Structural Equation Modeling with lavaan

Yves Rosseel
Department of Data Analysis
Ghent University

Maastricht, 8 March 2024

Before we start . . .

- slides (and all other files) are available here:

  https://users.ugent.be/~yrosseel/lavaan/maastricht2024

- in the interest of time, we will only spend a short time for the 'practical session'
- but after this course, you can use the `lavaan_labs.pdf` document to practice what you have learned
- a starting script (`lavaan_labs_start.R`) and a feedback script (`lavaan_labs_feedback.R`) are also available

Contents

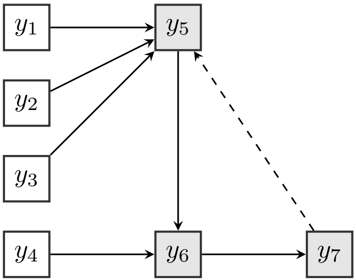
1 Introduction to SEM

1.1 What is SEM?

- SEM is a multivariate statistical modeling technique
- SEM allows us to test a hypothesis/model about the data
  - we postulate a data-generating model
  - this model may or may not fit the data
- what is so special about SEM?
  1. the model may contain latent variables
     - latent variables can be hypothetical 'constructs' (eg., depression) measured by a set of indicators
     - latent variables can be random effects (eg., random intercepts)
     - error terms, missing data, ...
  2. SEM allows for indirect effects (mediation), reciprocal effects, ...
  3. the model is depicted as a diagram

path analysis

- all variables are observed (manifest)
- we allow for indirect effects (eg., of y_5, via y_6 on y_7)
- we allow for cycles (eg. y_7 could influence y_5)



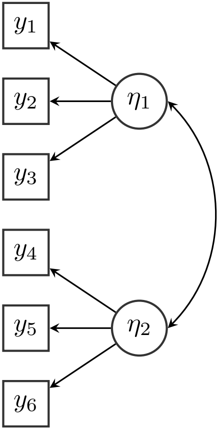
y_5 = reading motivation

y_6 = reading frequency

y_7 = reading ability

confirmatory factor analysis (CFA)

- measurement model: representing the relationship between one or more latent variables and their (observed) indicators

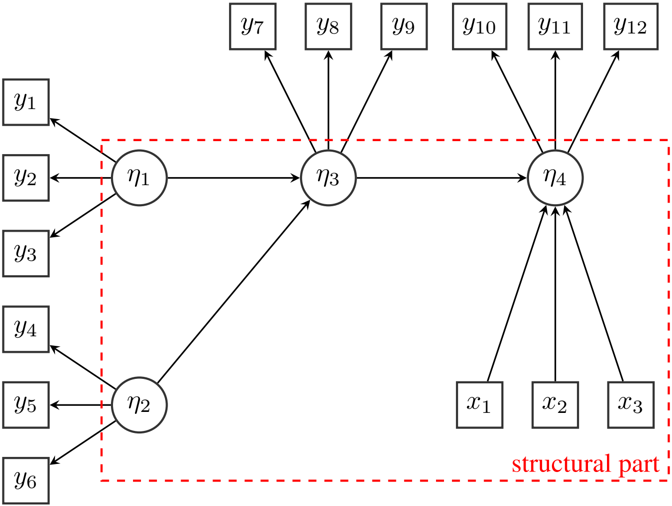


η_1 = depression

η_2 = neuroticism

structural equation modeling (SEM)

- path analysis with latent variables



1.2 How does SEM work?

a dataset: the Holzinger & Swineford dataset

```
> library(lavaan)
> var.names <- c("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9")
> summary(HolzingerSwineford1939[, var.names])
```

```
       x1               x2               x3               x4
 Min.   :0.6667   Min.   :2.250   Min.   :0.250   Min.   :0.000
 1st Qu.:4.1667   1st Qu.:5.250   1st Qu.:1.375   1st Qu.:2.333
 Median :5.0000   Median :6.000   Median :2.125   Median :3.000
 Mean   :4.9358   Mean   :6.088   Mean   :2.250   Mean   :3.061
 3rd Qu.:5.6667   3rd Qu.:6.750   3rd Qu.:3.125   3rd Qu.:3.667
 Max.   :8.5000   Max.   :9.250   Max.   :4.500   Max.   :6.333
       x5              x6               x7               x8
 Min.   :1.000   Min.   :0.1429   Min.   :1.304   Min.   : 3.050
 1st Qu.:3.500   1st Qu.:1.4286   1st Qu.:3.478   1st Qu.: 4.850
 Median :4.500   Median :2.0000   Median :4.087   Median : 5.500
 Mean   :4.341   Mean   :2.1856   Mean   :4.186   Mean   : 5.527
 3rd Qu.:5.250   3rd Qu.:2.7143   3rd Qu.:4.913   3rd Qu.: 6.100
 Max.   :7.000   Max.   :6.1429   Max.   :7.435   Max.   :10.000
       x9
 Min.   :2.778
 1st Qu.:4.750
 Median :5.417
 Mean   :5.374
```

```
 3rd Qu.:6.083
 Max.   :9.250
```

computing the variance-covariance matrix for $P = 9$ variables

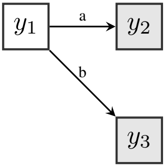
```
> N <- nrow(HolzingerSwineford1939)
> S <- cov( HolzingerSwineford1939[, var.names] )
> S <- S * (N-1)/N # ML version
> round(S, 3)

      x1     x2    x3    x4    x5    x6     x7    x8    x9
x1 1.358  0.407 0.580 0.505 0.441 0.455  0.085 0.264 0.458
x2 0.407  1.382 0.451 0.209 0.211 0.248 -0.097 0.110 0.244
x3 0.580  0.451 1.275 0.208 0.112 0.244  0.088 0.212 0.374
x4 0.505  0.209 0.208 1.351 1.098 0.896  0.220 0.126 0.243
x5 0.441  0.211 0.112 1.098 1.660 1.015  0.143 0.181 0.295
x6 0.455  0.248 0.244 0.896 1.015 1.196  0.144 0.165 0.236
x7 0.085 -0.097 0.088 0.220 0.143 0.144  1.183 0.535 0.373
x8 0.264  0.110 0.212 0.126 0.181 0.165  0.535 1.022 0.457
x9 0.458  0.244 0.374 0.243 0.295 0.236  0.373 0.457 1.015
```

the model-implied variance-covariance matrix

- the goal of SEM is to test an a priori specified theory/model, based on empirical data; we would like to know if our model 'fits' the data (or not)
- each model can be depicted by a path diagram (we may have several alternative models, each one with its own path diagram)
- each path diagram can be converted to a SEM
- SEM will tell us what the implications are for the data if (assumption!) our model is correct: how 'should' the data look like, which patterns should we observe?
- in practice, SEM will tell us how the variance-covariance matrix of the data should look like; we call this the 'model-implied' variance-covariance matrix ($\hat{\mathbf{\Sigma}}$)
- different models $\rightarrow$ different path diagrams $\rightarrow$ different $\hat{\mathbf{\Sigma}}$ matrices
- if $\hat{\mathbf{\Sigma}}$ is close to $\mathbf{S}$, the model fits well

example model-implied covariance matrix (1)

- suppose we have three observed (random) variables, y_1, y_2 and y_3; to explain why they are correlated, we may postulate the following model:



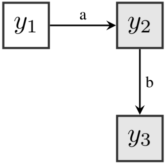
$$y_2 = a\, y_1 + \epsilon_2$$
$$y_3 = b\, y_1 + \epsilon_3$$

- suppose, we set $a = 3$ en $b = 5$, $\text{Var}(y_1) = 10$, $\text{Var}(\epsilon_2) = 20$, $\text{Var}(\epsilon_3) = 30$; then, it can be shown that the model-implied variance-covariance matrix equals

$$\hat{\boldsymbol{\Sigma}} = \begin{bmatrix} 10 & & \\ 30 & 110 & \\ 50 & 150 & 280 \end{bmatrix}$$

example model-implied covariance matrix (2)

- but if we change the path diagram (and keep the parameter values fixed), the model-implied covariance matrix will also change:



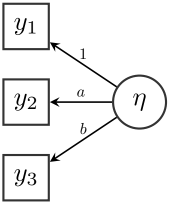
we find

$$\boldsymbol{\Sigma} = \begin{bmatrix} 10 & & \\ 30 & 110 & \\ 150 & 550 & 2780 \end{bmatrix}$$

- two models are said to be *equivalent*, if they imply the same covariance matrix (but note that we did not estimate the parameters here)

example model-implied covariance matrix (3)

- we can also postulate that the correlations among the three observed variables are explained by a common 'factor':

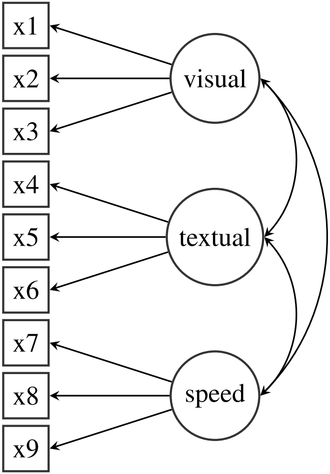


- we find using $\sigma^2(\epsilon_1) = 10$, $\sigma^2(\epsilon_2) = 20$, $\sigma^2(\epsilon_3) = 30$, $\sigma^2(\eta) = 1$:

$$\mathbf{\Sigma} = \begin{bmatrix} 11 & & \\ 4 & 36 & \\ 5 & 20 & 55 \end{bmatrix}$$

- we can compare all three $\hat{\mathbf{\Sigma}}$ matrices to $\mathbf{S}$ to find out which model fits best

1.3 A first example: a CFA with three factors



- ‘free’ parameters: factor loadings, variances for the factors, covariances between the factors, and residual variances for the indicators

the matrix representation of a CFA model

- the classic LISREL representation uses three model matrices for a CFA
- the LAMBDA matrix contains the 'factor structure':

$$\mathbf{\Lambda} = \begin{bmatrix} x & 0 & 0 \\ x & 0 & 0 \\ x & 0 & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix}$$

- the variances/covariances of the latent variables are summarized in the PSI matrix:

$$\boldsymbol{\Psi} = \begin{bmatrix} x & & \\ x & x & \\ x & x & x \end{bmatrix}$$

- what we can *not* explain by the set of common factors (the 'residual part' of the model) is written in the (typically diagonal) matrix THETA:

$$\boldsymbol{\Theta} = \begin{bmatrix} x & & & & & & & & \\ & x & & & & & & & \\ & & x & & & & & & \\ & & & x & & & & & \\ & & & & x & & & & \\ & & & & & x & & & \\ & & & & & & x & & \\ & & & & & & & x & \\ & & & & & & & & x \end{bmatrix}$$

- note that we have only 24 parameters (of which 21 are estimable)

the standard CFA model: the model implied covariance matrix

- in the standard CFA model, the 'implied' covariance matrix is:

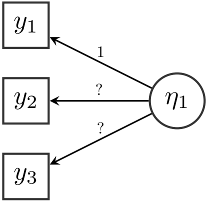
$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Psi}\mathbf{\Lambda}' + \mathbf{\Theta}$$

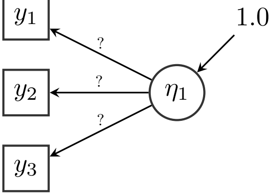
- all parameters are included in three model matrices
- simple matrix multiplication (and addition) gives us the model implied covariance matrix
- for identification purposes, some parameters need to be fixed to a constant (see next slide)
- estimation problem: choose the 'free' parameters, so that the estimated implied covariance matrix ($\hat{\mathbf{\Sigma}}$) is 'as close as possible' to the observed covariance matrix $\mathbf{S}$
  - generalized (weighted) least-squares estimation (GLS, WLS)
  - maximum likelihood estimation (ML)
  - Bayesian approaches

setting the metric of the latent variables: UVI of ULI

1. *Unit Loading Identification* (ULI):
   the factor loading of one (often the first) of the indicators is fixed to 1.0; this indicator is called the *reference* indicator

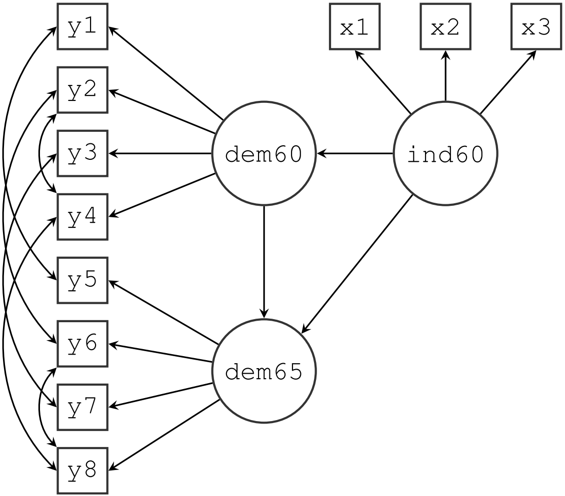
2. *Unit Variance Identification* (UVI):
   the variance of the factor is fixed to 1.0





- in many models, it does not matter
- in multigroup SEM analysis: we usually use ULI

1.4 A second example: the Political Democracy dataset



model matrices

- this is an example of a 'full SEM': the model contains both a measurement part, and a structural part
- we now need 4 model matrices:
  - LAMBDA: the factor loadings
  - THETA: the residual variances (and covariances) of the observed indicators
  - PSI: the (residual) variances and covariances of the latent variables
  - BETA: the regression coefficients of the structural part
- the formula to obtain the model-implied variance-covariance matrix is now slightly more complex:

$$\mathbf{\Sigma} = \mathbf{\Lambda}(\mathbf{I} - \mathbf{B})^{-1}\mathbf{\Psi}(\mathbf{I} - \mathbf{B})^{-1\prime}\mathbf{\Lambda}' + \mathbf{\Theta}$$

  where $\mathbf{I}$ is the identity matrix

1.5 Model estimation

- we seek those values for $\boldsymbol{\theta}$ that minimize the difference between what we observe in the data, $\mathbf{S}$, and what the model implies, $\boldsymbol{\Sigma}(\boldsymbol{\theta})$
- the final estimated values are denoted by $\hat{\boldsymbol{\theta}}$, and the estimated model-implied covariance matrix can be written as $\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$
- there are many ways to quantify this 'difference', leading to different discrepancy measures
- the most used discrepancy measure is based on maximum likelihood:

$$F_{ML}(\boldsymbol{\theta}) = \log|\boldsymbol{\Sigma}| + \mathsf{tr}(\mathbf{S}\boldsymbol{\Sigma}^{-1}) - \log|\mathbf{S}| - P$$

- in practice, we replace $\boldsymbol{\Sigma}$ by $\hat{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}(\hat{\boldsymbol{\theta}})$
- an alternative is (weighted) least squares, for some weight matrix $\mathbf{W}$:

$$F_{WLS}(\boldsymbol{\theta}) = (\mathbf{s} - \boldsymbol{\sigma})'\mathbf{W}^{-1}(\mathbf{s} - \boldsymbol{\sigma})$$

where $\mathbf{s}$ and $\boldsymbol{\sigma}$ are the unique elements of $\mathbf{S}$ and $\boldsymbol{\Sigma}$ respectively

1.6 Model evaluation

evaluation of global fit – chi-square test statistic

- the chi-square test statistic is the primary test of our model
- if the chi-square test statistic is NOT significant, we have a good fit of the model
- this becomes increasingly difficult if the sample size grows

evaluation of global fit – fit indices

- (some) rules of thumb: CFI/TLI > 0.95, RMSEA < 0.05, SRMR < 0.06
- there is a lot of controversy about the use (and misuse) of these fit indices
- a good reference is still Hu & Bentler (1999)
- current practice is to report: chi-square value + df + pvalue, RMSEA, CFI and SRMR (do not cherry pick your fit indices)

1.7 Model respecification

- if the fit of a model is not good, we can adapt (respecify) the model
  - change the number of factors
  - allow for indicators to be related to more than one factor (cross-loadings)
  - allow for correlated residual errors among the observed indicators
  - allow for correlated disturbances among the endogenous latent variables
  - remove problematic indicators . . .
- ideally, all changes should have a sound theoretical justification
- of course, we may let the data speak for itself, and have a look at the modification indices (a more explorative approach)

1.8 Further reading

Kline, R. B. (2015). Principles and practice of structural equation modeling (Fourth Edition). New York: Guilford Press.

Bollen, K.A. (1989). Structural equations with latent variables. New York: Wiley.

Loehlin, J. C., & Beaujean, A. A. (2016). Latent variable models: An introduction to factor, path, and structural equation analysis. Taylor & Francis.

Beaujean, A. A. (2014). Latent variable modeling using R: A step-by-step guide. New York: Routledge.

Finch, W.H., and French, B.F. (2015). Latent Variable Modeling with R. Routledge.

2 Introduction to lavaan

2.1 Software for SEM

software for SEM: commercial – closed-source

- LISREL, EQS, AMOS, MPLUS
- SAS/Stat: proc (T)CALIS, SEPATH (Statistica), RAMONA (Systat), Stata (12 or higher)
- Mx (free, closed-source)

software for SEM: non-commercial – open-source

- outside the R ecosystem: gllamm (Stata), Onyx, semopy, …
- R packages: sem, OpenMx, lavaan, lava, psychonetrics

2.2 The R package 'lavaan'

what is lavaan?

- **lavaan** is an R package for latent variable analysis:
  - user-friendly syntax, user-friendly interface (the `sem()` function)
  - support for continuous, binary and ordinal data
  - support for missing data, nonnormal data, clustered data, ...
  - available technology: CFA, SEM, path-analysis, growth curve modeling, exploratory factor analysis (EFA), ESEM, multilevel SEM, ...
  - unique for lavaan: bounded estimation, pairwise likelihood (PL) estimation, distributionally weighted least squares (DLS), structural-after-measurement (SAM) approach to SEM, ...
- under development, future plans:
  - technical interface, more algorithms, simulation framework, (multi-group) mixture EFA/CFA/SEM, IRT, new engine, ...

installing lavaan, finding documentation

- **lavaan** depends on the R project for statistical computing:

  https://www.r-project.org

- to install **lavaan**, simply start up an R session and type:

```
> install.packages("lavaan")
```

- more information about **lavaan**:

  http://lavaan.org

- the lavaan paper:

  Rosseel (2012). lavaan: an R package for structural equation modeling. *Journal of Statistical Software*, 48(2), 1–36.

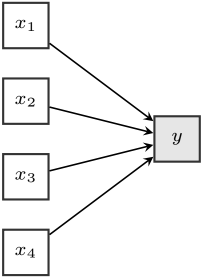
- **lavaan** discussion group (mailing list)

  https://groups.google.com/d/forum/lavaan

2.3 The lavaan model syntax

using standard R – a simple regression

- using the `lm` function in R:




```
# read in your data
myData <- read.csv("c:/temp/myData.csv")

# fit model using lm
fit <- lm(formula = y ~ x1 + x2 + x3 + x4,
          data    = myData)

# show results
summary(fit)
```

- the standard linear model:

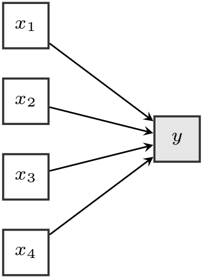
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$$

lm() output artificial data (N=100)

```
> summary(fit)

Call:
lm(formula = y ~ x1 + x2 + x3 + x4, data = myData)

Residuals:
     Min       1Q   Median       3Q      Max
-102.372  -29.458   -3.658   27.275  148.404

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  97.7210     4.7200  20.704   <2e-16 ***
x1            5.7733     0.5238  11.022   <2e-16 ***
x2           -1.3214     0.4917  -2.688   0.0085 **
x3            1.1350     0.4575   2.481   0.0149 *
x4            0.2707     0.4779   0.566   0.5724
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 46.74 on 95 degrees of freedom
Multiple R-squared:  0.5911,        Adjusted R-squared:  0.5738
F-statistic: 34.33 on 4 and 95 DF,  p-value: < 2.2e-16
```

the lavaan model syntax – a simple regression

- using lavaan's `sem` function:




```
library(lavaan)
myData <- read.csv("c:/temp/myData.csv")

myModel <- ' y ~ x1 + x2 + x3 + x4 '

# fit model
fit <- sem(model = myModel,
           data  = myData)

# show results
summary(fit, nd = 4, header = FALSE)
```

- to 'see' the intercept, use either

```
fit <- sem(model = myModel, data = myData, meanstructure = TRUE)
```

or include it explicitly in the syntax:

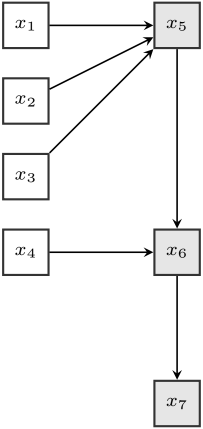
```
myModel <- ' y ~ 1 + x1 + x2 + x3 + x4 '
```

(partial) lavaan output

```
Parameter Estimates:

  Standard errors                                   Standard
  Information                                       Expected
  Information saturated (h1) model                Structured

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  y ~
    x1               5.7733   0.5105  11.3087   0.0000
    x2              -1.3214   0.4792  -2.7574   0.0058
    x3               1.1350   0.4459   2.5451   0.0109
    x4               0.2707   0.4658   0.5812   0.5611

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y             2075.0999 293.4634   7.0711   0.0000
```

the lavaan model syntax – path analysis

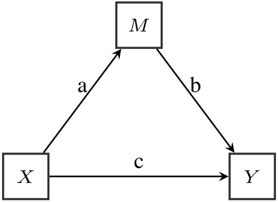
- for each dependent variable, we write a separate regression equation:




```
myModel <- ' x5 ~ x1 + x2 + x3
             x6 ~ x4 + x5
             x7 ~ x6           '
```

the lavaan model syntax – mediation analysis

- a mediation analysis is simple
- we can use labels to refer to specific parameters (here regression coefficients)
- standard errors are based on the bootstrap

