Structural Equation Modeling

Lab session: Introduction to `lavaan`

Names used

data #The dataset used

`model` #The model specification to be estimated

`fit` #An object of class `lavaan` typically returned from functions `cfa`, `sem`, and `lavaan`

Main functions

`cfa` #Estimate confirmatory factor analysis

`sem` #Estimate structural equation model

`lavaan` # Estimate latent variables model – the other two are wrappers for this function (set some parameters to different default values)

Calling the functions (`cfa`, `sem` or `lavaan`):

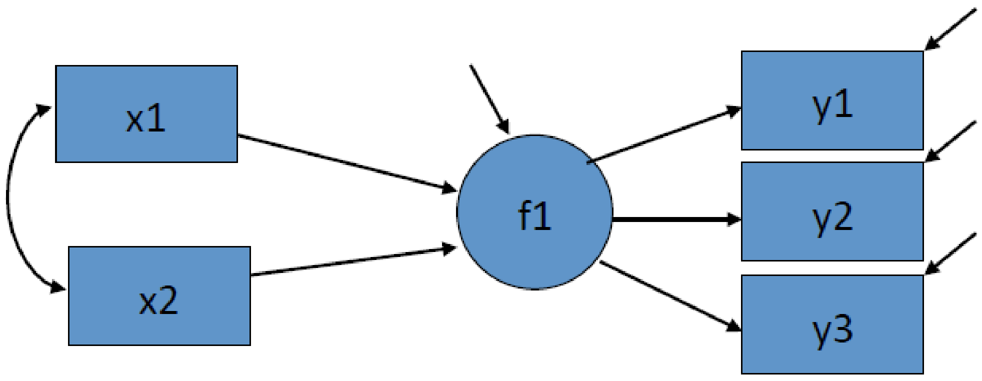
`lavaan(model, data = data, ...) #...` stands for additional parameters

Model specification

Model is specified using a "lavaan model syntax". See ?`model.`syntax for more details. The model is specified with a string object, one specification by line. The usual R modle syntax is used when applicable. The following operators are used:

Formula type	Operator	Mnemonic
Latent variable	=~	is manifested by (or is measured by)
Regression	~	is regressed on
(Residual) (co)variance	~~	is correlated with

Model 1



Lavaan syntax:

```
f1 =~ y1 + y2 + y3
f1 ~ x1 + x2
x1 ~~ x2
```

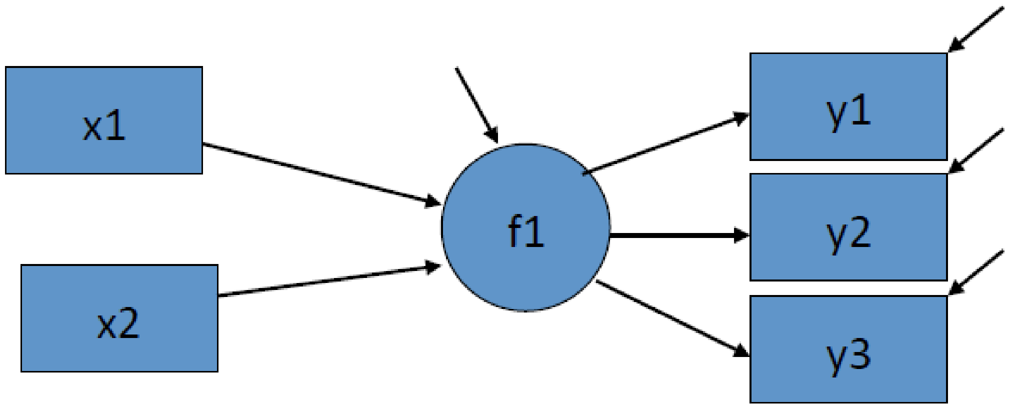
Defaults for functions `cfa` and `sem` (function `lavaan` has different defaults):

- Variances of latent variables and endogenous variables are estimated freely
- The first factor loading of a concept is fixed to 1
- Covariances between exogenous variables are estmated freely

This defaults can be changed.

- To fix a parameter, pre-multiply the corresponding variable by the selected value. E.
- To set a parameter free, pre-multiply the corresponding variable by the `NA`.

Model 2: Alternative parameterization: the first factor loading is set free; the variance of F1 is fixed at 1; no correlation between X1 and X2



Lavaan syntax:

```
f1 ~= NA*y1 + y2 + y3
f1 ~ x1 + x2
f1 ~~ 1*f1
x1 ~~ 0*x2
```

If we want to set variances of all latent variables to 1, one can use the argument `std.lv = TRUE`.

Restrictions and calculations of indirect effects

Both can be done using "labels". Labels are given to parameters by pre-multiplying the selected variable with a label. If two parameters are given equal labels, they are restricted to be the same.

Examples:

Loadings of variables y2 and y3 set to equal:

```
f1 ~= y1 + a*y2 + a*y3
```

Variances of x1 and x2 set to equal:

```
x1 ~~ b*x1
x2 ~~ b*x2
```

Covariances cov(x1, x2) and cov(x1, x3) set to equal:

```
x1 ~~ c*x2
x1 ~~ c*x3
```

Calculation of total, indirect and direct effects of x1 on y (via x2)

```
x2 ~ a*x1
y~ b*x1+ c*x2
indir := a*c
total := b + a*c
```

Important parameters for “fitting” functions (cfa, sem, lavaan)

`mimic=”Mplus”` #Which program should the program “mimic” in terms of default estimation techniques and output. “Mplus” means of courseto mimic Mplus (other options, e.g. “EQS” and “lavaan” (default), are also available)

`missing = ”fiml”` #What to do in case of missing data. “fiml” or “ml” or “direct” means to use Full information maximum likelihood, while “listwise” will use listwise deletion, if ML is used for estimation.

`estimator = “ml”` #Can be one of the following: "ML" for maximum likelihood, "GLS" for generalized least squares, "WLS" for weighted least squares (sometimes called ADF estimation), "ULS" for unweighted least squares and "DWLS" for diagonally weighted least squares. These are the main options that affect the estimation. For convenience, the "ML" option can be extended as "MLM", "MLMV","MLMVS", "MLF", and "MLR". The estimation will still be plain "ML", but now with robust standard errors and a robust (scaled) test statistic.

`group="region"` #The name of the variable defining groups for multigroup analysis.

`ordered = list` #The list of variables (endogeneous) that should be treated as ordinal variables. If the variables are in a form of `ordred` factor this is automatically detected by `lavaan` (this argument is not necessary)

Extracting results/output from results of “fitting” functions (cfa, sem, lavaan)

`summary(fit, fit.measures=TRUE, standardized=TRUE, rsquare=TRUE)` #Main output, with options for returning also fit measures, standardized results and R^2.

`modInd<-modificationIndices(fit)` #Computing modification indices.

`subset(modInd,mi>10)` #Selecting only those with value of the modification index over 10.

`semPaths(fit, what='path', whatLabels= 'std', intercepts = FALSE)` #Plotting sem diagram with some most common options – package `semPlot`

`inspect(fit, what = "coef")` #Extracting parts of the computed objects

`measurementInvariance(..., std.lv = FALSE, strict = FALSE, quiet = FALSE)` #Testing measurement invariance across groups using a typical sequence of model comparison tests. ... stands for paramters for any lavaan model. – Package `semTools`.

`anova(fit1, fit2)` #Comparing two nested models