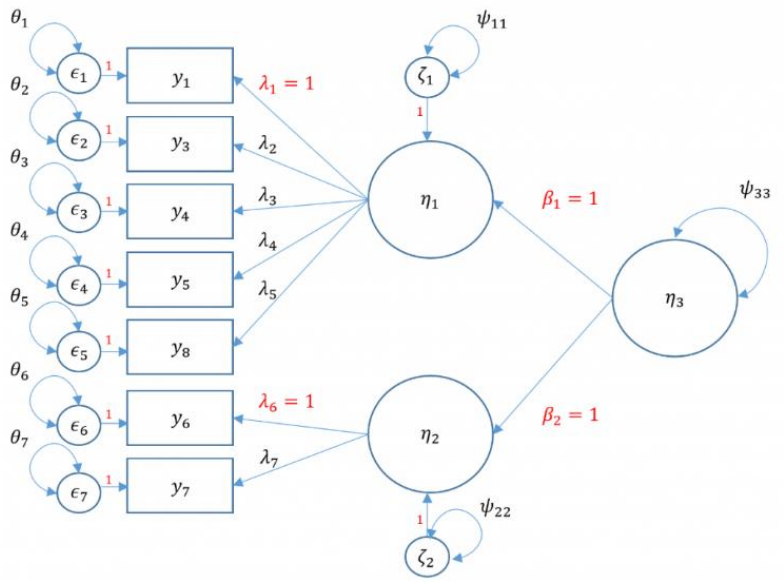
b. Correlated Factors




```
## Correlated two factor solution, marker method

m4b <- 'f1 =~ q01+ q03 + q04 + q05 + q08
      f2 =~ q06 + q07'
twofac7items_b <- cfa(m4b, data=dat, std.lv = T)
summary(twofac7items_b, fit.measures = T, standardized = T)
```

3. Second-Order CFA

```
## Second order three factor solution, marker method
m5a <- 'f1 =~ q01+ q03 + q04 + q05 + q08
      f2 =~ q06 + q07
      f3 =~ 1*f1 + 1*f2
      f3 ~~ f3'
secondorder <- cfa(m5a, data=dat)
summary(secondorder,fit.measures=TRUE,standardized=TRUE)
```



Note that there is no perfect way to specify a second order factor when you only have two first order factors. You either have to assume the variance standardization method assumes that the residual variance of the two first order factors is one which means that you assume homogeneous residual variance.

The marker method assumes that both loadings from the second order factor to the first factor is 1. To make sure you fit an equivalent method though, the degrees of freedom for the User model must be the same.

NOTE: changing the standardization method *should not* change the degrees of freedom and chi-square value. If you standardize it one way and get a different degrees of freedom, then you have identified it incorrectly. Even though the chi-square fit is the same however, you will get different standardized variances and loadings depending on the assumptions you make (to set the loadings to 1 for the two first order factors and freely estimate the variance or to freely estimate but equate the loadings and set the residual variance of the first order factors to 1).

CFA distinguishes itself from EFA as a method to assess the credibility of a previously defined hypothesis.

The observed indicators serve as measures of the unobserved construct or factor. A just identified model for a one-factor model has exactly three indicators, but some researchers require only two indicators per factor due to resource restrictions; however having more than three items per factor is ideal because it allows degrees of freedom which leads to measures of fit. Finally, if the fit indicates poor fit for a one-factor model, a two-factor model may be more appropriate, that the items measure not just one construct, and that there may be underlying correlation between the two constructors or factor. However, if theory is that the correlation between these two constructs is caused by a third factor, then these two first-order factors can serve as latent indicators of the underlying second order factor. However if the correlations between factors are represented as regression paths, then we move beyond the scope of CFA into what is known as s***tructural equation modeling***.