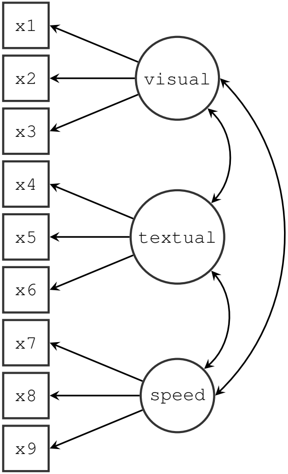



```
myModel <- '
             Y ~ b*M + c*X
             M ~ a*X

             indirect := a*b
             total    := c + (a*b)
           '

fit <- sem(model = myModel,
           data  = myData,
           se    = "bootstrap")

summary(fit)
```

partial output

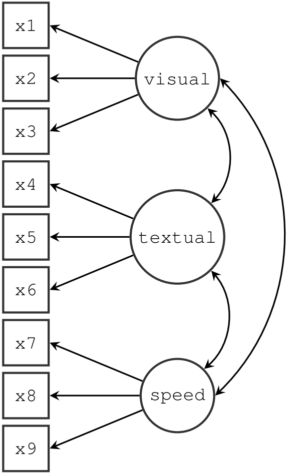
```
Parameter estimates:

  Information                                    Observed
  Standard Errors                               Bootstrap
  Number of requested bootstrap draws               1000
  Number of successful bootstrap draws              1000

Regressions:
                   Estimate  Std.err  z-value  P(>|z|)
  Y ~
    M         (b)     0.597    0.098    6.068    0.000
    X         (c)     2.594    1.210    2.145    0.032
  M ~
    X         (a)     2.739    0.999    2.741    0.006

Variances:
                   Estimate  Std.err  z-value  P(>|z|)
   .Y               108.700   17.747    6.125    0.000
   .M               105.408   16.556    6.367    0.000

Defined parameters:
                   Estimate  Std.err  z-value  P(>|z|)
    indirect          1.636    0.645    2.535    0.011
    total             4.230    1.383    3.059    0.002
```

the lavaan model syntax – using cfa() or sem()




```
HS.model <- ' visual   =~ x1 + x2 + x3
              textual  =~ x4 + x5 + x6
              speed    =~ x7 + x8 + x9
            '

fit <- cfa(model = HS.model,
           data  = HolzingerSwineford1939)

summary(fit, fit.measures = TRUE,
             standardized = TRUE)
```

the lavaan model syntax – using lavaan()




```
HS.model <- '
  # latent variables
    visual  =~ 1*x1 + x2 + x3
    textual =~ 1*x4 + x5 + x6
    speed   =~ 1*x7 + x8 + x9

  # factor (co)variances
    visual  ~~ visual; visual  ~~ textual
    visual  ~~ speed;  textual ~~ textual
    textual ~~ speed;  speed   ~~ speed

  # residual variances
    x1 ~~ x1; x2 ~~ x2; x3 ~~ x3
    x4 ~~ x4; x5 ~~ x5; x6 ~~ x6
    x7 ~~ x7; x8 ~~ x8; x9 ~~ x9
'

fit <- lavaan(model = HS.model,
              data  = HolzingerSwineford1939)

summary(fit, fit.measures = TRUE,
             standardized = TRUE)
```

full output

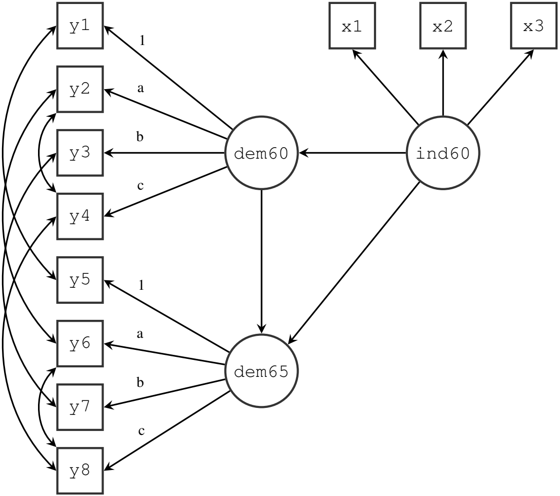
```
lavaan 0.6-12 ended normally after 35 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        21

  Number of observations                           301

Model Test User Model:

  Test statistic                                85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Model Test Baseline Model:

  Test statistic                               918.852
  Degrees of freedom                                36
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.931
  Tucker-Lewis Index (TLI)                       0.896

Loglikelihood and Information Criteria:
```

```
  Loglikelihood user model (H0)              -3737.745
  Loglikelihood unrestricted model (H1)      -3695.092

  Akaike (AIC)                                7517.490
  Bayesian (BIC)                              7595.339
  Sample-size adjusted Bayesian (BIC)         7528.739

Root Mean Square Error of Approximation:

  RMSEA                                          0.092
  90 Percent confidence interval - lower         0.071
  90 Percent confidence interval - upper         0.114
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.065

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
  visual =~
```

```
    x1              1.000                               0.900    0.772
    x2              0.554    0.100    5.554    0.000    0.498    0.424
    x3              0.729    0.109    6.685    0.000    0.656    0.581
  textual =~
    x4              1.000                               0.990    0.852
    x5              1.113    0.065   17.014    0.000    1.102    0.855
    x6              0.926    0.055   16.703    0.000    0.917    0.838
  speed =~
    x7              1.000                               0.619    0.570
    x8              1.180    0.165    7.152    0.000    0.731    0.723
    x9              1.082    0.151    7.155    0.000    0.670    0.665

Covariances:
                 Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    textual         0.408    0.074    5.552    0.000    0.459    0.459
    speed           0.262    0.056    4.660    0.000    0.471    0.471
  textual ~~
    speed           0.173    0.049    3.518    0.000    0.283    0.283

Variances:
                 Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1              0.549    0.114    4.833    0.000    0.549    0.404
   .x2              1.134    0.102   11.146    0.000    1.134    0.821
   .x3              0.844    0.091    9.317    0.000    0.844    0.662
   .x4              0.371    0.048    7.779    0.000    0.371    0.275
   .x5              0.446    0.058    7.642    0.000    0.446    0.269
   .x6              0.356    0.043    8.277    0.000    0.356    0.298
```

```
   .x7              0.799    0.081    9.823    0.000    0.799    0.676
   .x8              0.488    0.074    6.573    0.000    0.488    0.477
   .x9              0.566    0.071    8.003    0.000    0.566    0.558
    visual          0.809    0.145    5.564    0.000    1.000    1.000
    textual         0.979    0.112    8.737    0.000    1.000    1.000
    speed           0.384    0.086    4.451    0.000    1.000    1.000
```

the lavaan model syntax – equality constraints



fitting the model with lavaan

```
# 1. specifying the model
model <- '
  # latent variable definitions
    ind60 =~ x1 + x2 + x3
    dem60 =~ y1 + a*y2 + b*y3 + c*y4
    dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
    dem60 ~ ind60
    dem65 ~ ind60 + dem60

  # residual covariances
    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
'

# 2. fitting the model using the sem() function
fit <- sem(model, data = PoliticalDemocracy)

# 3. display the results
summary(fit, standardized = TRUE)
```

output

```
lavaan 0.6-12 ended normally after 66 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        31
  Number of equality constraints                     3

  Number of observations                            75

Model Test User Model:

  Test statistic                                40.179
  Degrees of freedom                                38
  P-value (Chi-square)                           0.374

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
  ind60 =~
    x1                1.000                               0.670    0.920
    x2                2.180    0.138   15.751    0.000    1.460    0.973
```

```
    x3                1.818    0.152   11.971    0.000    1.218    0.872
  dem60 =~
    y1                1.000                               2.201    0.850
    y2         (a)    1.191    0.139    8.551    0.000    2.621    0.690
    y3         (b)    1.175    0.120    9.755    0.000    2.586    0.758
    y4         (c)    1.251    0.117   10.712    0.000    2.754    0.838
  dem65 =~
    y5                1.000                               2.154    0.817
    y6         (a)    1.191    0.139    8.551    0.000    2.565    0.755
    y7         (b)    1.175    0.120    9.755    0.000    2.530    0.802
    y8         (c)    1.251    0.117   10.712    0.000    2.694    0.829

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  dem60 ~
    ind60             1.471    0.392    3.750    0.000    0.448    0.448
  dem65 ~
    ind60             0.600    0.226    2.661    0.008    0.187    0.187
    dem60             0.865    0.075   11.554    0.000    0.884    0.884

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
 .y1 ~~
   .y5                0.583    0.356    1.637    0.102    0.583    0.281
 .y2 ~~
   .y4                1.440    0.689    2.092    0.036    1.440    0.291
   .y6                2.183    0.737    2.960    0.003    2.183    0.356
 .y3 ~~
```

```
   .y7              0.712    0.611    1.165    0.244    0.712    0.169
 .y4 ~~
   .y8              0.363    0.444    0.817    0.414    0.363    0.111
 .y6 ~~
   .y8              1.372    0.577    2.378    0.017    1.372    0.338

Variances:
                 Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1              0.081    0.019    4.182    0.000    0.081    0.154
   .x2              0.120    0.070    1.729    0.084    0.120    0.053
   .x3              0.467    0.090    5.177    0.000    0.467    0.239
   .y1              1.855    0.433    4.279    0.000    1.855    0.277
   .y2              7.581    1.366    5.549    0.000    7.581    0.525
   .y3              4.956    0.956    5.182    0.000    4.956    0.426
   .y4              3.225    0.723    4.458    0.000    3.225    0.298
   .y5              2.313    0.479    4.831    0.000    2.313    0.333
   .y6              4.968    0.921    5.393    0.000    4.968    0.430
   .y7              3.560    0.710    5.018    0.000    3.560    0.357
   .y8              3.308    0.704    4.701    0.000    3.308    0.313
    ind60           0.449    0.087    5.175    0.000    1.000    1.000
   .dem60           3.875    0.866    4.477    0.000    0.800    0.800
   .dem65           0.164    0.227    0.725    0.469    0.035    0.035
```

2.4 lavaan: a brief user's guide

example: fitted()

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939)
> fitted(fit)

$cov
   x1    x2    x3    x4    x5    x6    x7    x8    x9
x1 1.358
x2 0.448 1.382
x3 0.590 0.327 1.275
x4 0.408 0.226 0.298 1.351
x5 0.454 0.252 0.331 1.090 1.660
x6 0.378 0.209 0.276 0.907 1.010 1.196
x7 0.262 0.145 0.191 0.173 0.193 0.161 1.183
x8 0.309 0.171 0.226 0.205 0.228 0.190 0.453 1.022
x9 0.284 0.157 0.207 0.188 0.209 0.174 0.415 0.490 1.015
```

example: lavInspect()

