Multilevel Structural Equation Modeling with lavaan

Yves Rosseel
Department of Data Analysis
Ghent University

University of Zurich – 2 + 4 November 2020

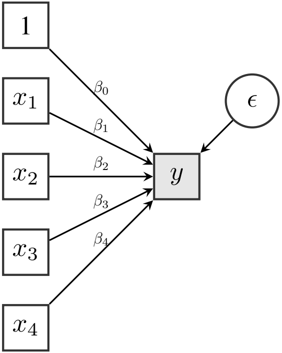
Contents

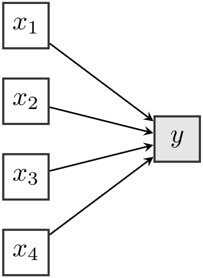
1 Introduction to SEM

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

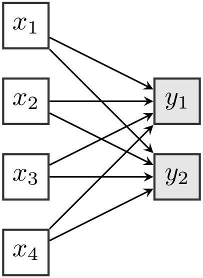
univariate linear regression





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

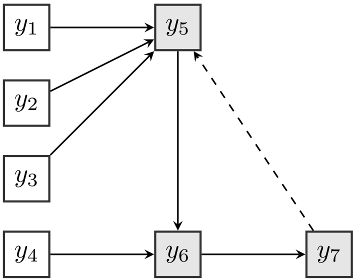
multivariate regression



- strict distinction between 'dependent' variables and 'independent' variables

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)



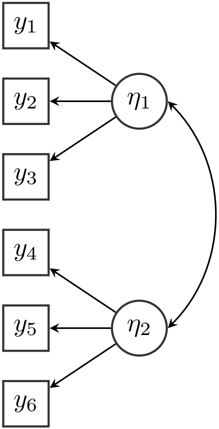
y_5 = reading motivation

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

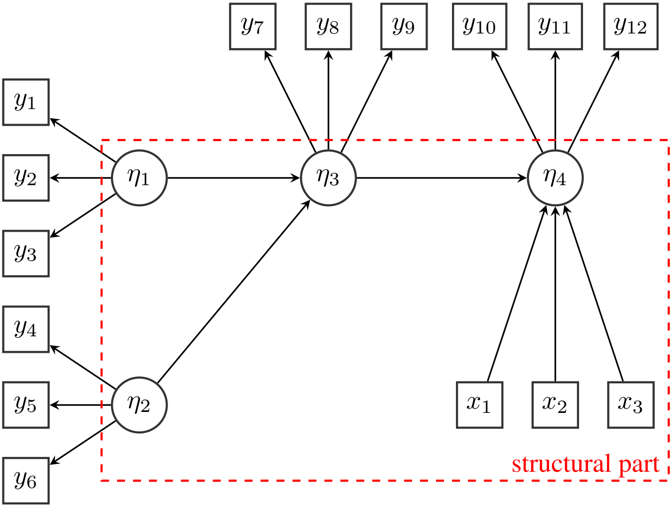


η_1 = depression

η_2 = neuroticism

structural equation modeling (SEM)

- path analysis with latent variables



1.2 How does SEM work?

a dataset: the Holzinger & Swineford dataset

- this is a 'classic' dataset, based on data collected by Holzinger & Swineford (1939)
- scores on 26 'Mental Ability tests' of seventh- and eighth-grade children from two different schools (Pasteur and Grant-White)
- the dataset was used in a seminal paper about CFA (Jöreskog, 1969)
- just like Jöreskog (1969), we will use a subset of 9 scores: x1 = Visual perception, x2 = Cubes, x3 = Lozenges, x4 = Paragraph comprehension, x5 = Sentence completion, x6 = Word meaning, x7 = Speeded addition, x8 = Speeded counting of dots, x9 = Speeded discrimination
- these 9 scores are often regarded as indicators of 3 latent variables: 'visual intelligence' (x1, x2, x3), 'textual intelligence' (x4, x5, x6), en 'speed' (x7, x8, x9)
- we will investigate this later using CFA

reading in data + descriptives

```
> library(lavaan)
> dim(HolzingerSwineford1939)

[1] 301  15


> 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)
```

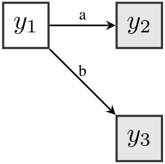
```
      x1     x2    x3    x4    x5    x6     x7    x8    x9
x1 1.358  0.407 0.580 0.505 0.441 0.455  0.085 0.264 0.458
x2 0.407  1.382 0.451 0.209 0.211 0.248 -0.097 0.110 0.244
x3 0.580  0.451 1.275 0.208 0.112 0.244  0.088 0.212 0.374
x4 0.505  0.209 0.208 1.351 1.098 0.896  0.220 0.126 0.243
x5 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:



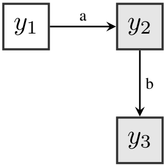
$$y_2 = a\, y_1 + \epsilon_2$$
$$y_3 = b\, y_1 + \epsilon_3$$

- 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:



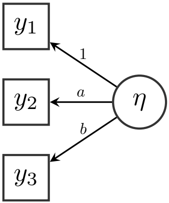
we find

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

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

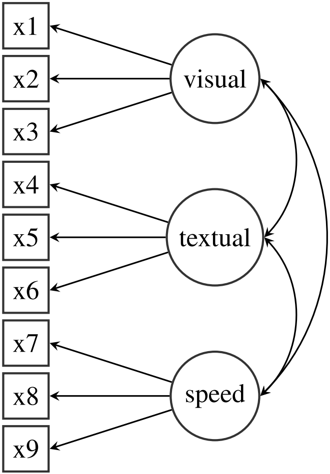
$$\boldsymbol{\Sigma} = \begin{bmatrix} 11 & & \\ 4 & 36 & \\ 5 & 20 & 55 \end{bmatrix}$$

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

1.3 A first example: a CFA with three factors

- for this example, we use the Holzinger & Swineford (1939) data
- we postulate a CFA with three latent variables (‘factors’):
  - a ‘visual’ factor measured by x1, x2 and x3
  - a ‘textual’ factor measured by x4, x5 and x6
  - a ‘speed’ factor measured by x7, x8 and x9
- we assume the three factors are correlated
- the next slide shows a path diagram of this model
- we will discuss later how we can ‘fit’ this model using SEM software
- in the next subsection, we introduce the matrix representation of a CFA model, in order to have a convenient way to compute the model-implied variance-covariance matrix

diagram of the model



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

1.4 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:

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

- 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:

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

- 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:

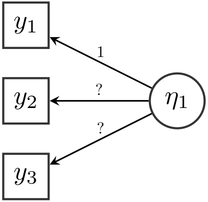
$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Psi}\boldsymbol{\Lambda}' + \boldsymbol{\Theta}$$

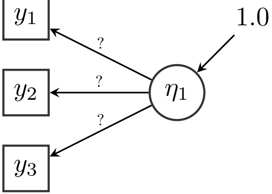
- all parameters are included in three model matrices
- simple matrix multiplication (and addition) gives us the model implied covariance matrix
- for identification purposes, some parameters need to be fixed to a constant (see next slide)
- estimation problem: choose the 'free' parameters, so that the estimated implied covariance matrix ($\hat{\boldsymbol{\Sigma}}$) 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

number of free parameters and degrees of freedom

- in our example, we have used ULI: the first factor loading (of each latent variable) was fixed to 1.0
- therefore, we only have 21 free parameters in our model:
  - 6 factor loadings
  - 3 variances for the factors
  - 3 covariances between the factors
  - 9 residual variances for the indicators
- our sample variance-covariance matrix ($\mathbf{S}$) contains $P(P+1)/2 = 45$ (non-redundant) elements ('sample statistics')
- the difference between the number of sample statistics and the number of free parameters is called the 'degrees of freedom' of the model; for this model, we have $45 - 21 = 24$ degrees of freedom (df = 24)
- the number of free parameters cannot exceed the number of sample statistics; if df = 0, we say the model is 'saturated' because in this case $\hat{\mathbf{\Sigma}} = \mathbf{S}$

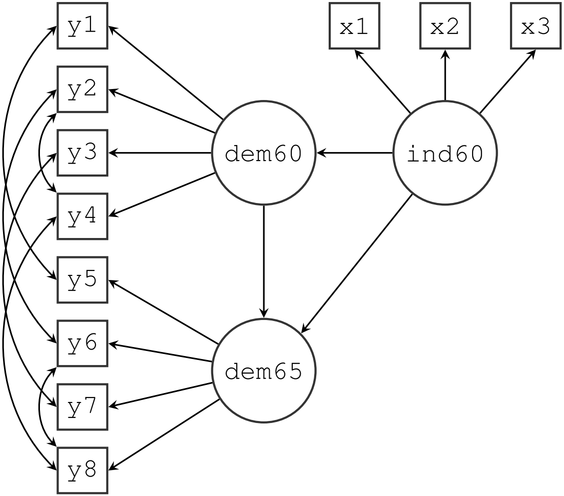
1.5 A second example: the Political Democracy dataset

- data from $N = 75$ developing countries regarding the amount of 'industrialization' (in 1960) and the level of 'political democracy' (in 1960, and again in 1965)
- this dataset is used throughout Bollen's 1989 book
- overview of the observed variables (indicators):