Structural Equation Modeling (SEM)

1. Simple Regression

Simple regression models the relationship of an observed exogenous variable on a single observed endogenous variable. For a single subject, the simple linear regression equation is most commonly defined as:

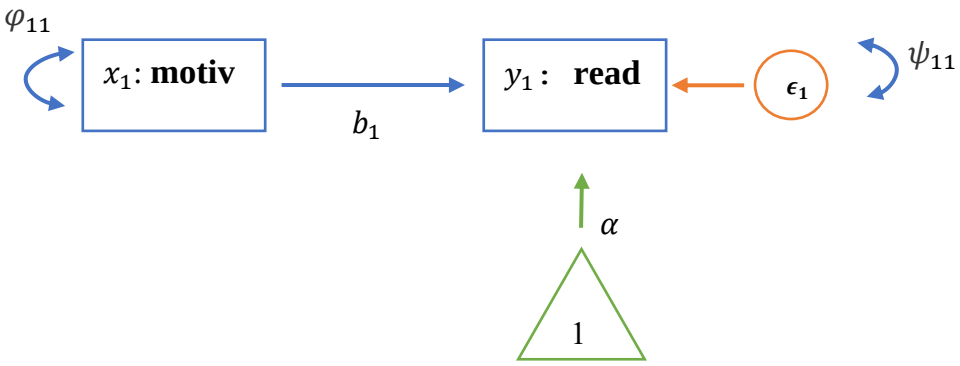
$$y_1 = \alpha + b_1 x_1 + \epsilon_1$$

Where α is the intercept, b_1 is the coefficient, x is an observed predictor, and ϵ is the residual.

φ: variance or covariance of the exogenous variable.

ψ: residual variance or covariance of the endogenous variable.

To see the matrix visually, we can use a path diagram (Simple Regression):



In R, the most basic way to run a linear regression is to use the lm() function which is available in base R.

```
# Simple regression using lm()
m1a <- lm(read ~ motiv, data=dat)
```

We can run the equivalent code in lavaan. The syntax is very similar to lm() in that **read ~ motiv** specifies the predictor **motiv** on the outcome read. However, by default the intercept is not included in the output but is implied. If we want to add an intercept, we need to include **read ~ 1 + motiv**. Optionally, you can request the variance of **motiv using motiv ~~ motiv**.

```
# Simple regression using lavaan
m1b <-  ' # regressions
          read ~ 1 + motiv
          # variance (optional)
          motiv ~~ motiv'

fit1b <- sem(m1b, data=dat)
summary(fit1b)
```

2. Multiple Regression

Simple regression is limited to just a single exogenous variable. In practice, a researcher may be interested in how a group of exogenous variables predict an outcome. Suppose we still have one endogenous outcome but two exogenous predictors; this is known as multiple regression. Matrix form allows us to concisely represent the equation for all observations:

$$y_1 = \alpha + \boldsymbol{b}\mathbf{x} + \epsilon_1$$

where α is the intercept for y_1, $\boldsymbol{b}$ is a vector (q×1) of regression coefficients where q is the total number of exogenous variables, **x** is a vector (1×q) of exogenous variables, and ϵ is the residual of y_1.

φ: variance or covariance of the exogenous variable.

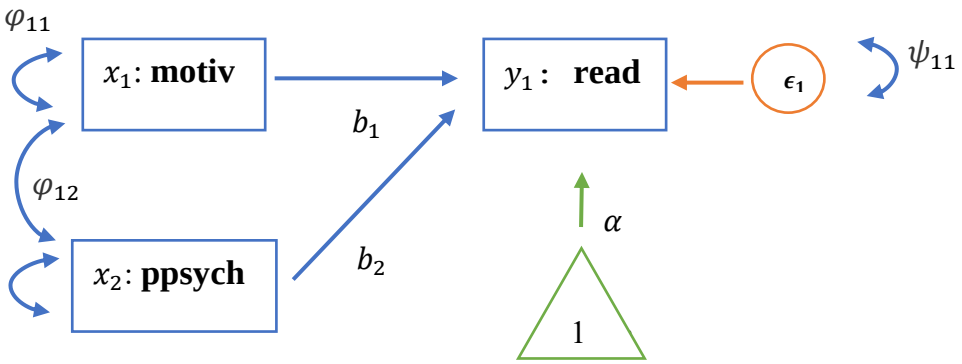
ψ: residual variance or covariance of the endogenous variable.