```
> lavInspect(fit)

$lambda
   visual textul speed
x1      0      0     0
x2      1      0     0
x3      2      0     0
x4      0      0     0
x5      0      3     0
x6      0      4     0
x7      0      0     0
x8      0      0     5
x9      0      0     6

$theta
   x1 x2 x3 x4 x5 x6 x7 x8 x9
x1  7
x2  0  8
x3  0  0  9
x4  0  0  0 10
x5  0  0  0  0 11
x6  0  0  0  0  0 12
x7  0  0  0  0  0  0 13
x8  0  0  0  0  0  0  0 14
x9  0  0  0  0  0  0  0  0 15
```

```
$psi
        visual textul speed
visual  16
textual 19     17
speed   20     21     18
```

```
> lavInspect(fit, "sampstat")

$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  1.358
x2  0.407  1.382
x3  0.580  0.451  1.275
x4  0.505  0.209  0.208  1.351
x5  0.441  0.211  0.112  1.098  1.660
x6  0.455  0.248  0.244  0.896  1.015  1.196
x7  0.085 -0.097  0.088  0.220  0.143  0.144  1.183
x8  0.264  0.110  0.212  0.126  0.181  0.165  0.535  1.022
x9  0.458  0.244  0.374  0.243  0.295  0.236  0.373  0.457  1.015
```

```
> lavInspect(fit, "cov.lv")

        visual textul speed
visual  0.809
textual 0.408  0.979
speed   0.262  0.173  0.384
```

```
> lavTech(fit, "cov.lv")

[[1]]
          [,1]      [,2]      [,3]
[1,] 0.8093160 0.4082324 0.2622246
[2,] 0.4082324 0.9794914 0.1734947
[3,] 0.2622246 0.1734947 0.3837476
```

```
> lavTech(fit, "cov.lv", add.labels = TRUE, drop.list.single.group = TRUE)

           visual  textual     speed
visual  0.8093160 0.4082324 0.2622246
textual 0.4082324 0.9794914 0.1734947
speed   0.2622246 0.1734947 0.3837476
```

example: fitMeasures()

```
> fitMeasures(fit)

              npar               fmin              chisq                 df 
            21.000              0.142             85.306             24.000 
            pvalue     baseline.chisq        baseline.df    baseline.pvalue 
             0.000            918.852             36.000              0.000 
               cfi                tli               nnfi                rfi 
             0.931              0.896              0.896              0.861 
               nfi               pnfi                ifi                rni 
             0.907              0.605              0.931              0.931 
              logl  unrestricted.logl                aic                bic 
         -3737.745          -3695.092           7517.490           7595.339 
            ntotal               bic2              rmsea     rmsea.ci.lower 
           301.000           7528.739              0.092              0.071 
    rmsea.ci.upper       rmsea.pvalue                rmr         rmr_nomean 
             0.114              0.001              0.082              0.082 
              srmr       srmr_bentler srmr_bentler_nomean               crmr 
             0.065              0.065              0.065              0.073 
       crmr_nomean         srmr_mplus  srmr_mplus_nomean              cn_05 
             0.073              0.065              0.065            129.490 
             cn_01                gfi               agfi               pgfi 
           152.654              0.943              0.894              0.503 
               mfi               ecvi 
             0.903              0.423 
```

example: parameterTable()

```
> parameterTable(fit)[1:21,1:13]

   id     lhs op     rhs user block group free ustart exo label plabel start
1   1  visual =~      x1    1     1     1    0      1   0          .p1. 1.000
2   2  visual =~      x2    1     1     1    1     NA   0          .p2. 0.778
3   3  visual =~      x3    1     1     1    2     NA   0          .p3. 1.107
4   4 textual =~      x4    1     1     1    0      1   0          .p4. 1.000
5   5 textual =~      x5    1     1     1    3     NA   0          .p5. 1.133
6   6 textual =~      x6    1     1     1    4     NA   0          .p6. 0.924
7   7   speed =~      x7    1     1     1    0      1   0          .p7. 1.000
8   8   speed =~      x8    1     1     1    5     NA   0          .p8. 1.225
9   9   speed =~      x9    1     1     1    6     NA   0          .p9. 0.854
10 10      x1 ~~      x1    0     1     1    7     NA   0         .p10. 0.679
11 11      x2 ~~      x2    0     1     1    8     NA   0         .p11. 0.691
12 12      x3 ~~      x3    0     1     1    9     NA   0         .p12. 0.637
13 13      x4 ~~      x4    0     1     1   10     NA   0         .p13. 0.675
14 14      x5 ~~      x5    0     1     1   11     NA   0         .p14. 0.830
15 15      x6 ~~      x6    0     1     1   12     NA   0         .p15. 0.598
16 16      x7 ~~      x7    0     1     1   13     NA   0         .p16. 0.592
17 17      x8 ~~      x8    0     1     1   14     NA   0         .p17. 0.511
18 18      x9 ~~      x9    0     1     1   15     NA   0         .p18. 0.508
19 19  visual ~~  visual    0     1     1   16     NA   0         .p19. 0.050
20 20 textual ~~ textual    0     1     1   17     NA   0         .p20. 0.050
21 21   speed ~~   speed    0     1     1   18     NA   0         .p21. 0.050
```

example: parameterEstimates()

```
> parameterEstimates(fit)[1:21,]

       lhs op     rhs   est    se      z pvalue ci.lower ci.upper
1   visual =~      x1 1.000 0.000     NA     NA    1.000    1.000
2   visual =~      x2 0.554 0.100  5.554      0    0.358    0.749
3   visual =~      x3 0.729 0.109  6.685      0    0.516    0.943
4  textual =~      x4 1.000 0.000     NA     NA    1.000    1.000
5  textual =~      x5 1.113 0.065 17.014      0    0.985    1.241
6  textual =~      x6 0.926 0.055 16.703      0    0.817    1.035
7    speed =~      x7 1.000 0.000     NA     NA    1.000    1.000
8    speed =~      x8 1.180 0.165  7.152      0    0.857    1.503
9    speed =~      x9 1.082 0.151  7.155      0    0.785    1.378
10      x1 ~~      x1 0.549 0.114  4.833      0    0.326    0.772
11      x2 ~~      x2 1.134 0.102 11.146      0    0.934    1.333
12      x3 ~~      x3 0.844 0.091  9.317      0    0.667    1.022
13      x4 ~~      x4 0.371 0.048  7.779      0    0.278    0.465
14      x5 ~~      x5 0.446 0.058  7.642      0    0.332    0.561
15      x6 ~~      x6 0.356 0.043  8.277      0    0.272    0.441
16      x7 ~~      x7 0.799 0.081  9.823      0    0.640    0.959
17      x8 ~~      x8 0.488 0.074  6.573      0    0.342    0.633
18      x9 ~~      x9 0.566 0.071  8.003      0    0.427    0.705
19  visual ~~  visual 0.809 0.145  5.564      0    0.524    1.094
20 textual ~~ textual 0.979 0.112  8.737      0    0.760    1.199
21   speed ~~   speed 0.384 0.086  4.451      0    0.215    0.553
```

example: modindices()

```
> modindices(fit, sort = TRUE, minimum.value = 5)

       lhs op rhs     mi    epc sepc.lv sepc.all sepc.nox
30  visual =~  x9 36.411  0.577   0.519    0.515    0.515
76      x7 ~~  x8 34.145  0.536   0.536    0.859    0.859
28  visual =~  x7 18.631 -0.422  -0.380   -0.349   -0.349
78      x8 ~~  x9 14.946 -0.423  -0.423   -0.805   -0.805
33 textual =~  x3  9.151 -0.272  -0.269   -0.238   -0.238
55      x2 ~~  x7  8.918 -0.183  -0.183   -0.192   -0.192
31 textual =~  x1  8.903  0.350   0.347    0.297    0.297
51      x2 ~~  x3  8.532  0.218   0.218    0.223    0.223
59      x3 ~~  x5  7.858 -0.130  -0.130   -0.212   -0.212
26  visual =~  x5  7.441 -0.210  -0.189   -0.147   -0.147
50      x1 ~~  x9  7.335  0.138   0.138    0.247    0.247
65      x4 ~~  x6  6.220 -0.235  -0.235   -0.646   -0.646
66      x4 ~~  x7  5.920  0.098   0.098    0.180    0.180
48      x1 ~~  x7  5.420 -0.129  -0.129   -0.195   -0.195
77      x7 ~~  x9  5.183 -0.187  -0.187   -0.278   -0.278
```

example: lavResiduals()

```
> lavResiduals(fit)

$type
[1] "cor.bentler"

$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  0.000
x2 -0.030  0.000
x3 -0.008  0.094  0.000
x4  0.071 -0.012 -0.068  0.000
x5 -0.009 -0.027 -0.151  0.005  0.000
x6  0.060  0.030 -0.026 -0.009  0.003  0.000
x7 -0.140 -0.189 -0.084  0.037 -0.036 -0.014  0.000
x8 -0.039 -0.052 -0.012 -0.067 -0.036 -0.022  0.075  0.000
x9  0.149  0.073  0.147  0.048  0.067  0.056 -0.038 -0.032  0.000

$cov.z
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  0.000
x2 -1.996  0.000
x3 -0.997  2.689  0.000
x4  2.679 -0.284 -1.899  0.000
x5 -0.359 -0.591 -4.157  1.545  0.000
x6  2.155  0.681 -0.711 -2.588  0.942  0.000
x7 -3.773 -3.654 -1.858  0.865 -0.842 -0.326  0.000
```

```
x8 -1.380 -1.119 -0.300 -2.021 -1.099 -0.641  4.823  0.000
x9  4.077  1.606  3.518  1.225  1.701  1.423 -2.325 -4.132  0.000

$summary
                          cov
srmr                    0.065
srmr.se                 0.006
srmr.exactfit.z         6.063
srmr.exactfit.pvalue    0.000
usrmr                   0.058
usrmr.se                0.010
usrmr.ci.lower          0.042
usrmr.ci.upper          0.074
usrmr.closefit.h0.value 0.050
usrmr.closefit.z        0.832
usrmr.closefit.pvalue   0.203
```

example: lavTestLRT()

```
> fit0 <- update(fit, orthogonal = TRUE)
> lavTestLRT(fit0, fit)

Chi-Squared Difference Test

     Df    AIC    BIC   Chisq Chisq diff Df diff Pr(>Chisq)
fit  24 7517.5 7595.3  85.305
fit0 27 7579.7 7646.4 153.527     68.222       3  1.026e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.5 Meanstructures

adding the means in lavaan

- when the `meanstructure` argument is set to `TRUE`, a meanstructure is added to the model