```
y1: Expert ratings of the freedom of the press in 1960
y2: The freedom of political opposition in 1960
y3: The fairness of elections in 1960
y4: The effectiveness of the elected legislature in 1960
y5: Expert ratings of the freedom of the press in 1965
y6: The freedom of political opposition in 1965
y7: The fairness of elections in 1965
y8: The effectiveness of the elected legislature in 1965
x1: The gross national product (GNP) per capita in 1960
x2: The inanimate energy consumption per capita in 1960
x3: The percentage of the labor force in industry in 1960
```

- three latent variables: `ind60`, measured by x1, x2 and x3; `dem60`, measured by y1, y2, y3 and y4; `dem65` measured by y5, y6, y7 en y8

model diagram



preview of (a selection of) the lavaan output

```
Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  ind60 =~
    x1                1.000
    x2                2.180    0.139   15.742    0.000
    x3                1.819    0.152   11.967    0.000
  dem60 =~
    y1                1.000
    y2                1.257    0.182    6.889    0.000
    y3                1.058    0.151    6.987    0.000
    y4                1.265    0.145    8.722    0.000
  dem65 =~
    y5                1.000
    y6                1.186    0.169    7.024    0.000
    y7                1.280    0.160    8.002    0.000
    y8                1.266    0.158    8.007    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  dem60 ~
    ind60             1.483    0.399    3.715    0.000
  dem65 ~
    ind60             0.572    0.221    2.586    0.010
    dem60             0.837    0.098    8.514    0.000
```

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

1.6 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

1.7 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)

evaluation of fit – new developments

- renewed attention for SRMR; see for example

  Maydeu-Olivares, A. (2017). Assessing the size of model misfit in structural equation models. *Psychometrika, 82*, 533–558

- the SRMR is (more or less) the 'average' of the (standardized) squared residuals (e.g., between the elements of $\mathbf{S}$ and $\mathbf{\Sigma}$); the CRMR converts first to correlation matrices
- unlike other fit measures, SRMR/CRMR has a straightforward interpretation
- an unbiased estimate is available, as well as a standard error, and a confidence interval
- another approach is to focus on 'local' fit measures: looking at just one part of the model; see for example

  Thoemmes, F., Rosseel, Y., & Textor, J. (2018). Local fit evaluation of structural equation models using graphical criteria. *Psychological methods, 23*, 27–41.

admissibility of the results

- are the parameter values valid? Often a sign of a bad-fitting model
  - negative (residual) variances
  - correlations larger than one
- have the regression coefficients, factor loadings, covariances the proper (expected) sign (positive or negative)?
- are all free parameters significant?
- are there any excessively large standard errors?

1.8 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)

1.9 Reporting your results

- see Boomsma (2000)
- report enough information so that the analysis can be replicated
  - always report the observed covariance matrix (or the correlation matrix + standard deviations)
  - or make sure the full dataset is available (either as an electronic appendix or via a website)

1.10 Further reading

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

> *...The companion website supplies data, syntax, and output for the book's examples–now including files for Amos, EQS, LISREL, Mplus, Stata, and R (lavaan).*

Brown, T. A. (2015). Confirmatory Factor Analysis for Applied Research (Second Edition) New York: Guilford Press.

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

Hancock, G. R., & Mueller, R. O. (Eds.). (2013). Structural equation modeling: A second course (Second Edition). Greenwich, CT: Information Age Publishing, Inc.

Boomsma, A. (2000). Reporting Analyses of Covariance Structures. *Structural Equation Modeling: A Multidisciplinary Journal, 7*, 461–483.

SEM in R, using lavaan

Gana, K., & Broc, G. (2019). Structural Equation Modeling with Lavaan. London: Wiley-ISTE.

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.

Little, T.D. (2013). Longitudinal Structural Equation Modeling (Methodology in the Social Sciences). The Guilford Press.

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, ...
- R packages: sem, OpenMx, lavaan, lava

2.2 The R package ‘lavaan’

what is lavaan?

- **lavaan** is an R package for latent variable analysis:
  - general mean/covariance structure modeling: function `lavaan()`
  - user-friendly interface: function `sem()` or `cfa()`
  - support for continuous, binary and ordinal data
  - support for missing data, multiple groups, clustered data, ...
- under development, future plans:
  - EFA, ESEM, mixture/latent-class SEM, IRT, new engine, ...
- the long-term goal of **lavaan** is
  1. to implement all the state-of-the-art capabilities that are currently available in commercial packages
  2. to provide a modular and extensible platform that allows for easy implementation and testing of new statistical and modeling ideas

installing lavaan, finding documentation

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

  `http://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

the lavaan ecosystem

- **blavaan** (Ed Merkle, Yves Rosseel)

  Bayesian SEM (using jags or stan) with a lavaan interface

- **lavaan.survey** (Daniel Oberski)

  survey weights, clustering, strata, and finite sampling corrections in SEM

- **Onyx** (Timo von Oertzen, Andreas M. Brandmaier, Siny Tsang)

  interactive graphical interface for SEM (written in Java)

- **semTools** (Terrence Jorgensen and many others)

  collection of useful functions for SEM

- **simsem** (Terrence Jorgensen and many others)

  simulation of SEM models

the lavaan ecosystem (2)

- **semPlot** (Sacha Epskamp)

  visualizations of SEM models

- **EffectLiteR** (Axel Mayer, Lisa Dietzfelbinger)

  using SEM to estimate average and conditional effects

- **MIIVsem** (Zachary Fisher, Kenneth Bollen, and others)

  Functions for estimating structural equation models using instrumental variables.

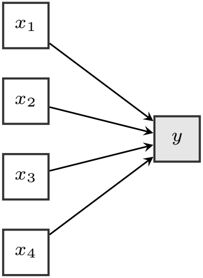
- many others

  bmem, coefficientalpha, eqs2lavaan, fSRM, influence.SEM, nlsem, profileR, RAMpath, regsem, RMediation, RSA, rsem, stremo, faoutlier, gimme, lavaan.shiny, matrixpls, MBESS, NlsyLinks, nonnest2, piecewiseSEM, pscore, psytabs, qgraph, sesem, sirt, TAM, userfriendlyscience, ...

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:

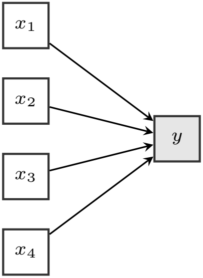
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$$

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)
```

- 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 '
```

```
lavaan 0.6-7 ended normally after 32 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          5

  Number of observations                           100

Model Test User Model:

  Test statistic                                0.0000
  Degrees of freedom                                 0

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
```

small note: why are the standard errors (slightly) different?

- recall that in a linear model, the standard error for b_j is computed by

$$\text{SE}(b_j) = \sqrt{\hat{\sigma}_y^2 \left[(\mathbf{X}'\mathbf{X})^{-1}\right]_{jj}}$$

- in the least-squares approach, $\hat{\sigma}_y^2$ (the residual variance of Y) is computed by:

$$\hat{\sigma}_y^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n - (p+1)}$$

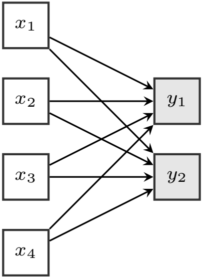
- if maximum likelihood is used, $\hat{\sigma}_y^2$ is computed by:

$$\hat{\sigma}_y^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

and this affects the standard errors.

the lavaan model syntax – multivariate regression

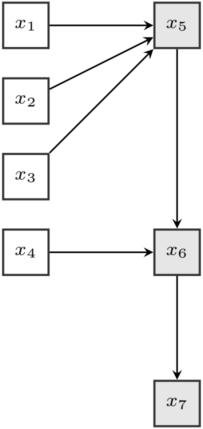
- for each dependent variable, we write a separate regression equation:




```
myModel <- ' y1 ~ x1 + x2 + x3 + x4
             y2 ~ x1 + x2 + x3 + x4 '
```

the lavaan model syntax – path analysis

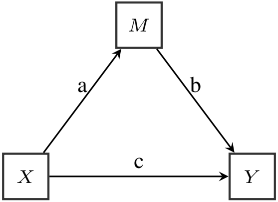
- for each dependent variable, we write a separate regression equation:




```
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

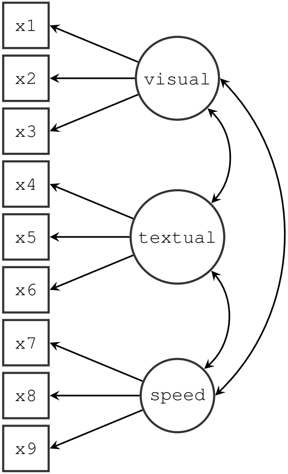



```
myModel <- '
             Y ~ b*M + c*X
             M ~ a*X

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

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

summary(fit)
```

partial output

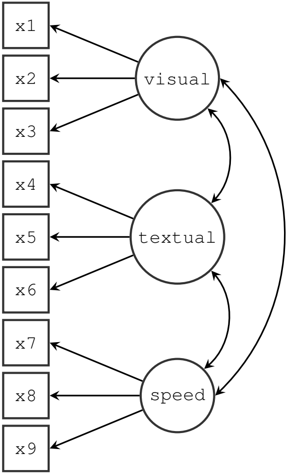
```
Parameter estimates:

  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

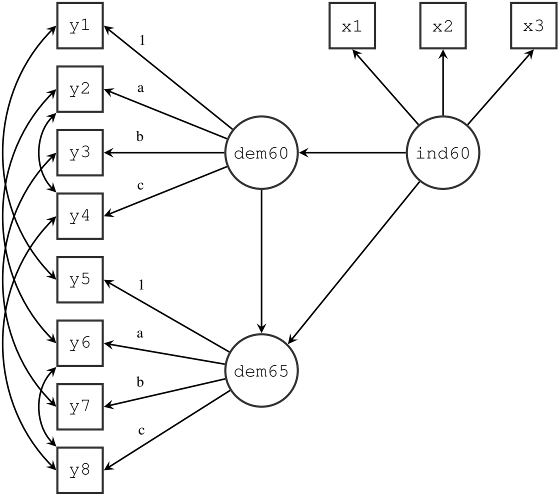
```
lavaan 0.6-7 ended normally after 35 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free 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 0.6-7 ended normally after 66 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free 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
```

3 Multiple groups and measurement invariance

3.1 Meanstructures

- traditionally, SEM has focused on covariance structure analysis
- but we can also include the means
- typical situations where we would include the means are:
  - multiple group analysis
  - growth curve models
  - analysis of non-normal data, and/or missing data
- we have more data: the p-dimensional mean vector
- we have more parameters:
  - means/intercepts for the observed variables
  - means/intercepts for the latent variables (often fixed to zero)

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

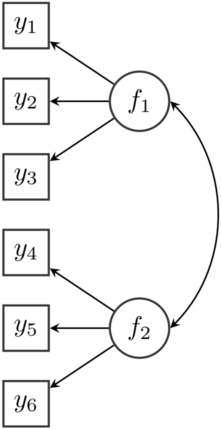
```
lavaan 0.6-5 ended normally after 35 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free 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:

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

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
```

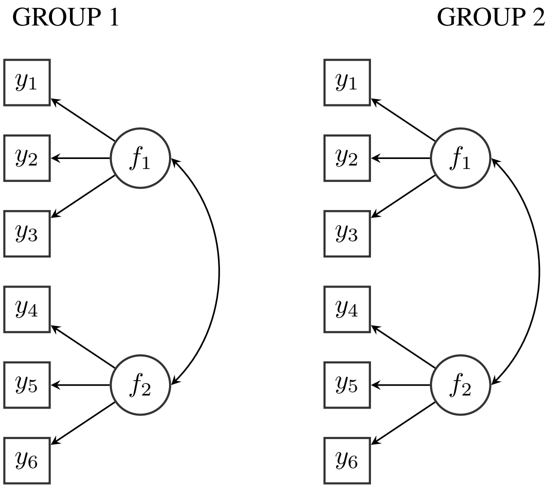
3.2 Multiple groups

single group analysis (CFA)



- factor means typically fixed to zero

multiple group analysis (CFA)



- can we compare the means of the latent variables?

3.3 Measurement invariance

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

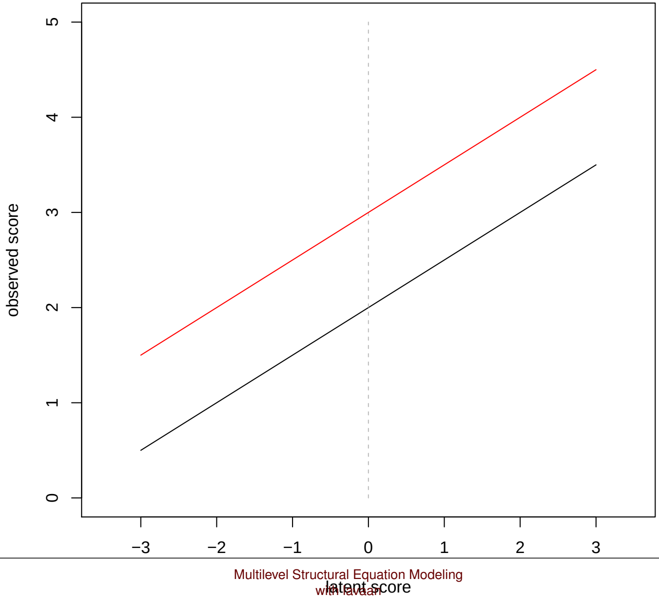
- testing for measurement invariance involves a fixed sequence of model comparison tests

- 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
```

(optional) measurement invariance tests – manual

```
> # configural model (manual)
> HS.model.configural <- '
+     visual  =~ c(1,1)*x1 + c(l2.1, l2.2)*x2 + c(l3.1, l3.2)*x3
+     textual =~ c(1,1)*x4 + c(l5.1, l5.2)*x5 + c(l6.1, l6.2)*x6
+     speed   =~ c(1,1)*x7 + c(l8.1, l8.2)*x8 + c(l9.1, l9.2)*x9
+
+     # ov intercepts
+     x1 ~ c(i1.1, i1.2)*1
+     x2 ~ c(i2.1, i2.2)*1
+     x3 ~ c(i3.1, i3.2)*1
+     x4 ~ c(i4.1, i4.2)*1
+     x5 ~ c(i5.1, i5.2)*1
+     x6 ~ c(i6.1, i6.2)*1
+     x7 ~ c(i7.1, i7.2)*1
+     x8 ~ c(i8.1, i8.2)*1
+     x9 ~ c(i9.1, i9.2)*1
+
+     # lv means (optional, zero by default)
+     visual  ~ c(0,0)*1
+     textual ~ c(0,0)*1
+     speed   ~ c(0,0)*1
+ '
> fit1b <- cfa(HS.model.configural, data = HolzingerSwineford1939,
+              group = "school")
> # weak invariance model (manual)
> # equal factor loadings
```

```
> HS.model.weak <- '
+     visual  =~ c(1,1)*x1 + c(l2, l2)*x2 + c(l3, l3)*x3
+     textual =~ c(1,1)*x4 + c(l5, l5)*x5 + c(l6, l6)*x6
+     speed   =~ c(1,1)*x7 + c(l8, l8)*x8 + c(l9, l9)*x9
+
+     # ov intercepts
+     x1 ~ c(i1.1, i1.2)*1
+     x2 ~ c(i2.1, i2.2)*1
+     x3 ~ c(i3.1, i3.2)*1
+     x4 ~ c(i4.1, i4.2)*1
+     x5 ~ c(i5.1, i5.2)*1
+     x6 ~ c(i6.1, i6.2)*1
+     x7 ~ c(i7.1, i7.2)*1
+     x8 ~ c(i8.1, i8.2)*1
+     x9 ~ c(i9.1, i9.2)*1
+
+     # lv means (optional, zero by default)
+     visual  ~ c(0,0)*1
+     textual ~ c(0,0)*1
+     speed   ~ c(0,0)*1
+ '
> fit2b <- cfa(HS.model.weak, data = HolzingerSwineford1939,
+              group = "school")
> # strong invariance model (manual)
> #  - equal factor loadings
> #  - equal intercepts
> #  - free latent means for the second group
> HS.model.strong <- '
```

```
+     visual  =~ c(1,1)*x1 + c(l2, l2)*x2 + c(l3, l3)*x3
+     textual =~ c(1,1)*x4 + c(l5, l5)*x5 + c(l6, l6)*x6
+     speed   =~ c(1,1)*x7 + c(l8, l8)*x8 + c(l9, l9)*x9
+
+     # ov intercepts
+     x1 ~ c(i1, i1)*1
+     x2 ~ c(i2, i2)*1
+     x3 ~ c(i3, i3)*1
+     x4 ~ c(i4, i4)*1
+     x5 ~ c(i5, i5)*1
+     x6 ~ c(i6, i6)*1
+     x7 ~ c(i7, i7)*1
+     x8 ~ c(i8, i8)*1
+     x9 ~ c(i9, i9)*1
+
+     # lv means
+     visual  ~ c(0, NA)*1
+     textual ~ c(0, NA)*1
+     speed   ~ c(0, NA)*1
+ '
> fit3b <- cfa(HS.model.strong, data = HolzingerSwineford1939,
+              group = "school")
```