Assumptions

E (ϵ_1)=0 the mean of the residuals is zero

ϵ_1 is uncorrelated with **x**

Suppose we have two exogenous variables x_1, x_2 predicting a single endogenous variable x_1. The path diagram for this multiple regression is:



Specifying a multiple regression in lavaan is as easy as adding another predictor. Suppose the researcher is interested in how Negative Parental Psychology **ppsych** and Motivation **motiv** to predict student readings scores read.

```
# Multiple Regression
m2 <- ' # regressions
        read ~ 1 + ppsych + motiv
        # covariance
        ppsych ~~ motiv'

fit2 <- sem(m2, data=dat)
summary(fit2)
```

3. Multivariate Regression

a. Multivariate regression with default covariance

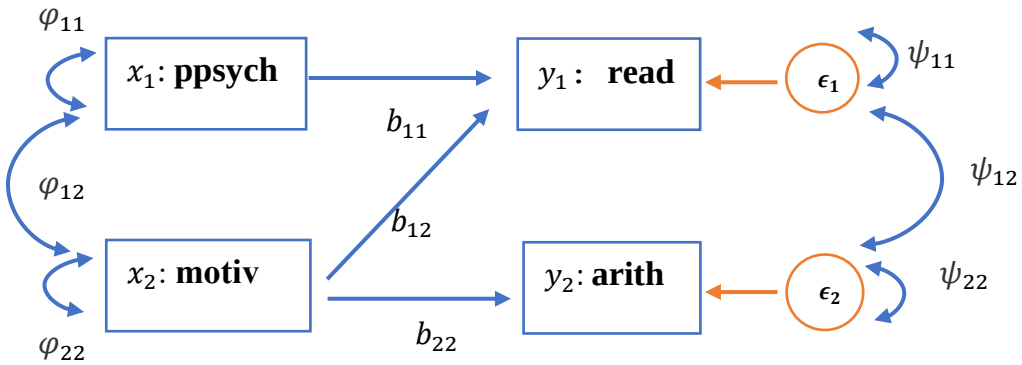
Simple and multiple regression model one outcome (y) at a time. In multivariate or simultaneous linear regression, multiple outcomes $y_1, y_2,\ldots, y_p$ are modeled simultaneously, where q is the number of outcomes. The General Multivariate Linear Model is defined as

$$\mathbf{y} = \alpha + \mathbf{bx} + \epsilon$$

To see the matrix formulation more clearly, consider two (i.e., bivariate) endogenous variables y_1, y_2 predicted by two exogenous predictors x_1, x_2.

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix}$$

Due to the added complexity of this and the following multivariate models, we will explicitly model intercepts in lavaan code but exclude them from the path diagrams. With that said, the path diagram for a multivariate regression model is depicted as:



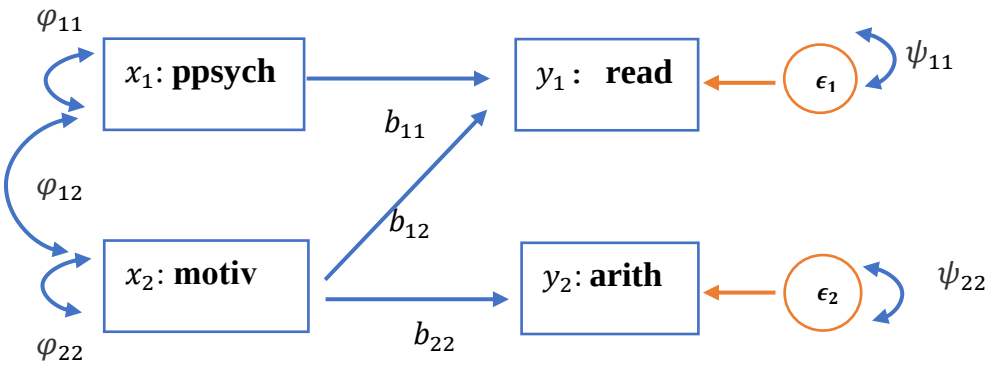
Here x_1 **ppsych** and x_2**motiv** predict y_1**read** and only x_2**motiv** predicts y_1**arith**. The parameters φ_{11}, φ_{22} represent the variance of the two exogenous variables respectively and $\varphi_{12}$is the covariance. Note that these parameters are modeled implicitly and not depicted in lavaan output. The parameters $\in_1$, $\in_2$ refer to the residuals of **read** and **arith**. Finally, ψ_{11}, ψ_{22} represent the residual *variances* of **read** and **arith** and ψ_{12} is its covariance.