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939,
+            meanstructure = TRUE)
```

- if no restrictions are imposed on the means, the fit will be identical to the non-meanstructure fit
- we add p datapoints (the mean vector)
- we add p free parameters (the intercepts of the observed variables)
- we fix the latent means to zero
- the number of degrees of freedom does not change

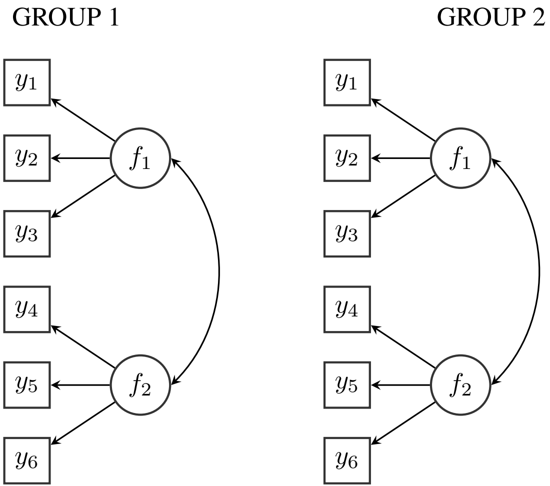
output meanstructure = TRUE

```
lavaan 0.6-12 ended normally after 35 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        30

  Number of observations                           301

Model Test User Model:

  Test statistic                                85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                1.000
    x2                0.554    0.100    5.554    0.000
    x3                0.729    0.109    6.685    0.000
```

```
  textual =~
    x4                1.000
    x5                1.113    0.065   17.014    0.000
    x6                0.926    0.055   16.703    0.000
  speed =~
    x7                1.000
    x8                1.180    0.165    7.152    0.000
    x9                1.082    0.151    7.155    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual ~~
    textual           0.408    0.074    5.552    0.000
    speed             0.262    0.056    4.660    0.000
  textual ~~
    speed             0.173    0.049    3.518    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x1                4.936    0.067   73.473    0.000
   .x2                6.088    0.068   89.855    0.000
   .x3                2.250    0.065   34.579    0.000
   .x4                3.061    0.067   45.694    0.000
   .x5                4.341    0.074   58.452    0.000
   .x6                2.186    0.063   34.667    0.000
   .x7                4.186    0.063   66.766    0.000
   .x8                5.527    0.058   94.854    0.000
   .x9                5.374    0.058   92.546    0.000
```

```
    visual          0.000
    textual         0.000
    speed           0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x1                0.549    0.114    4.833    0.000
   .x2                1.134    0.102   11.146    0.000
   .x3                0.844    0.091    9.317    0.000
   .x4                0.371    0.048    7.779    0.000
   .x5                0.446    0.058    7.642    0.000
   .x6                0.356    0.043    8.277    0.000
   .x7                0.799    0.081    9.823    0.000
   .x8                0.488    0.074    6.573    0.000
   .x9                0.566    0.071    8.003    0.000
    visual            0.809    0.145    5.564    0.000
    textual           0.979    0.112    8.737    0.000
    speed             0.384    0.086    4.451    0.000
```

2.6 Multiple groups



- can we compare the means of the latent variables?

2.7 Measurement invariance

- we can only compare the means of the latent variables across groups if 'measurement invariance' across groups has been established

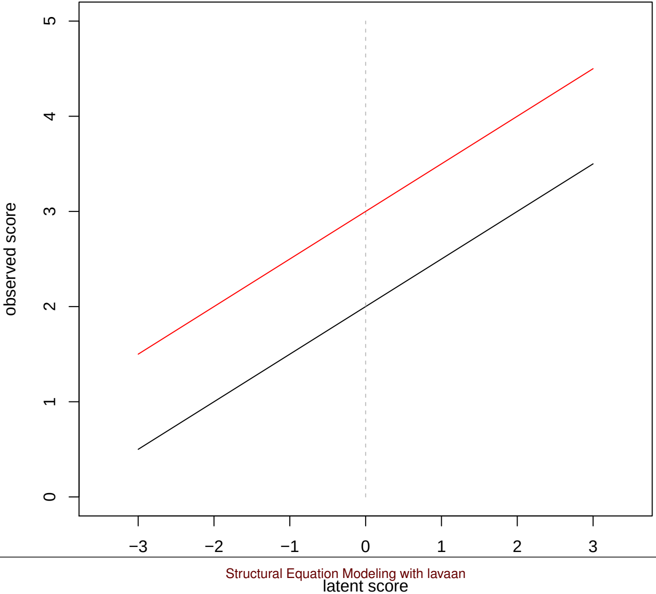
- testing for measurement invariance involves a fixed sequence of model comparison tests

- one typical sequence involves 3 steps:

  1. Model 1: configural invariance. The same factor structure is imposed on all groups.
  2. Model 2: weak invariance. The factor loadings are constrained to be equal across groups.
  3. Model 3: strong invariance. The factor loadings and intercepts are constrained to be equal across groups.

- other sequences involve more steps; for example 'strict invariance' implies constraining the residual variances too

example weak invariance (two groups)



measurement invariance in lavaan - using the group.equal argument

- step 1: fit the configural invariance model (fit1)

```
> fit1 <- cfa(HS.model, data = HolzingerSwineford1939, group = "school")
> fitMeasures(fit1, c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr"))

  chisq       df   pvalue      cfi    rmsea     srmr
115.851   48.000    0.000    0.923    0.097    0.068
```

- step 2: fit the weak invariance model (fit2)

```
> fit2 <- cfa(HS.model, data = HolzingerSwineford1939, group = "school",
+             group.equal = "loadings")
> fitMeasures(fit2, c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr"))

  chisq       df   pvalue      cfi    rmsea     srmr
124.044   54.000    0.000    0.921    0.093    0.072
```

- step 2b: compare with configural invariance model

```
> anova(fit1, fit2)
```

```
Chi-Squared Difference Test

     Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit1 48 7484.4 7706.8 115.85
fit2 54 7480.6 7680.8 124.04     8.1922       6     0.2244
```

- step 3: fit the strong invariance model (fit3)

```
> fit3 <- cfa(HS.model, data = HolzingerSwineford1939, group = "school",
+              group.equal = c("loadings", "intercepts"))
> fitMeasures(fit3, c("chisq", "df", "pvalue", "cfi", "rmsea", "srmr"))

  chisq      df  pvalue     cfi   rmsea    srmr
164.103  60.000   0.000   0.882   0.107   0.082
```

- step 3a: compare with weak invariance model

```
> anova(fit2, fit3)

Chi-Squared Difference Test

     Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit2 54 7480.6 7680.8 124.04
fit3 60 7508.6 7686.6 164.10     40.059       6  4.435e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

output strong invariance model

```
lavaan 0.6-12 ended normally after 60 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        63
  Number of equality constraints                    15

  Number of observations per group:
    Pasteur                                        156
    Grant-White                                    145

Model Test User Model:

  Test statistic                               164.103
  Degrees of freedom                                60
  P-value (Chi-square)                           0.000
  Test statistic for each group:
    Pasteur                                     90.210
    Grant-White                                 73.892

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Expected
  Information saturated (h1) model          Structured
```

```
Group 1 [Pasteur]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                1.000
    x2      (.p2.)    0.576    0.101    5.713    0.000
    x3      (.p3.)    0.798    0.112    7.146    0.000
  textual =~
    x4                1.000
    x5      (.p5.)    1.120    0.066   16.965    0.000
    x6      (.p6.)    0.932    0.056   16.608    0.000
  speed =~
    x7                1.000
    x8      (.p8.)    1.130    0.145    7.786    0.000
    x9      (.p9.)    1.009    0.132    7.667    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual ~~
    textual           0.410    0.095    4.293    0.000
    speed             0.178    0.066    2.687    0.007
  textual ~~
    speed             0.180    0.062    2.900    0.004

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
```

```
   .x1       (.25.)    5.001    0.090   55.760    0.000
   .x2       (.26.)    6.151    0.077   79.905    0.000
   .x3       (.27.)    2.271    0.083   27.387    0.000
   .x4       (.28.)    2.778    0.087   31.953    0.000
   .x5       (.29.)    4.035    0.096   41.858    0.000
   .x6       (.30.)    1.926    0.079   24.426    0.000
   .x7       (.31.)    4.242    0.073   57.975    0.000
   .x8       (.32.)    5.630    0.072   78.531    0.000
   .x9       (.33.)    5.465    0.069   79.016    0.000
    visual             0.000
    textual            0.000
    speed              0.000

Variances:
                    Estimate  Std.Err  z-value  P(>|z|)
   .x1                 0.555    0.139    3.983    0.000
   .x2                 1.296    0.158    8.186    0.000
   .x3                 0.944    0.136    6.929    0.000
   .x4                 0.445    0.069    6.430    0.000
   .x5                 0.502    0.082    6.136    0.000
   .x6                 0.263    0.050    5.264    0.000
   .x7                 0.888    0.120    7.416    0.000
   .x8                 0.541    0.095    5.706    0.000
   .x9                 0.654    0.096    6.805    0.000
    visual             0.796    0.172    4.641    0.000
    textual            0.879    0.131    6.694    0.000
    speed              0.322    0.082    3.914    0.000
```

```
Group 2 [Grant-White]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                1.000
    x2      (.p2.)    0.576    0.101    5.713    0.000
    x3      (.p3.)    0.798    0.112    7.146    0.000
  textual =~
    x4                1.000
    x5      (.p5.)    1.120    0.066   16.965    0.000
    x6      (.p6.)    0.932    0.056   16.608    0.000
  speed =~
    x7                1.000
    x8      (.p8.)    1.130    0.145    7.786    0.000
    x9      (.p9.)    1.009    0.132    7.667    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual ~~
    textual           0.427    0.097    4.417    0.000
    speed             0.329    0.082    4.006    0.000
  textual ~~
    speed             0.236    0.073    3.224    0.001

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
```

```
   .x1      (.25.)    5.001    0.090   55.760    0.000
   .x2      (.26.)    6.151    0.077   79.905    0.000
   .x3      (.27.)    2.271    0.083   27.387    0.000
   .x4      (.28.)    2.778    0.087   31.953    0.000
   .x5      (.29.)    4.035    0.096   41.858    0.000
   .x6      (.30.)    1.926    0.079   24.426    0.000
   .x7      (.31.)    4.242    0.073   57.975    0.000
   .x8      (.32.)    5.630    0.072   78.531    0.000
   .x9      (.33.)    5.465    0.069   79.016    0.000
    visual           -0.148    0.122   -1.211    0.226
    textual           0.576    0.117    4.918    0.000
    speed            -0.177    0.090   -1.968    0.049

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x1                0.654    0.128    5.094    0.000
   .x2                0.964    0.123    7.812    0.000
   .x3                0.641    0.101    6.316    0.000
   .x4                0.343    0.062    5.534    0.000
   .x5                0.376    0.073    5.133    0.000
   .x6                0.437    0.067    6.559    0.000
   .x7                0.625    0.095    6.574    0.000
   .x8                0.434    0.088    4.914    0.000
   .x9                0.522    0.086    6.102    0.000
    visual            0.708    0.160    4.417    0.000
    textual           0.870    0.131    6.659    0.000
    speed             0.505    0.115    4.379    0.000
```