4 Missing data and non-normal (continuous) data

4.1 Missing data

missing data mechanisms

- MCAR: missing completely at random
  - listwise deletion is ok (data is lost, but the estimates are still unbiased)
- MAR: missing at random
  - what caused the data to be missing does not depend upon the missing data itself, but may depend on the non-missing data
  - listwise deletion is NOT ok: estimates are biased
  - alternatives: full information ML (FIML), multiple imputation, ...
- NMAR: not missing at random
  - we can only try to understand the missingness mechanism at hand, and take this into account when modeling the data

missing data in SEM

- assumption: missing data mechanism is MAR + continuous data
- three approaches:
  1. multiple imputation (Rubin, 1987):
     - create several 'completed' datasets by imputing the missing data under an imputation model
     - fit the model for each dataset
     - pool the results to obtain point estimates, standard errors, test statistics
  2. 'full information' (case-wise) ML estimation:
     - for each observation, compute the (log)likelihood with the available information
  3. two-stage approach (eg., Yuan & Bentler, 2000)
     - estimate mean vector and sample covariance matrix
     - using these sample statistics, perform SEM

missing data in lavaan

- in lavaan 0.6, the default is listwise deletion (but this may change in future versions)

```
lavaan 0.6-3 ended normally after 35 iterations

                                                  Used       Total
  Number of observations                           156         301
```

  - the goal is to alert the user that data is missing

- 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 + fiml

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

lavaan 0.6-7 ended normally after 54 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free 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 0.6-7 ended normally after 37 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         30

  Number of observations                           301
  Number of missing patterns                        13

Model Test User Model:
                                              Standard      Robust
  Test Statistic                                91.404      88.217
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  1.036
       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)

4.2 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 ($\hat{\boldsymbol{\Sigma}}$) 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")
```

```
> summary(fit, fit.measures = TRUE, estimates = FALSE)

lavaan 0.6-7 ended normally after 35 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free 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

Model Test Baseline Model:

  Test statistic                               918.852     789.298
  Degrees of freedom                                36          36
  P-value                                        0.000       0.000
  Scaling correction factor                                  1.164

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    0.931       0.925
```

```
  Tucker-Lewis Index (TLI)                       0.896       0.887

  Robust Comparative Fit Index (CFI)                         0.932
  Robust Tucker-Lewis Index (TLI)                            0.897

Loglikelihood and Information Criteria:

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

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

Root Mean Square Error of Approximation:

  RMSEA                                          0.092       0.089
  90 Percent confidence interval - lower         0.071       0.068
  90 Percent confidence interval - upper         0.114       0.110
  P-value RMSEA <= 0.05                          0.001       0.001

  Robust RMSEA                                               0.091
  90 Percent confidence interval - lower                     0.070
  90 Percent confidence interval - upper                     0.113

Standardized Root Mean Square Residual:

  SRMR                                           0.065       0.065
```

robust method 3: bootstrapping

1. **parameter estimates: vanilla ML**

2. **bootstrapping standard errors**

   - for the standard errors, we can use the usual nonparametric bootstrap:
     (a) take a bootstrap sample (random selection of cases with replacement)
     (b) fit the model using this bootstrap sample
     (c) extract the t estimated values of the free parameters
     (d) repeat steps 1–3 R times (typically, $R > 1000$)
   - collect all these values in a matrix of size $R \times t$
   - the bootstrap standard errors are the square root of the diagonal elements of the covariance matrix of this $R \times t$ matrix

3. **bootstrapping the test statistic**

- for the test statistic, we can not use the usual nonparametric bootstrap, because it reflects not only non-normality and sampling variability, but also model misfit
- the original sample must first be transformed so that the sample covariance matrix corresponds with the model-implied covariance matrix
- in the SEM literature, this model-based bootstrap procedure is known as **the Bollen-Stine bootstrap**
- the standard p value of the chi-square test can be replaced by a bootstrap p value: the proportion of test statistics from the bootstrap samples that exceed the value of the test statistic from the original (parent) sample

bootstrapping in lavaan

- 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

using bootstrapLavaan() to compute the Bollen-Stine p-value (optional)

```
fit <- cfa(HS.model, data = HolzingerSwineford1939, se = "none")

# get the test statistic for the original sample

T.orig <- fitMeasures(fit, "chisq")

# bootstrap to get bootstrap test statistics
# we only generate 10 bootstrap sample in this example; in practice
# you may wish to use a much higher number

T.boot <- bootstrapLavaan(fit,
                          R = 10,
                          type = "bollen.stine",
                          FUN = fitMeasures,
                          fit.measures = "chisq")

# compute a bootstrap based p-value

pvalue.boot <- length(which(T.boot > T.orig))/length(T.boot)
```

5 Categorical data

5.1 Handling categorical endogenous variables

categorical exogenous variables

- categorical exogenous covariates; eg. gender, country
- we simply need to construct 'dummy variables' and proceed as usual
- just like in ordinary regression

categorical endogenous variables

- need special treatment
- binary data, ordinal (ordered) data
- censored data, limited dependent data
- count data, nominal (unordered) data, ...

5.2 Two approaches for handling categorical data in a SEM framework

- 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

5.3 A limited information approach: the WLSMV estimator

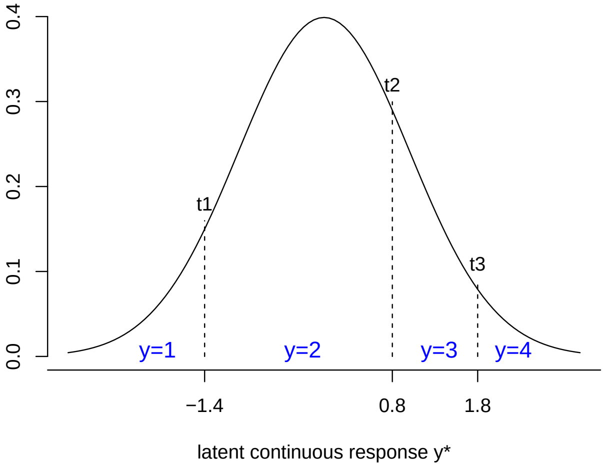
- developed by Bengt Muthén, in a series of papers; the seminal paper is

  Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika, 49*, 115–132

- this approach has been the 'golden standard' in the SEM literature
- first available in LISCOMP (Linear Structural Equations using a Comprehensive Measurement Model), distributed by SSI, 1987 – 1997
- follow up program: Mplus (Version 1: 1998), currently version 8
- other authors (Jöreskog 1994; Lee, Poon, Bentler 1992) have proposed similar approaches (implemented in LISREL and EQS respectively)
- another great program: MECOSA (Arminger, G., Wittenberg, J., Schepers, A.) written in the GAUSS language (mid 90's)

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

- š 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 (š)
- 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 '

> # binary CFA
> fit <- cfa(model, data=HSbinary, ordered = names(HSbinary))
```