You can identify residual terms lavaan by noting a **.** before the term in the output.

```
# Multivariate Regression

m3a <- ' # regressions
          read ~ 1 + ppsych + motiv
          arith ~ 1 + motiv '

fit3a <- sem(m3a, data=dat)
summary(fit3a)
```

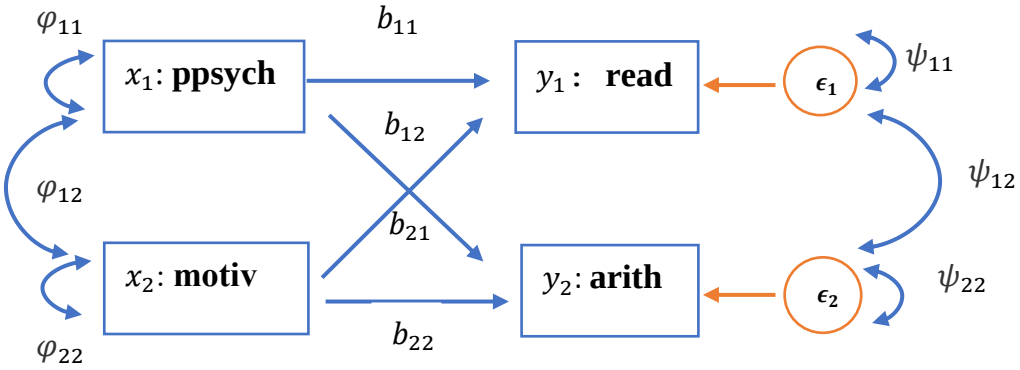
b. *Multivariate regression removing default covariances*

lavaan by default will covary residual variances of endogenous variables **.read ~~ .arith**. Removing the default residual covariances, we see in the path diagram:




```
m3d <- ' # regressions
           read ~ 1 + ppsych + motiv
           arith ~ 1 + motiv
           # covariance
           read ~~ 0*arith '

fit3d <- sem(m3d, data=dat)
summary(fit3d)
```

c. *Fully saturated Multivariate Regression*



In **lavaan**, this is easy as adding the additional path of **arith** on **ppsych** but remembering that **lavaan** also by default models the covariance of ψ_{12} which is the residual covariance between **read** and **arith**.

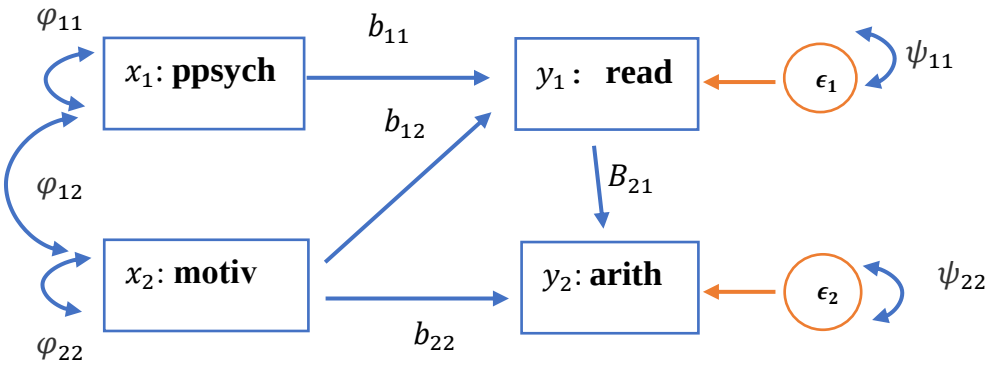
```
m3e <- ' # regressions
           read ~ 1 + ppsych + motiv
           arith ~ 1 + ppsych + motiv '

fit3e <- sem(m3e, data=dat)
summary(fit3e)
```

4. Path Analysis

Multivariate regression is a special case of path analysis where only exogenous variables predict endogenous variables. Path analysis is a more general model where all variables are still manifest but endogenous variables are allowed to explain other endogenous variables. Since **b** specifies relations between an endogenous (y) and exogenous (x) variable, we need to create a new matrix B that specifies the relationship between two endogenous (y) variables.