2.8 Missing data

missing data in lavaan

- in lavaan 0.6, the default is listwise deletion (but this may change in future versions)
  - the goal is to alert the user that data is missing
  - the user should examine: which variables have many missing values? is there a pattern?
- available approaches in lavaan:
  - 'full information' ML (`missing = "fiml"`)
  - two-stage approach (`missing = "two.stage"`)
- multiple imputation in lavaan:
  - create imputed datasets (eg., using the `mice` package) + `lavaanList()`
  - the `runMI()` function in the `semTools` package

example: lavaan + listwise

```
> fit <- cfa(HS.model, data = HS.missing, missing = "listwise")
> fit

lavaan 0.6-12 ended normally after 36 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        21

                                                  Used       Total
  Number of observations                           259         301

Model Test User Model:

  Test statistic                                89.144
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000
```

example: lavaan + fiml

```
> fit <- cfa(HS.model, data = HS.missing, missing = "fiml") # or missing = "ml"
> fit

lavaan 0.6-12 ended normally after 54 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        30

  Number of observations                           301
  Number of missing patterns                        13

Model Test User Model:

  Test statistic                                86.624
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000
```

example: lavaan + two.stage

```
> fit <- cfa(HS.model, data = HS.missing, missing = "two.stage")
> fit

lavaan 0.6-12 ended normally after 37 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        30

  Number of observations                           301
  Number of missing patterns                        13

Model Test User Model:
                                              Standard      Robust
  Test Statistic                                91.404      87.698
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  1.042
    Satorra-Bentler correction
```

- a robust test statistic and robust standard errors are needed to take the two-stage estimation process into account
- outperforms ‘fiml’ in the non-normal case (see Savalei & Falk, 2014)

2.9 Nonnormal data and alternative estimators

what if the data are NOT normally distributed?

- in the real world, data may never be normally distributed
- two types:
  - categorical and/or limited-dependent outcomes: binary, ordinal, nominal, counts, censored (WLSMV, logit/probit)
  - continuous outcomes, not normally distributed: skewed, too flat/too peaked (kurtosis), ...
- three strategies to deal with continuous non-normal data
  1. asymptotically distribution-free estimation
  2. ML estimation with 'robust' standard errors, and a 'robust' test statistic for model evaluation
  3. bootstrapping

robust method 1: asymptotically distribution-free (ADF) estimation

- the ADF estimator (Browne, 1984) makes no assumption of normality and is part of a larger family of estimators called weighted least squares (WLS) estimators:

$$F_{WLS} = (\mathbf{s} - \hat{\boldsymbol{\sigma}})^\top \mathbf{W}^{-1} (\mathbf{s} - \hat{\boldsymbol{\sigma}})$$

  where $\mathbf{s}$ and $\hat{\boldsymbol{\sigma}}$ are vectors containing the non-duplicated elements in the sample ($\mathbf{S}$) and model-implied ($\hat{\mathbf{\Sigma}}$) covariance matrix respectively

- the weight matrix $\mathbf{W}$ utilized with the ADF estimator is the asymptotic covariance matrix: a matrix of the covariances of the observed sample variances and covariances

- unfortunately, empirical research has shown that the ADF method breaks down unless the sample size is huge (e.g., N > 5000)

- in lavaan:

```
fit <- cfa(HS.model, data = HolzingerSwineford1939, estimator = "WLS")
```

robust method 2: robust ML

1. **parameter estimates: vanilla ML**
    - if ML is used, the parameter estimates are still consistent (if the model is identified and correctly specified)
2. **'robust' standard errors**
    - if data is non-normal, the standard errors tend to be too small (as much as 25-50%)
    - 'robust' standard errors correct for non-normality
3. **'robust' scaled (chi-square) test statistic**
    - if data is non-normal, the usual model (chi-square) test statistic tends to be too large
    - the **Satorra-Bentler scaled test statistic** rescales the value of the ML-based chi-square test statistic by an amount that reflects the degree of kurtosis

robust ML in lavaan

- robust standard errors

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           se = "robust")
```

- Satorra-Bentler scaled test statistic

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           test = "Satorra-Bentler")
```

- robust standard errors + scaled test statistic

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           se = "robust", test = "Satorra-Bentler")
```

- estimator MLM = robust standard errors + scaled test statistic

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           estimator = "MLM")
```

- alternative: estimator MLR (also for missing data)

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           estimator = "MLR", missing = "ml")
```

output: robust standard errors and scaled test statistic (MLM)

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939, estimator = "MLM")
> summary(fit)

lavaan 0.6-12 ended normally after 35 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        21

  Number of observations                           301

Model Test User Model:
                                              Standard      Robust
  Test Statistic                                85.306      80.872
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  1.055
    Satorra-Bentler correction

Parameter Estimates:

  Standard errors                           Robust.sem
  Information                                 Expected
  Information saturated (h1) model          Structured

Latent Variables:
```

```
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                1.000
    x2                0.554    0.103    5.359    0.000
    x3                0.729    0.115    6.367    0.000
  textual =~
    x4                1.000
    x5                1.113    0.066   16.762    0.000
    x6                0.926    0.060   15.497    0.000
  speed =~
    x7                1.000
    x8                1.180    0.152    7.758    0.000
    x9                1.082    0.132    8.169    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual ~~
    textual           0.408    0.082    4.966    0.000
    speed             0.262    0.055    4.762    0.000
  textual ~~
    speed             0.173    0.055    3.139    0.002

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .x1                0.549    0.138    3.968    0.000
   .x2                1.134    0.107   10.554    0.000
   .x3                0.844    0.085    9.985    0.000
   .x4                0.371    0.050    7.423    0.000
```

```
   .x5              0.446    0.058    7.688    0.000
   .x6              0.356    0.046    7.700    0.000
   .x7              0.799    0.079   10.168    0.000
   .x8              0.488    0.074    6.567    0.000
   .x9              0.566    0.068    8.332    0.000
    visual          0.809    0.167    4.837    0.000
    textual         0.979    0.121    8.109    0.000
    speed           0.384    0.083    4.635    0.000
```

robust method 3: bootstrapping

- bootstrapping standard errors:

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           se = "bootstrap", verbose = TRUE, bootstrap = 1000)
```

- bootstrapping the test statistic

```
fit <- cfa(HS.model, data = HolzingerSwineford1939,
           test = "bootstrap", verbose = TRUE, bootstrap = 1000)
```

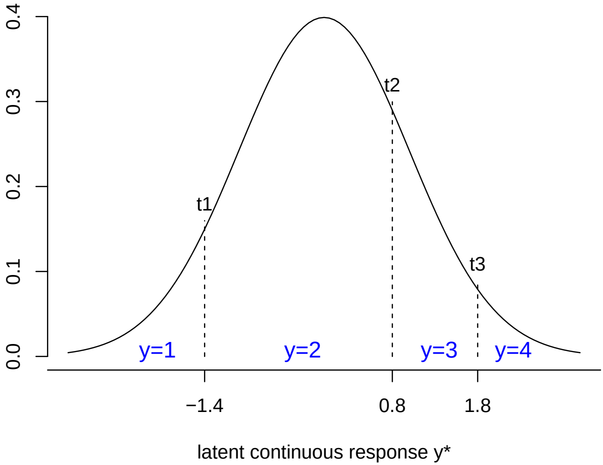
- when we use se = ”bootstrap”, the parameterEstimates() output will contain bootstrap based confidence intervals

2.10 Categorical data

- limited information approach
  - only univariate and bivariate information is used
  - estimation often proceeds in two or three stages; the first stages use maximum likelihood, the last stage uses (weighted) least squares
  - mainly developed in the SEM literature
  - perhaps the best known implementation is in Mplus (WLSMV)
- full information approach
  - all information is used
  - most practical: marginal maximum likelihood estimation
  - requires numerical integration (number of dimensions = number of latent variables)
  - mainly developed in the IRT literature (and GLMM literature)
  - only recently incorporated in modern SEM software

WLS(MV) approach: stage 1 – estimating the thresholds

- an observed variable y can often be viewed as a partial observation of a latent continuous response $y^\star$; eg ordinal variable with $K = 4$ response categories:



stage 2 – estimating tetrachoric, polychoric, . . . , correlations

- estimate tetrachoric/polychoric/. . . correlation from bivariate data:
  - tetrachoric (binary – binary)
  - polychoric (ordered – ordered)
  - polyserial (ordered – numeric)
  - biserial (binary – numeric)
  - pearson (numeric – numeric)
- ML estimation is available (see eg. Olsson 1979 and 1982)
  - two-step: first estimate thresholds using univariate information only; then, keeping the thresholds fixed, estimate the correlation
  - one-step: estimate thresholds and correlation simultaneously
- if exogenous covariates are involved, the correlations are based on the residual values of $y^\star$ (eg bivariate probit regression)

stage 3 – estimating the SEM model

- third stage uses weighted least squares:

$$F_{WLS} = (\mathbf{s} - \hat{\boldsymbol{\sigma}})^\top \mathbf{W}^{-1} (\mathbf{s} - \hat{\boldsymbol{\sigma}})$$

  where $\mathbf{s}$ and $\hat{\boldsymbol{\sigma}}$ are vectors containing all relevant sample-based and model-based statistics respectively

- $\mathbf{s}$ contains: thresholds, correlations, optionally regression slopes of exogenous covariates, optionally variances and means of continuous variables
- the weight matrix $\mathbf{W}$ is (a consistent estimator of) the asymptotic covariance matrix of the sample statistics ($\mathbf{s}$)
- robust version: WLSMV
  - use the diagonal of $\mathbf{W}$ only for estimation (DWLS)
  - use the full matrix for inference (standard errors and test statistic)
  - 'MV' stands for the Satterthwaite's mean and variance corrected test statistic

example

```
> # binary version of Holzinger & Swineford
> HS9 <- HolzingerSwineford1939[,c("x1","x2","x3","x4","x5",
+                                  "x6","x7","x8","x9")]
> HSbinary <- as.data.frame( lapply(HS9, cut, 2, labels = FALSE) )

> # single factor model
> model <- ' visual  =~ x1 + x2 + x3
+            textual =~ x4 + x5 + x6
+            speed   =~ x7 + x8 + x9 '

> # method 1: list all the `ordered' observed variables
> fit <- cfa(model, data = HSbinary, ordered = c("x1","x2","x3","x4","x5",
+                                                "x6","x7","x8","x9"))
> #
> # method 2: these are also the names of the data.frame `HSbinary'
> fit <- cfa(model, data = HSbinary, ordered = names(HSbinary))
> #
> # method 3: because ALL observed variables are ordered, we can use the shortcut:
> fit <- cfa(model, data = HSbinary, ordered = TRUE)
```

output

```
> summary(fit, fit.measures = TRUE, standardized = TRUE)

lavaan 0.6-12 ended normally after 35 iterations

  Estimator                                         DWLS
  Optimization method                             NLMINB
  Number of model parameters                          21