output

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

lavaan 0.6-7 ended normally after 35 iterations

  Estimator                                       DWLS
  Optimization method                           NLMINB
  Number of free 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
```

6 Longitudinal Structural Equation Modeling

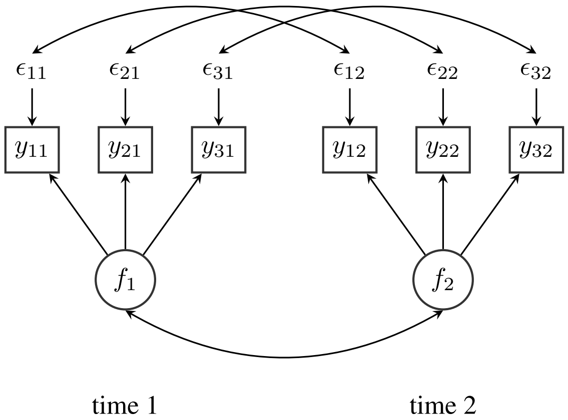
- long history, mostly for 'balanced data': same number of time points for each observation
  - repeated measures models
  - panel models, simplex models, autoregressive models
  - growth curve models (random coefficient models)
  - hybrid models (growth curve + autoregressive)
  - latent-state, latent-trait models
  - latent difference scores models
  - ...
- multilevel SEM
  - combines 'mixed models' with path analysis and latent variables
  - allows for unbalanced data
  - relatively new, active research; major software package: Mplus

6.1 Repeated measures ANOVA, using SEM

- we can mimic the classical repeated measures ANOVA in a SEM framework
- using two time-points only, this is the SEM equivalent of the paired t-test
- but we can relax the compound symmetry restriction
  - we can allow for an unstructured covariance structure
  - or we could impose an autoregressive AR(1) structure
  - ...
- but above all, we can replace the observed variables by latent variables

repeated measures using latent variables

- example with 2 time points:

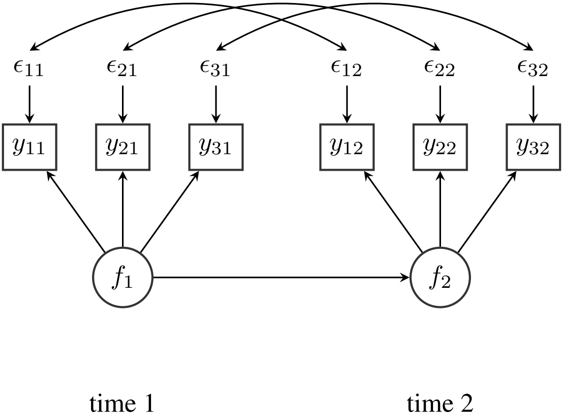


6.2 Panel models for longitudinal data

- panel models postulate *directional* (regression) relationships among the repeated measures
- the 'covariance' is replaced by a 'regression'
- 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
  - ...

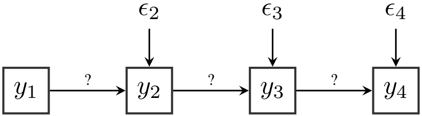
example panel model with a single latent variable

- example with 2 time points:



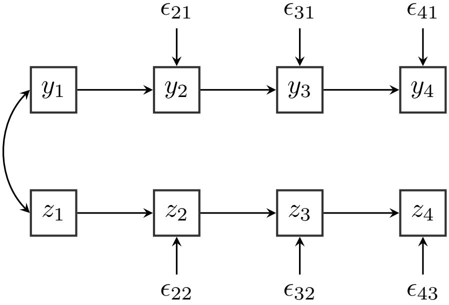
autoregressive models

- 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:



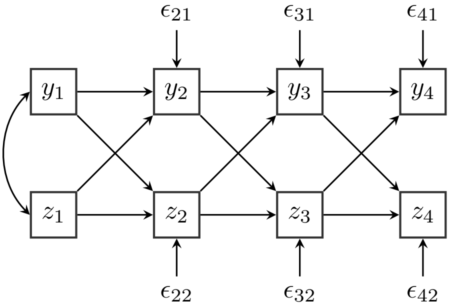
multivariate panel models

- 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)



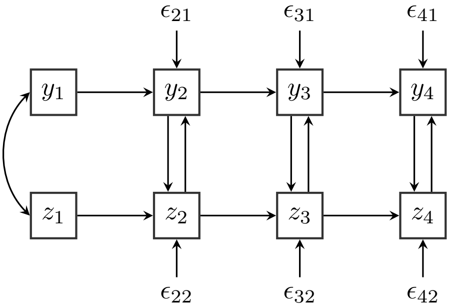
crosslagged effects

- 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?



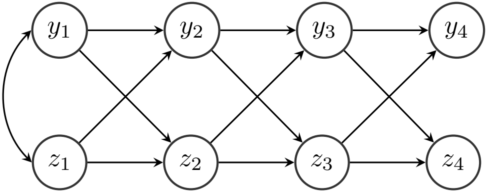
contemporaneous effects

- 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

strengths and limitations of panel models

- panel models can be very useful for examining the relations of two (or more) variables (observed or latent) over time
- often, we are equally interested in the lack of relations over time
- panel models do not tell us anything about group level tendencies (overall increase or decrease of the scores)
- panel models do not tell us anything about individual tendencies

6.3 Growth curve models

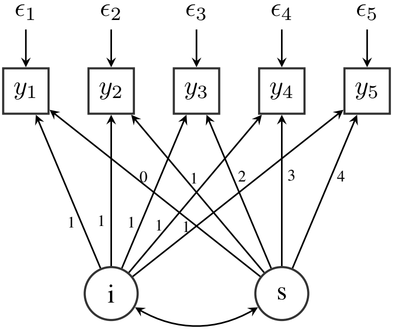
- 'time' is typically considered as a continuous variable
- two components:
  - fixed effects: what is the nature of the average trend (linear, quadratic)
  - random effects: individual differences
- in addition, we may try to explain these individual differences by taking into account:
  - time-invariant 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

7 Multilevel regression

7.1 Brief overview

different types of data with non-independent observations

- clustered data (family members, teeth in a mouth)
- dyadic data (romantic couples)
- hierarchical data (students within schools within regions)
- matched data (case-control studies)
- survey data (nested sampling)
- longitudinal data (blood pressure of patients measured every week)
- repeated measures (within-subjects design)
- ...

balanced versus unbalanced data

- when the data is balanced, we have the same number of units within each cluster
- typical examples of balanced data:
  - dyadic data: always two units per cluster
  - repeated measures data: everyone has scores for the same set of conditions
  - longitudinal data where the number of observations (over time) is the same for all individuals (often called panel data)
  - hierarchical data where a fixed number of units was sampled for each cluster
- when the data is unbalanced, we have different cluster sizes
  - this may be due to missing values
  - in hierarchical data, the number of units for each cluster may vary considerably from cluster to cluster

wide versus long data

- when data is arranged in ‘wide’ format, each row corresponds to a single cluster
  - we may end up with many columns (one for each measure/variable, for each unit)
  - rows are independent
  - unbalanced data can be handled by filling in missing values for the smaller clusters
- when data is arranged in ‘long’ format, each row corresponds to a single unit
  - the columns contain the variables for that unit (only)
  - multiple rows belong to the same clusters
  - rows are not independent
  - higher-level variables (for example school characteristics) are duplicated for each unit

example wide format

```
  cluster.id y1 m1 x1 y2 m2 x2 y3 m3 x3 schoolsize
1          1 16  4 60 28 36  6  4 22 12      large
2          2 24 14 10 18  6 20 38 28 22     medium
3          3 26  2  2 32  4  8  4  4 10     medium
4          4  4 36 14  2  2  0  8  8 10      small
5          5 14 10 16 28  2  4  8 22  6      small
6          6 24 20 16 42 18  2  2 28 18      large
7          7 22  0 14 32  6  2 18 18 10     medium
8          8  0  8 34 16 16 14  8 28 18      large
```

example long format

```
   cluster.id  y  m  x schoolsize
1           1 16  4 60      large
2           1 28 36  6      large
3           1  4 22 12      large
4           2 24 14 10     medium
5           2 18  6 20     medium
6           2 38 28 22     medium
7           3 26  2  2     medium
8           3 32  4  8     medium
9           3  4  4 10     medium
10          4  4 36 14      small
11          4  2  2  0      small
12          4  8  8 10      small
13          5 14 10 16      small
14          5 28  2  4      small
15          5  8 22  6      small
16          6 24 20 16      large
17          6 42 18  2      large
18          6  2 28 18      large
19          7 22  0 14     medium
20          7 32  6  2     medium
21          7 18 18 10     medium
22          8  0  8 34      large
23          8 16 16 14      large
24          8  8 28 18      large
```

two traditions

1. the multivariate or ('wide-format') approach:
   - data is in wide format
   - number of observations within a cluster is (very) small (say, 2 to 10)
   - mostly for balanced data
   - dominant approach for longitudinal/repeated-measures data with a small number of timepoints
   - examples: repeated-measures MANOVA, panel models, growth curve models, ...
2. the multilevel (or random-effects, or mixed modeling) approach:
   - data is in long format
   - mostly for unbalanced data
   - dominant approach for hierchical data with large clusters
   - examples: repeated-measures ANOVA, multilevel regression, (generalized) linear mixed models, multilevel SEM, ...

example dataset: Demo.twolevel (N=2500, J=200)

```
> library(lavaan)
> head(Demo.twolevel)

          y1         y2         y3         y4         y5         y6         x1
1  0.2293216  1.3555232 -0.6911702  0.8028079 -0.3011085 -1.7260671  1.1739003
2  0.3085801 -1.8624397 -2.4179783  0.7659289  1.6750617  1.1764210 -1.0039958
3  0.2004934 -1.3400514  0.4376087  1.1974194  1.1951594  1.4988962 -0.4402545
4  1.0447982 -0.9624490 -0.4464898 -0.2027252 -0.4590574  1.1734061 -0.6253657
5  0.6881792 -0.4565633 -0.6422296  0.9900408  1.7662535  0.7944601 -0.8450025
6 -2.0687644 -0.5997856  0.3148418  0.6764432 -0.6519928  1.8405605 -0.7831784
           x2         x3         w1         w2 cluster
1 -0.62315173  0.6470414 -0.2479975 -0.4989800       1
2 -0.56689380  0.0201264 -0.2479975 -0.4989800       1
3 -2.13432572 -0.4591246 -0.2479975 -0.4989800       1
4 -0.33688869  1.2852093 -0.2479975 -0.4989800       1
5 -0.04229954  1.5598970 -0.2479975 -0.4989800       1
6 -0.22441996 -0.3814231 -2.3219338 -0.6910567       2
```

- level-1 ('within') variables: y1-y6, x1-x3
- level-2 ('between') variables: w1-w2
- cluster identifier (`cluster`)

a typical setting: students nested within schools

- suppose this dataset contains scores from N=2500 students (or pupils), aged 10-12, sampled from J=200 primary schools
- assume (for now) we only have a single outcome variable (`y1`), say an (observed) measure of reading motivation (range: -7.95 to 5.99)
- we wish to analyse a model that looks like this:

```
> y1 ~ x1 + x2 + x3 + w1 + w2
```

- the model contains both level-1 and level-2 predictors
  - the x-variables are student characteristics (eg., ses, test scores, ...)
  - the w-variables are school characteristics (eg., average ses, schoolsize)
- the observations are clustered
- unbalanced (cluster sizes are: 5, 10, 15 and 20)
- how shall we proceed?

ignoring the dependency structure

- we could treat the sample as a simple random sample with N independent observations, and use an ordinary regression model
- in the multilevel context, this is often called a 'disaggregated analysis', as higher-level variables (e.g., school characteristics) are assigned to the individual level
- although (still) sometimes used, ignoring the clustering in the data may have severe consequences:
  - biased point estimates
  - wrong standard errors, wrong test statistics, wrong p-values
- what about reviewers?
  - the 'tolerance' for ignoring the clustering in data is now almost non-existing in most fields

ignoring the dependency structure: using lm()

```
> fit.lm <- lm(y1 ~ x1 + x2 + x3 + w1 + w2, data = Demo.twolevel)
> summary(fit.lm)

Call:
lm(formula = y1 ~ x1 + x2 + x3 + w1 + w2, data = Demo.twolevel)

Residuals:
   Min     1Q Median     3Q    Max
-6.844 -1.061 -0.029  1.098  5.438

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.03416    0.03192   1.070    0.285
x1           0.49412    0.03207  15.409  < 2e-16 ***
x2           0.37635    0.03161  11.905  < 2e-16 ***
x3           0.17035    0.03112   5.474 4.85e-08 ***
w1           0.21396    0.03372   6.345 2.64e-10 ***
w2           0.13502    0.03341   4.041 5.48e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.589 on 2494 degrees of freedom
Multiple R-squared:  0.1574,	Adjusted R-squared:  0.1557
F-statistic: 93.18 on 5 and 2494 DF,  p-value: < 2.2e-16
```

taking the dependency structure into account: using lmer()

```
> library(lme4)
> fit.lmer <- lmer(y1 ~ x1 + x2 + x3 + w1 + w2 + (1 | cluster),
+                  data = Demo.twolevel)
> summary(fit.lmer, corr = FALSE)

Linear mixed model fit by REML ['lmerMod']
Formula: y1 ~ x1 + x2 + x3 + w1 + w2 + (1 | cluster)
   Data: Demo.twolevel

REML criterion at convergence: 8615.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.4055 -0.6383  0.0102  0.6366  3.4714

Random effects:
 Groups   Name        Variance Std.Dev.
 cluster  (Intercept) 0.9724   0.9861
 Residual             1.5416   1.2416
Number of obs: 2500, groups:  cluster, 200

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.02499    0.07544   0.331
x1           0.49895    0.02590  19.264
x2           0.40787    0.02556  15.960
```

```
x3            0.21080    0.02533   8.324
w1            0.16438    0.08102   2.029
w2            0.11661    0.07866   1.483
```

```
> # library(car)
> # Anova(fit.lmer, type = "II", test = "F")
> # to get Type II Wald F tests with Kenward-Roger df

# Response: y1
# F Df  Df.res  Pr(>F)
# x1 371.0325  1 2342.21 < 2e-16 ***
# x2 254.6577  1 2344.59 < 2e-16 ***
# x3  69.2648  1 2350.63 < 2e-16 ***
# w1   4.1156  1  193.45 0.04386 *
# w2   2.1977  1  196.41 0.13982
```

how to handle clustered data: many solutions

- avoiding the clustering (only pick one observation per cluster)
- aggregating the data (bad idea: may lead to ecological fallacies)
- reshape the data to wide-format, use a multivariate approach
  - ideal for balanced data with a small number of observations per cluster
- cluster-robust standard errors (clustering is just a nuisance)
  - often called 'design-based' or 'survey' approach
- fixed-effects approach (cluster/school as a fixed factor)
  - conclusions are only valid for the clusters/schools present in the sample
  - often used if the number of clusters is rather small (< 10)
- **mixed-effects approach** (cluster/school as a random factor): the focus of this course
- ...

the many faces of mixed-effects models

- mixed-effects models have been developed in a variety of disciplines, with varying names and terminology:
  - random-effects (ANOVA) models (statistics, econometrics)
  - linear mixed models (statistics)
  - variance components models (statistics)
  - hierarchical linear models (education, Bayesian)
  - multilevel models (sociology, education)
  - contextual-effects models (sociology)
  - random-coefficient models (econometrics)
  - repeated-measures models, repeated measures ANOVA (statistics, psychology)
  - ...
- the different terminology is still a source of much confusion

multilevel regression: brief history

- multilevel regression is the application of mixed-effects statistical models to analyze hierarchical (or multilevel) data
- this branch of statistics was mainly developed in the educational sciences, and in quantitative sociology
- Blalock (1984) introduced ‘contextual effect models’ in sociology
- school effectiveness researchers realized early on (’70s, ’80s) that taking the cluster structure into account was important
  - a regression analysis per school was one solution, but this ignored the fact that many regression coefficients (across schools) should be similar; this similarity should be used (‘borrowing strength’)
  - on the other hand, requiring regression coefficients in all schools to be the same, was regarded as too restrictive
  - clearly, some intermediate form of analysis was needed

- this led to the idea of random coefficient models, but it left open the problem of combining predictors of different levels
- Burstein (and others) suggested in the early ’80s to proceed in two stages:
  - in a first stage, a regression analysis was done for each school
  - in a second stage, the resulting regression coefficients were entered as outcome variables in a regression, where the predictors were cluster variables
  - this became known as the ‘slopes-as-outcomes’ approach
- in the mid ’80s, it became clear that the models that educational researchers were looking for had been around for quite some time in other branches of statistics (e.g., linear mixed models)
- a number of authors published a series of papers that would eventually lead to what we now call today ‘multilevel regression’ (Mason et al., 1983; Aitkin and Longford, 1986; de Leeuw and Kreft, 1986; Goldstein, 1986; Raudenbush and Bryk, 1986)

- some important textbooks paved the way for a wide adoption of multilevel regression in the social and behavioural sciences:
  - Goldstein, H. (1987). *Multilevel Statistical Models.* London: Edward Arnold. (Cfr. MLwiN software)
  - Raudenbush, S.W. & Bryk A.S. (1992) *Hierarchical Linear Models: Applications and Data Analysis Methods.* Thousand Oaks, Calif.: Sage. (Cfr. HLM software)
  - Hox, J. (1995). *Applied Multilevel Analysis.* Amsterdam: TT-Publikaties.
  - Kreft, I.G.G. & De Leeuw, J. (1998) *Introducing Multilevel Modeling.* Sage, London.
  - Snijders, T. & Bosker, R. (1999). *Multilevel Analysis: An Introduction to Basic and Advanced Multilevel Modeling.* Thousand Oaks, Calif.: Sage.
- additional information:
  http://www.bristol.ac.uk/cmm/learning/

7.2 The linear mixed model (optional)

the structure of the model

- the *Laird-Ware form* of the linear mixed model (two-level):

$$\begin{aligned} y_{ji} = \; & \beta_1 x_{1ji} + \beta_2 x_{2ji} + \beta_3 x_{3ji} + \ldots + \beta_p x_{pji} + \\ & b_{1j} z_{1ji} + b_{2j} z_{2ji} + \ldots + b_{qj} z_{qji} + \\ & \epsilon_{ji} \end{aligned}$$

- y_{ji} is the value of the response variable for the ith of n_j observations of cluster $j = 1, 2, \ldots, J$
- $x_{1ji}, \ldots, x_{pji}$ are the values of the p regressors for observation i in cluster j; they are *fixed* constants (with respect to repeated sampling), and can be anything (product terms, indicator variables, $\ldots$)
- in many regression models, a constant term $x_{0ji} = 1$ is added; β_0 is called the (fixed) intercept

- the regression coefficients $\beta_1, \ldots, \beta_p$ are the fixed-effect coefficients, which are identical for all clusters
- $b_{1j}, b_{2j}, \ldots, b_{qj}$ are the random-effect coefficients for cluster j; the random-effect coefficients are thought of as random variables, not as parameters (similar to the errors ϵ_{ji})
- $z_{1ji}, z_{2ji}, \ldots, z_{qji}$ are the random-effect regressors; they are typically a subset of the fixed regressors; in many models, a random intercept term $z_{0ji} = 1$ is added and b_{0j} is called a random intercept
- ϵ_{ji} is the random error for the ith observation of cluster j
- the model Laird-ware model in matrix form:

$$\mathbf{y}_j = \mathbf{X}_j\boldsymbol{\beta} + \mathbf{Z}_j\mathbf{b}_j + \boldsymbol{\epsilon}_j \qquad j = 1, 2, \ldots, J$$

- the model for the all clusters:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}$$

stochastic assumptions in matrix notation

- the random-effect coefficients:
  - $\mathbf{b}_j \sim \mathrm{N}_q(\mathbf{0}, \mathbf{D})$
  - $\mathbf{b}_j$ and $\mathbf{b}_{j'}$ are independent for $j \neq j'$
- the error terms:
  - $\boldsymbol{\epsilon}_j \sim \mathrm{N}_{n_j}(\mathbf{0}, \boldsymbol{\Sigma}_j)$
  - $\boldsymbol{\epsilon}_j$ and $\boldsymbol{\epsilon}_{j'}$ are independent for $j \neq j'$
  - $\mathsf{Cov}[\boldsymbol{\epsilon}_j, \boldsymbol{b}_j] = \mathbf{0}$
- $\mathbf{D}$ contains $q(q+1)/2$ parameters; $\boldsymbol{\Sigma}_j$ contains $n_j(n_j+1)/2$ (non-redundant) elements
- the elements of $\mathbf{D}$ and the elements of $\boldsymbol{\Sigma}_j$ (for each j) are collectively called the *variance components*
- the elements of $\mathbf{D}$ are often parameterized in terms of a smaller number of fundamental parameters

conditional and marginal distributions

- the conditional distribution: $\mathbf{y}_j | \mathbf{b}_j \sim \mathrm{N}(\mathbf{X}_j \boldsymbol{\beta} + \mathbf{Z}_j \mathbf{b}_j,\, \boldsymbol{\Sigma}_j)$
- the marginal distribution:

$$\mathbf{y}_j \sim \mathrm{N}(\mathbf{X}_j \boldsymbol{\beta}, \mathbf{V}_j)$$

  where

$$\mathsf{Cov}[\mathbf{y}_j] = \mathbf{V}_j = \mathbf{Z}_j \mathbf{D} \mathbf{Z}_j' + \boldsymbol{\Sigma}_j$$

- note the similarity with the CFA model:

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

  – in a LMM, $\mathbf{Z}_j$ are known constants, while in CFA, $\boldsymbol{\Lambda}$ are often (unknown) parameters (factor loadings)
  – that is why a growth curve model can be defined as a CFA with 'fixed' factor loadings
  – remember: random effects are just latent variables with fixed factor-loadings

7.3 The two-level regression model with random intercepts

- in this framework, we decompose the total score of the outcome variable into two parts: a within part, and a between part:

$$y_{ji} = (y_{ji} - \bar{y}_j) + \bar{y}_j$$
$$y_T = y_W + y_B$$

where $j = 1, \ldots, J$ is an index for the clusters, and $i = 1, \ldots, n_j$ is an index for the units within a cluster

- $\bar{y}_j$ is the cluster mean of cluster j; however, in a real dataset, the observed cluster means are not necessary equal to the 'true' cluster means (due to sampling error, and perhaps measurement error)
  - therefore, we will treat the cluster means as (unobserved) scores from a latent variable
  - if $\bar{y}_j$ in $(y_{ji} - \bar{y}_j)$ is unobserved, then the result (y_W) is also unobserved
  - conclusion: both components y_W and y_B are treated as unknown (latent) variables

lavaan syntax setup for two-level regression

within part

$$\frac{\text{Between}}{\text{Within}}$$

between part