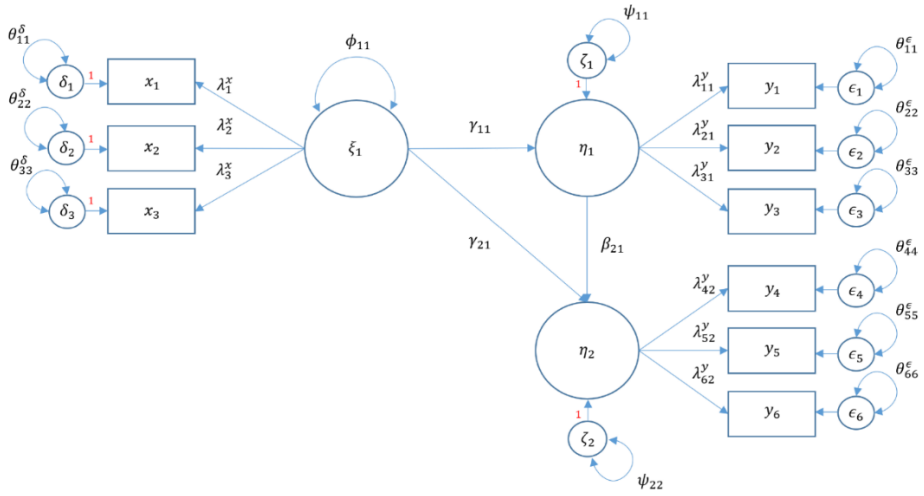
$$\mathbf{y} = \alpha + \mathbf{bx} + \mathbf{By} + \epsilon$$

Let's extend our previous multivariate regression so that we believe **read** which is an endogenous variable also predicts **arith**. Then the path diagram for this model is shown below:




```
# Regressions
m4a <- ' read ~ 1 + ppsych + motiv
          arith ~ 1 + motiv + read '

fit4a <- sem(m4a, data=dat)
summary(fit4a)
```

5. Structural regression with two endogenous variables



```
m6b <- ' # measurement model
          adjust =~ motiv + harm + stabi
          risk =~ verbal + ses + ppsych
          achieve =~ read + arith + spell
          # Regressions
          adjust ~ risk
          achieve ~ adjust + risk '

fit6b <- sem(m6b, data=dat)
summary(fit6b, standardized=TRUE, fit.measures=TRUE)
```

Note: You can download all the example codes mentioned above from this *Repository* and run them in RStudio.

Degrees of Freedom

Both simple regression and multiple regression are *saturated* models which means that all parameters are fully estimated and there are no degrees of freedom.

Models that are just-identified or saturated have degrees of freedom = 0. An under-identified model means that the number known values is less than the number of free parameters and an over-identified model means that the number of known values is greater than the number of free parameters. To summarize

- known < free ➔ degrees of freedom < 0 (under-identified, bad)
- known = free ➔ degrees of freedom = 0 (just identified, neither bad nor good)
- known > free ➔ degrees of freedom > 0 (over-identified, good)

Modification Index

Over-identified models allow flexibility in modeling the remaining degrees of freedom. For example in the path analysis model mentioned above, we can add an additional path between **ppsych** and **read** but we can also add a covariance between **.read** and **.arith**. Adding either of these parameters results in a fully saturated model. **Without a strong a priori hypothesis, it may be difficult ascertain the best parameter to estimate.** One solution is to use the **modification index,** which is a one degree of freedom chi-square test that assesses how the model chi-square will change as a result of including the parameter in the model. The higher the chi-square change, the bigger the impact of adding the additional parameter. To implement the modification index in lavaan, we must input into the **modindices** function a previously estimated lavaan model, which in this case is fit4a. The option sort=TRUE requests the most impactful parameters be placed first based on the change in chi-square.

```
modindices(fit4a, sort = T)
```

Just because modification indexes present us with suggestions for improving our model fit does not mean as a researcher, we can freely alter our model.