  Number of observations                             301

Model Test User Model:
                                              Standard      Robust
  Test Statistic                                30.918      38.427
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.156       0.031
  Scaling correction factor                                  0.869
  Shift parameter                                            2.861
    simple second-order correction

Model Test Baseline Model:

  Test statistic                               582.533     468.233
  Degrees of freedom                                36          36
  P-value                                        0.000       0.000
  Scaling correction factor                                  1.264
```

```
User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.987       0.967
  Tucker-Lewis Index (TLI)                       0.981       0.950

  Robust Comparative Fit Index (CFI)                            NA
  Robust Tucker-Lewis Index (TLI)                               NA

Root Mean Square Error of Approximation:

  RMSEA                                          0.031       0.045
  90 Percent confidence interval - lower         0.000       0.014
  90 Percent confidence interval - upper         0.059       0.070
  P-value RMSEA <= 0.05                          0.847       0.600

  Robust RMSEA                                                  NA
  90 Percent confidence interval - lower                        NA
  90 Percent confidence interval - upper                        NA

Standardized Root Mean Square Residual:

  SRMR                                           0.083       0.083

Parameter Estimates:

  Standard errors                           Robust.sem
  Information                                 Expected
  Information saturated (h1) model        Unstructured
```

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    x1                1.000                               0.639    0.639
    x2                0.900    0.188    4.788    0.000    0.575    0.575
    x3                0.939    0.197    4.766    0.000    0.600    0.600
  textual =~
    x4                1.000                               0.835    0.835
    x5                0.976    0.118    8.241    0.000    0.815    0.815
    x6                1.078    0.125    8.601    0.000    0.900    0.900
  speed =~
    x7                1.000                               0.471    0.471
    x8                1.569    0.461    3.403    0.001    0.740    0.740
    x9                1.449    0.409    3.541    0.000    0.683    0.683

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual ~~
    textual           0.303    0.061    4.981    0.000    0.569    0.569
    speed             0.132    0.049    2.700    0.007    0.439    0.439
  textual ~~
    speed             0.076    0.046    1.656    0.098    0.192    0.192

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1                0.000                               0.000    0.000
   .x2                0.000                               0.000    0.000
```

```
   .x3              0.000                                   0.000    0.000
   .x4              0.000                                   0.000    0.000
   .x5              0.000                                   0.000    0.000
   .x6              0.000                                   0.000    0.000
   .x7              0.000                                   0.000    0.000
   .x8              0.000                                   0.000    0.000
   .x9              0.000                                   0.000    0.000
    visual          0.000                                   0.000    0.000
    textual         0.000                                   0.000    0.000
    speed           0.000                                   0.000    0.000

Thresholds:
                 Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    x1|t1          -0.388    0.074   -5.223    0.000   -0.388   -0.388
    x2|t1          -0.054    0.072   -0.748    0.454   -0.054   -0.054
    x3|t1           0.318    0.074    4.309    0.000    0.318    0.318
    x4|t1           0.180    0.073    2.473    0.013    0.180    0.180
    x5|t1          -0.257    0.073   -3.506    0.000   -0.257   -0.257
    x6|t1           1.024    0.088   11.641    0.000    1.024    1.024
    x7|t1           0.231    0.073    3.162    0.002    0.231    0.231
    x8|t1           1.128    0.092   12.284    0.000    1.128    1.128
    x9|t1           0.626    0.078    8.047    0.000    0.626    0.626

Variances:
                 Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1              0.592                                   0.592    0.592
   .x2              0.670                                   0.670    0.670
   .x3              0.640                                   0.640    0.640
```

```
   .x4          0.303                              0.303    0.303
   .x5          0.336                              0.336    0.336
   .x6          0.191                              0.191    0.191
   .x7          0.778                              0.778    0.778
   .x8          0.453                              0.453    0.453
   .x9          0.534                              0.534    0.534
    visual      0.408    0.112    3.651    0.000    1.000    1.000
    textual     0.697    0.101    6.883    0.000    1.000    1.000
    speed       0.222    0.094    2.363    0.018    1.000    1.000

Scales y*:
             Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
    x1          1.000                              1.000    1.000
    x2          1.000                              1.000    1.000
    x3          1.000                              1.000    1.000
    x4          1.000                              1.000    1.000
    x5          1.000                              1.000    1.000
    x6          1.000                              1.000    1.000
    x7          1.000                              1.000    1.000
    x8          1.000                              1.000    1.000
    x9          1.000                              1.000    1.000
```

estimated thresholds and tetrachoric correlations

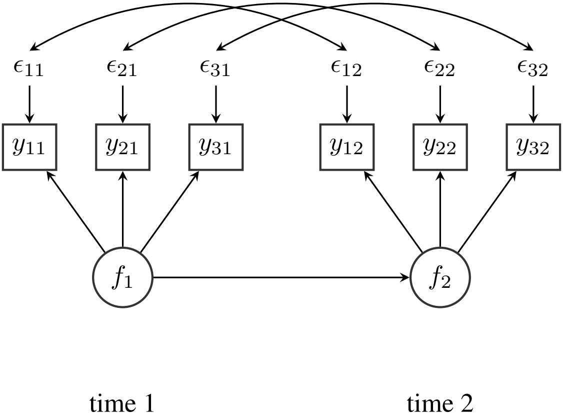
```
> lavInspect(fit, "sampstat")

$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  1.000
x2  0.284  1.000
x3  0.415  0.389  1.000
x4  0.364  0.328  0.232  1.000
x5  0.319  0.268  0.138  0.688  1.000
x6  0.422  0.322  0.206  0.720  0.761  1.000
x7 -0.048  0.061  0.041  0.200  0.023 -0.029  1.000
x8  0.159  0.105  0.439 -0.029 -0.059  0.183  0.464  1.000
x9  0.165  0.210  0.258  0.146  0.183  0.230  0.335  0.403  1.000

$mean
x1 x2 x3 x4 x5 x6 x7 x8 x9
 0  0  0  0  0  0  0  0  0

$th
 x1|t1  x2|t1  x3|t1  x4|t1  x5|t1  x6|t1  x7|t1  x8|t1  x9|t1
-0.388 -0.054  0.318  0.180 -0.257  1.024  0.231  1.128  0.626
```

2.11 Panel models for longitudinal data

- panel models postulate *directional* (regression) relationships among the repeated measures
- both within repeated variables (autoregressive) and between repeated variables (cross-lagged)
- focus on the model-implied covariance/correlation structure
- the means are usually ignored
- some subtypes:
  - autoregressive models (the simplex model)
  - cross-lagged models
  - latent autoregressive/cross-lagged models
  - ...

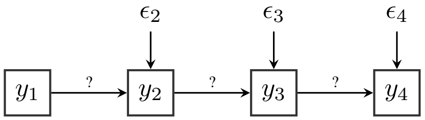
example panel model with a single latent variable

- example with 2 time points:



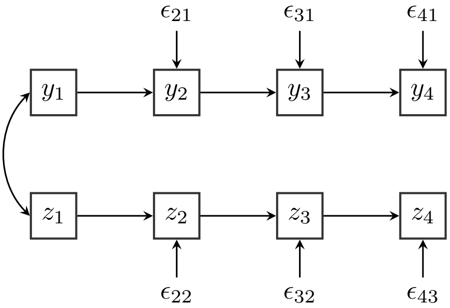
autoregressive models

- each time point is regressed on a previous time point (first order) , or an even further time point (second order, third order, . . .)
- alternative names: Markov models, simplex models, panel models, . . .
- earliest development dates back to the seminal work of Guttman (1954)
- example first-order univariate autoregressive model:



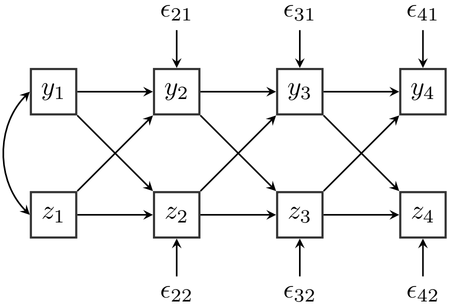
multivariate panel models

- in a multivariate panel model, we have more than one outcome, measured at (the same) t time points
- example: a bivariate panel/simplex model where Y is a measure of mathematical achievement, and Z is a measure of reading ability (4 time points: grade 3, grade 4, grade 5 and grade 6)



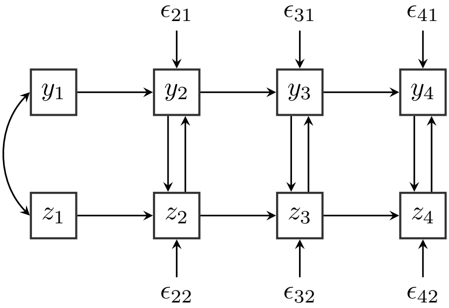
crosslagged effects

- what is the directional effect of one variable on the other?
  - do the two variables develop independently of each other?
  - or does Y exert a greater influence on Z, or vice versa?



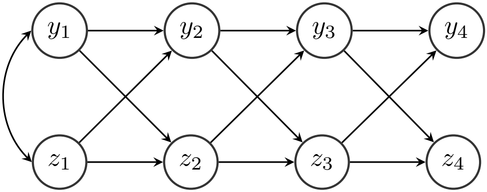
contemporaneous effects

- sometimes, the crossed effects between two variables are not lagged, but contemporaneous (exerting an effect at the same time point)
- this can be unidirectional, or reciprocal
- not everyone believes this approach is useful (in addition: often convergence issues)



panel model with latent variables

- if the 'repeated' outcomes are not directly observable, we may replace them with a latent variable with a proper measurement model
- but first, we need to establish 'measurement invariance' for the latent variables across time



- in this diagram, the observed indicators have been omitted

2.12 Growth curve models

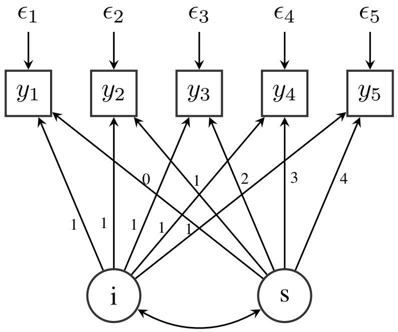
- 'time' is typically considered as a continuous variable
- two components:
  - fixed effects: what is the nature of the average trend (linear, quadratic)
  - random effects: individual differences
- in addition, we may try to explain these individual differences by taking into account:
  - time-invariant covariates (age, gender, ...)
  - time-varying covariates (measured at each time point)
- closely related to 'mixed models' (linear mixed models, generalized mixed models)
  - limited to balanced data
  - but we can add indirect paths and latent variables
- focus on the mean structure (not the covariance structure)

some references

- Bollen, K.A., & Curran, P.J. (2006). *Latent curve models: A structural equation perspective.* John Wiley & Sons.
- Duncan, T.E., Duncan, S.C., & Strycker, L.A. (2006). *An introduction to latent variable growth curve modeling: Concepts, issues, and applications*. Routledge Academic.
- Preacher, K.J., Wichman, A.L., MacCallum, R.C., & Briggs, N.E. (2008). *Latent Growth Curve Modeling.* Quantitative Applications in the Social Sciences, No. 157, Sage.