```
model <- '

  level: 1

    # here comes the within level

  level: 2

    # here comes the between level
'

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

model 1: the empty (univariate) model

- it is called the ‘empty’ model, since it contains no predictors, but simply reflects the nested structure
- no level-1 predictors, no level-2 predictors
- the model in Laird-Ware form:

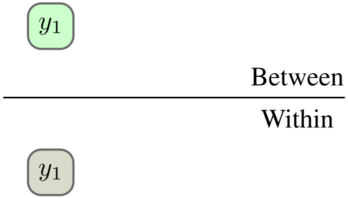
$$y_{ji} = \beta_0 + b_{0j} + \epsilon_{ji}$$

- this is an example of a random-effects one-way ANOVA model with one fixed effect (the intercept, β_0) representing the general population mean of y_1, and two random effects:
  - b_{0j} representing the deviation of the cluster mean of y in cluster j from the general mean
  - ϵ_{ji} representing the deviation of observation i's scores for y in cluster j from the cluster mean

- there are two variance components for this model:
  - $\mathsf{Var}(b_{0j}) = d^2$: the variance among cluster means
  - $\mathsf{Var}(\epsilon_{ji}) = \sigma^2$: the variance among observations in the same cluster
- since b_{0j} and ϵ_{ji} are assumed to be independent, the variation in y_1 among observations can be decomposed into these two variance components:

$$\mathsf{Var}(y_{ji}) = d^2 + \sigma^2$$

model 1: diagram and lavaan syntax




```
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

```
lavaan 0.6-7 ended normally after 17 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          3

  Number of observations                          2500
  Number of clusters [cluster]                     200

Model Test User Model:

  Test statistic                                0.0000
  Degrees of freedom                                 0

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian


Level 1 [within]:

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 [cluster]:

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
```

lmer version

```
> library(lme4)
> fit.lmer <- lmer(y1 ~ 1 + (1 | cluster), data = Demo.twolevel, REML = FALSE)
> summary(fit.lmer)

Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: y1 ~ 1 + (1 | cluster)
   Data: Demo.twolevel

     AIC      BIC   logLik deviance df.resid
  9203.4   9220.9  -4598.7   9197.4     2497

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.7565 -0.6399  0.0276  0.6473  2.9744

Random effects:
 Groups   Name        Variance Std.Dev.
 cluster  (Intercept) 0.9436   0.9714
 Residual             2.0003   1.4143
Number of obs: 2500, groups:  cluster, 200

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.01977    0.07553   0.262
```

intra-class correlation (icc)

- the *intra-class correlation* is the proportion of variance 'at the between level' (or due to differences among the clusters):

$$\frac{\mathsf{Var}(y_B)}{\mathsf{Var}(y_W) + \mathsf{Var}(y_B)} = \frac{d^2}{d^2 + \sigma^2} = \rho$$

- ρ may also be interpreted as the correlation between the scores of two observations from the same cluster:

$$\mathsf{Cor}(y_{ji}, y_{ji'}) = \frac{\mathsf{Cov}(y_{ji}, y_{ji'})}{\sqrt{\mathsf{Var}(y_{ji}) \times \mathsf{Var}(y_{ji'})}} = \frac{d^2}{\sqrt{(d^2 + \sigma^2)(d^2 + \sigma^2)}} = \rho$$

- typical range: between 0.05 and 0.5
- some textbooks suggest that if the icc is low (say, < 0.05), you don't need to use multilevel modeling; it turns out this is bad advice: always use a method that takes the clustering into account

computing the icc

- by hand:

```
> 0.9436/(0.9436 + 2.0003)

[1] 0.3205272
```

- using lmer:

```
> var.comp <- as.data.frame(VarCorr(fit.lmer))$vcov
> rho <- var.comp[1]/(var.comp[1] + var.comp[2]); rho

[1] 0.3205193
```

- using lavaan:

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

   y1
0.321
```

- about 32 percent of the variation in students' scores (on y_1) is "attributable" to differences among clusters

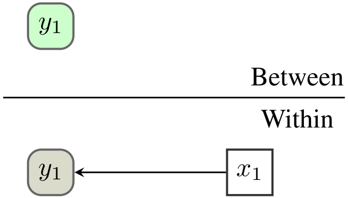
model 2a: simple twolevel regression (predictor at within level)

- 1 level-1 predictor (x_1, not centered), no level-2 predictors
- the model in the Laird-Ware form:

$$y_{ji} = \beta_0 + \beta_1 x_{1ji} + b_{0j} + \epsilon_{ji}$$

- the model has two variance-covariance components:
  - $\mathsf{Var}(b_{0j}) = d^2$: the variance among cluster *intercepts*
  - $\mathsf{Var}(\epsilon_{ji}) = \sigma^2$: the error variance around the within-cluster regressions
- interpretation of the (fixed) regression coefficients:
  - β_0: predicted value for y if $x_1 = 0$
  - β_1: predicted change in y for a one-unit increase of x_1
- note that x_1 varies within clusters, but also between clusters; as a result, β_1 is a combination of both the within-cluster and the between-cluster effect of x_1 on y_1; because of this, x_1 is often group-mean centered

model 2a: diagram and lavaan syntax




```
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
```

lmer version

```
> fit.lmer <- lmer(y1 ~ 1 + x1 + (1 | cluster), data = Demo.twolevel,
                   REML = FALSE)
> summary(fit.lmer, corr = FALSE)

Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: y1 ~ 1 + x1 + (1 | cluster)
   Data: Demo.twolevel

     AIC      BIC   logLik deviance df.resid
  8905.5   8928.8  -4448.7   8897.5     2496

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.4544 -0.6505  0.0157  0.6221  3.3789

Random effects:
 Groups   Name        Variance Std.Dev.
 cluster  (Intercept) 0.9366   0.9678
 Residual             1.7599   1.3266
Number of obs: 2500, groups:  cluster, 200

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.02225    0.07454   0.298
x1           0.49435    0.02765  17.880
```

centering of a level-1 predictor (x_1)