Model Fit Statistics for CFA & SEM

Modification indexes gives suggestions about ways to improve model fit, but it is helpful to assess the model fit of your current model to see if improvements are necessary. As we have seen, multivariate regression and path analysis models are not always saturated, meaning the degree of freedom is not zero. This allows us to look at what are called Model Fit Statistics, which measure how closely the (population) model-implied covariance matrix $\Sigma(\theta)$ matches the (population) observed covariance matrix Σ. SEM is also known as covariance structure analysis, which means the hypothesis of interest is regarding the covariance matrix.

By default, **lavaan** outputs the model chi-square a.k.a **Model Test User Model**. To request additional fit statistics you add the **fit.measures = T** option to summary.

```
# Fit statistics
summary(fit4a, fit.measures = T)
```

When fit measures are requested, lavaan outputs a plethora of statistics, but we will focus on the four commonly used ones:

- **Model chi-square** is the chi-square statistic we obtain from the maximum likelihood statistic (in lavaan, this is known as the Test Statistic for the Model Test User Model)
- **CFI** is the Comparative Fit Index – values can range between 0 and 1 (values greater than 0.90, conservatively 0.95 indicate good fit)
- **TLI** Tucker Lewis Index which also ranges between 0 and 1 (if it's greater than 1 it should be rounded to 1) with values greater than 0.90 indicating good fit. If the CFI and TLI are less than one, the CFI is always greater than the TLI.
- **RMSEA** is the root mean square error of approximation.
  - ≤ 0.05 (*close-fit*)
  - between $.05$ and $.08$ (*reasonable approximate fit*, fails close-fit but also fails poor-fit)
  - $>= 0.10$ (*poor-fit*)

Some of the most important functions in the lavaan package, which is commonly used for structural equation modeling (SEM) in R:

1. cfa(): This function is used to specify and estimate confirmatory factor analysis (CFA) models. It allows you to define the measurement model and estimate factor loadings, item intercepts, and factor variances.
2. sem(): The sem() function is used to specify and estimate structural equation models. It allows you to define the structural relationships between latent variables, as well as the measurement model for observed variables.
3. lavaan() or lavaanList(): These functions are used to fit multiple models simultaneously. lavaan() fits a single model, while lavaanList() can fit multiple models and compare them using model fit indices.
4. fitMeasures(): This function calculates various fit indices to assess the goodness-of-fit of a model. It provides statistics such as chi-square, RMSEA, CFI, TLI, and others.
5. summary(): The summary() function provides a summary of the fitted model, including parameter estimates, standard errors, t-values, and p-values.
6. modificationIndices(): This function calculates the modification indices for a fitted model. Modification indices suggest potential improvements to the model by indicating which parameters might be worth adding or removing.
7. parameterEstimates(): This function extracts the parameter estimates from a fitted model. It provides information about the estimated values, standard errors, t-values, and p-values for each parameter.
8. lavInspect(): This function allows you to inspect various aspects of the fitted model, such as residual covariances, standardized estimates, modification indices, and standardized residuals.
9. lavPredict(): The lavPredict() function is used to generate predicted values based on a fitted model. It can be used to estimate latent variable scores or predict outcomes based on the model.
10. lavTestScore(): This function performs a Wald test of measurement invariance across groups in a multiple-group analysis. It tests whether the measurement model is invariant across different groups (e.g., gender, age).

These are just a selection of important functions in the lavaan package. There are many more functions available that provide additional capabilities for SEM analysis, model modification, model comparison, and more. The package documentation and resources provide more detailed information on using these functions and performing structural equation modeling with lavaan.