a typical growth curve model

- random intercept and random slope



- y_t = (initial time at time 1) + (growth per unit time)*time + error
- y_t = intercept + slope*time + error

example growth curve model: use the lavaan() function

```
> model.syntax <- '
+   # intercept and slope with fixed coefficients
+     i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
+     s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4
+
+   # fixed effects
+     i ~ 1 # intercept
+     s ~ 1 # slope
+
+  # random effects
+     i ~~ i # variance random intercept
+     s ~~ s # variance random slope
+     i ~~ s # covariance random intercept/slope
+ '
> fit <- lavaan(model.syntax, data = Demo.growth, auto.var = TRUE)
```

2.13 Two-level SEM with random intercepts

lavaan syntax setup for two-level SEM

$\boldsymbol{\Sigma}_B$

Between

Within

$\boldsymbol{\Sigma}_W$

```
model <- '

  level: 1

    # here comes the within level

  level: 2

    # here comes the between level
'

fit <- sem(myModel, myData,
           cluster = "school")
```

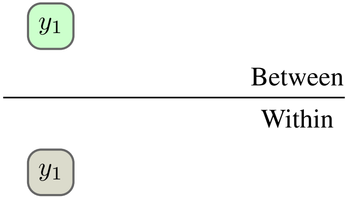
example: Demo.twolevel (simulated data)

- data: 200 clusters, 2500 observations, cluster sizes: 5, 10, 15 and 20
- measures at the within level $y_1, y_2, y_3, \ldots$
- covariates at the within level $x_1, x_2 \ldots$
- covariates at the between level w_1 and w_2
- explore the data:

```
> library(lavaan)
> head(round(Demo.twolevel[,c(1:4,7:12)], 3), n = 9)
```

```
      y1     y2     y3     y4     x1     x2     x3     w1     w2 cluster
1  0.229  1.356 -0.691  0.803  1.174 -0.623  0.647 -0.248 -0.499       1
2  0.309 -1.862 -2.418  0.766 -1.004 -0.567  0.020 -0.248 -0.499       1
3  0.200 -1.340  0.438  1.197 -0.440 -2.134 -0.459 -0.248 -0.499       1
4  1.045 -0.962 -0.446 -0.203 -0.625 -0.337  1.285 -0.248 -0.499       1
5  0.688 -0.457 -0.642  0.990 -0.845 -0.042  1.560 -0.248 -0.499       1
6 -2.069 -0.600  0.315  0.676 -0.783 -0.224 -0.381 -2.322 -0.691       2
7 -0.787 -0.488  1.132 -0.256 -0.178 -0.583  3.748 -2.322 -0.691       2
8  3.454  1.409  0.930  1.280  0.950  0.259  0.709 -2.322 -0.691       2
9  0.599 -0.291 -1.070  1.930 -1.189  0.815 -0.321 -2.322 -0.691       2
```

model 1: the empty (univariate) model

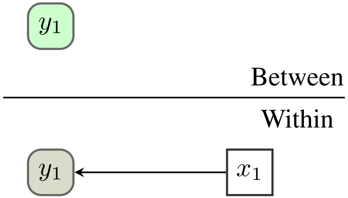



```
library(lavaan)

model <- '
  level: 1
    y1 ~~ y1
  level: 2
    y1 ~~ y1
'

fit <- sem(model,
           data = Demo.twolevel,
           cluster = "cluster")

summary(fit, nd = 4)
```

lavaan output (parameter estimates only)

```
Level 1 []:

Intercepts:
                   Estimate   Std.Err   z-value   P(>|z|)
    y1               0.0000

Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
    y1               2.0003    0.0589   33.9574    0.0000


Level 2 []:

Intercepts:
                   Estimate   Std.Err   z-value   P(>|z|)
    y1               0.0198    0.0755    0.2617    0.7935

Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
    y1               0.9436    0.1124    8.3931    0.0000
```

model 2: simple twolevel regression (predictor within)




```
model <- '

  level: 1

    y1 ~ x1

  level: 2

    y1 ~~ y1

'
```

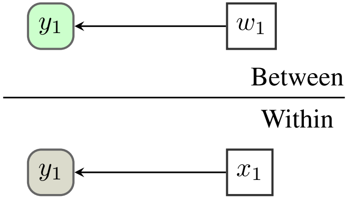
```
fit <- sem(model,
           data = Demo.twolevel,
           cluster = "cluster")
```

```
summary(fit, nd = 4)
```

lavaan output (parameter estimates only)

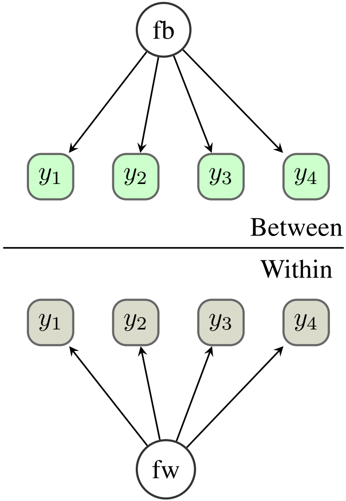
```
Level 1 []:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  y1 ~
    x1               0.4944   0.0276  17.8804   0.0000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.0000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               1.7599   0.0518  33.9532   0.0000


Level 2 []:

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.0222   0.0745   0.2985   0.7653

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.9367   0.1096   8.5436   0.0000
```

model 3: simple twolevel regression (no output)




```
model <- '

  level: 1

    y1 ~ x1

  level: 2

    y1 ~ w1

'



fit <- sem(model,
           data = Demo.twolevel,
           cluster = "cluster")



summary(fit, nd = 4)
```

model 4: one-factor model at both levels




```
model <- '
    level: 1
        fw =~ y1 + y2 + y3 + y4
    level: 2
        fb =~ y1 + y2 + y3 + y4
'

fit <- sem(model,
           data = Demo.twolevel,
           cluster = "cluster")
```

lavaan output

> summary(fit)

```
lavaan 0.6-10 ended normally after 44 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        20

  Number of observations                          2500
  Number of clusters [cluster]                     200

Model Test User Model:

  Test statistic                                 1.274
  Degrees of freedom                                 4
  P-value (Chi-square)                           0.866

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian


Level 1 [within]:
```

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fw =~
    y1                1.000
    y2                0.751    0.042   18.051    0.000
    y3                0.713    0.040   18.034    0.000
    y4                0.315    0.028   11.189    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.000
   .y2                0.000
   .y3                0.000
   .y4                0.000
    fw                0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.949    0.059   15.990    0.000
   .y2                1.081    0.044   24.586    0.000
   .y3                1.024    0.041   25.177    0.000
   .y4                1.080    0.033   32.458    0.000
    fw                1.052    0.074   14.269    0.000


Level 2 [cluster]:

Latent Variables:
```

```
                   Estimate  Std.Err  z-value  P(>|z|)
  fb =~
    y1                1.000
    y2                0.714    0.056   12.801    0.000
    y3                0.579    0.050   11.474    0.000
    y4                0.057    0.094    0.611    0.541

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.020    0.076    0.265    0.791
   .y2               -0.019    0.061   -0.318    0.750
   .y3               -0.045    0.055   -0.817    0.414
   .y4                0.022    0.080    0.280    0.779
    fb                0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.055    0.049    1.122    0.262
   .y2                0.122    0.032    3.805    0.000
   .y3                0.148    0.028    5.272    0.000
   .y4                1.159    0.127    9.111    0.000
    fb                0.891    0.122    7.318    0.000
```

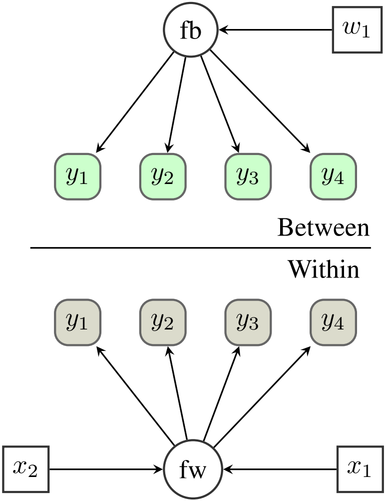
more output

```
> fitMeasures(fit)

               npar                fmin               chisq                  df
             20.000               2.904               1.274               4.000
             pvalue      baseline.chisq         baseline.df     baseline.pvalue
              0.866            1511.382              12.000               0.000
                cfi                 tli                nnfi                 rfi
              1.000               1.005               1.005               0.997
                nfi                pnfi                 ifi                 rni
              0.999               0.333               1.002               1.002
               logl   unrestricted.logl                 aic                 bic
         -16448.595          -16447.958           32937.191           33053.672
             ntotal                bic2               rmsea      rmsea.ci.lower
           2500.000           32990.127               0.000               0.000
     rmsea.ci.upper        rmsea.pvalue                srmr         srmr_within
              0.016               1.000               0.020               0.001
       srmr_between
              0.018

> lavInspect(fit, "h1")

$within
$within$cov
   y1    y2    y3    y4
y1 2.000
y2 0.788 1.673
```

```
y3 0.749 0.564 1.557
y4 0.333 0.250 0.231 1.184

$within$mean
    y1     y2     y3     y4
 0.001 -0.002 -0.001  0.002


$cluster
$cluster$cov
   y1    y2    y3    y4
y1 0.946
y2 0.635 0.575
y3 0.517 0.368 0.448
y4 0.048 0.019 0.069 1.163

$cluster$mean
    y1     y2     y3     y4
 0.019 -0.017 -0.044  0.020

> lavInspect(fit, "implied")

$within
$within$cov
   y1    y2    y3    y4
y1 2.000
y2 0.789 1.673
y3 0.749 0.562 1.558
```

```
y4 0.331 0.248 0.236 1.184

$within$mean
y1 y2 y3 y4
 0  0  0  0


$cluster
$cluster$cov
   y1    y2    y3    y4
y1 0.946
y2 0.636 0.576
y3 0.516 0.368 0.447
y4 0.051 0.036 0.030 1.162

$cluster$mean
    y1     y2     y3     y4
 0.020 -0.019 -0.045  0.022

> lavInspect(fit, "icc")

   y1    y2    y3    y4
0.321 0.256 0.223 0.495
```

model 5: adding covariates (no output)




```
model <- '

    level: 1

        fw =~ y1 + y2 + y3 + y4
        fw ~ x1 + x2

    level: 2

        fb =~ y1 + y2 + y3 + y4
        fb ~ w1
'

fit <- sem(model,
           data = Demo.twolevel,
           cluster = "cluster")
```

. . . and our time is up.

Thank you for following this workshop.

Thank you for choosing open-source software.