- centering a variable is shifting its origin: we place the '0' at a different value
- grand-mean centering: using the 'grand' mean to center x_1
  - will only change the intercept (not the slope)
  - β_0 is now the predicted value when x_1 equals the grand mean
  - β_1 is still a combination of both the within and between effect
- manifest group-mean or cluster-mean centering: subtract the (observed) cluster means
  - changes both β_0 and β_1
  - has the advantage that β_1 now only reflects the 'within' effect of x_1
- latent group-mean or cluster-mean centering: (implicitly) subtract the latent cluster means
- this is for cross-sectional studies; for longitudinal studies, other approaches may be more appropriate if 'time' is a predictor

manifest group-mean centering in R (the ugly way)

```
> x1.mean <- tapply(Demo.twolevel$x1, Demo.twolevel$cluster, mean)

> cluster.idx <- Demo.twolevel$cluster
> Demo.twolevel$x1.c <- Demo.twolevel$x1 - x1.mean[cluster.idx]
> Demo.twolevel$x1.mean <- x1.mean[cluster.idx]

> head(Demo.twolevel[, c("cluster", "x1", "x1.mean", "x1.c")], 12)

   cluster         x1     x1.mean        x1.c
1        1  1.1739003 -0.34814363  1.52204398
2        1 -1.0039958 -0.34814363 -0.65585216
3        1 -0.4402545 -0.34814363 -0.09211089
4        1 -0.6253657 -0.34814363 -0.27722209
5        1 -0.8450025 -0.34814363 -0.49685883
6        2 -0.7831784 -0.01326785 -0.76991060
7        2 -0.1776050 -0.01326785 -0.16433720
8        2  0.9498142 -0.01326785  0.96308203
9        2 -1.1891041 -0.01326785 -1.17583630
10       2  1.3785743 -0.01326785  1.39184214
11       2  1.0355019 -0.01326785  1.04876978
12       2 -0.5227660 -0.01326785 -0.50949815
```

comparing uncentered versus (manifest) group-mean centered

```
> fit.lmer <- lmer(y1 ~ 1 + x1 + (1 | cluster), data = Demo.twolevel,
                   REML = FALSE)
> round(fixef(fit.lmer), 3)

(Intercept)          x1
      0.022       0.494
```

```
> fit.lmer <- lmer(y1 ~ 1 + x1.c + (1 | cluster), data = Demo.twolevel,
                   REML = FALSE)
> round(fixef(fit.lmer), 3)

(Intercept)        x1.c
      0.020       0.493
```

- here, the differences are small, but in other datasets, the differences may be substantial

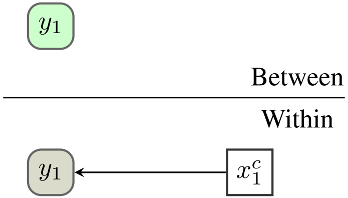
model 2b: manifest group-mean centering

- 1 level-1 predictor (x_1, group-mean centered), no level-2 predictors
- the model in the Laird-Ware form:

$$y_{ji} = \beta_0 + \beta_1(x_{1ji} - \bar{x}_{1j}) + b_{0j} + \epsilon_{ji}$$

- interpretation of the (fixed) regression coefficients:
  - β_0: the (estimated) average of the (unobserved) cluster means
  - β_1: predicted change in y for a one-unit increase of x_1 *at the within level only*

model 2b: diagram and lavaan syntax




```
model <- '
  level: 1
    y1 ~ x1.c
  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.c             0.4934    0.0278   17.7381    0.0000

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

Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
   .y1               1.7601    0.0518   33.9494    0.0000


Level 2 []:

Intercepts:
                   Estimate   Std.Err   z-value   P(>|z|)
   .y1               0.0197    0.0754    0.2614    0.7938

Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
   .y1               0.9632    0.1124    8.5681    0.0000
```

lmer version

```
> fit.lmer <- lmer(y1 ~ 1 + x1.c + (1 | cluster), data = Demo.twolevel,
                   REML = FALSE)
> summary(fit.lmer, corr = FALSE)

Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: y1 ~ 1 + x1.c + (1 | cluster)
   Data: Demo.twolevel

     AIC      BIC   logLik deviance df.resid
  8910.5   8933.8  -4451.2   8902.5     2496

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.4616 -0.6552  0.0121  0.6235  3.3758

Random effects:
 Groups   Name        Variance Std.Dev.
 cluster  (Intercept) 0.9632   0.9814
 Residual             1.7601   1.3267
Number of obs: 2500, groups:  cluster, 200

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.01972    0.07543   0.261
x1.c         0.49342    0.02782  17.738
```

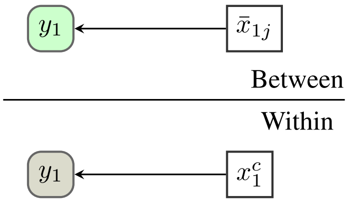
model 2c: manifest group-mean centering + between-effect

- 1 level-1 predictor (x_1, group-mean centered), 1 level-2 predictor (the group mean of x_1)
- the model in the Laird-Ware form:

$$y_{ji} = \beta_0 + \beta_1(x_{1ji} - \bar{x}_{1j}) + \beta_2\bar{x}_{1j} + b_{0j} + \epsilon_{ji}$$

- interpretation of the (fixed) regression coefficients:
  - β_0: the (estimated) average of the (unobserved) cluster means
  - β_1: predicted change in y for a one-unit increase of x_1 at the within level only
  - β_2: predicted change in y for a one-unit increase of x_1 at the between level only
- adding the between-part of x_1 will not change the regression coefficient (β_1) of the within-part

model 2c: diagram and lavaan syntax




```
model <- '
  level: 1
    y1 ~ x1.c
  level: 2
    y1 ~ x1.mean
'

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.c             0.4934   0.0278  17.7390   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 []:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  y1 ~
    x1.mean          0.5704   0.2518   2.2653   0.0235

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.0226   0.0745   0.3037   0.7614
```

```
Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
   .y1               0.9361    0.1096    8.5429    0.0000
```

lmer version

```
> fit.lmer <- lmer(y1 ~ 1 + x1.c + x1.mean + (1 | cluster), data = Demo.twolevel,
                   REML = FALSE)
> summary(fit.lmer, corr = FALSE)
```

```
Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: y1 ~ 1 + x1.c + x1.mean + (1 | cluster)
   Data: Demo.twolevel

     AIC      BIC   logLik deviance df.resid
  8907.4   8936.5  -4448.7   8897.4     2495

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.4541 -0.6491  0.0123  0.6221  3.3786

Random effects:
 Groups   Name        Variance Std.Dev.
 cluster  (Intercept) 0.9361   0.9675
 Residual             1.7599   1.3266
Number of obs: 2500, groups:  cluster, 200

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.02263    0.07453   0.304
x1.c         0.49342    0.02782  17.739
x1.mean      0.57037    0.25178   2.265
```

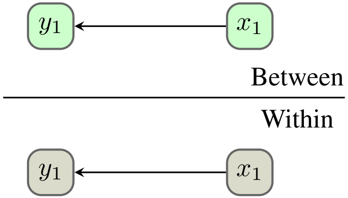
model 2d: latent group-mean centering + between-effect

- 1 level-1 predictor (the within-part of x_1), 1 level-2 predictor (the between-part of x_1)
- the model in the Laird-Ware form:

$$y_{ji} = \beta_0 + \beta_1 x_{1W} + \beta_2 x_{1B} + b_{0j} + \epsilon_{ji}$$

- interpretation of the (fixed) regression coefficients:
  - β_0: the (estimated) average of the (unobserved) cluster means
  - β_1: predicted change in y for a one-unit increase of x_1 at the within level only
  - β_2: predicted change in y for a one-unit increase of x_1 at the between level only
- in many settings, the β_2 coefficient is less biased (compared to the manifest centering approach), but also slightly less precise (more variance)
- only with SEM software

model 2d: diagram and lavaan syntax




```
model <- '

  level: 1

    y1 ~ x1

  level: 2

    y1 ~ x1

'
```

```
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               0.4939   0.0278  17.7885   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  33.9533   0.0000


Level 2 []:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  y1 ~
    x1              -0.0024   2.9408  -0.0008   0.9994

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.0186   0.0776   0.2399   0.8104
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.9381   0.1113   8.4269   0.0000
```

literature about centering

- manifest group-mean centering:

  Enders, C.K., & Tofighi, D. (2007). Centering predictor variables in cross-sectional multilevel models: a new look at an old issue. *Psychological methods, 12*, 121–138.

- latent group-mean centering:

  Lüdtke, O., Marsh, H.W., Robitzsch, A., Trautwein, U., Asparouhov, T., & Muthén, B. (2008). The multilevel latent covariate model: a new, more reliable approach to group-level effects in contextual studies. *Psychological methods, 13*, 203–229.

model 3: adding a level-2 predictor

- 1 level-1 predictor (x_1, group-mean centered), 2 level-2 predictors (the group mean of x_1, and w_1)

- the model in the Laird-Ware form:

$$y_{ji} = \beta_0 + \beta_1(x_{1ji} - \bar{x}_{1j}) + \beta_2\bar{x}_{1j} + \beta_3 w_{1j} + b_{0j} + \epsilon_{ji}$$

- interpretation of the (fixed) regression coefficients:

  – β_0: the (estimated) average of the (unobserved) cluster means

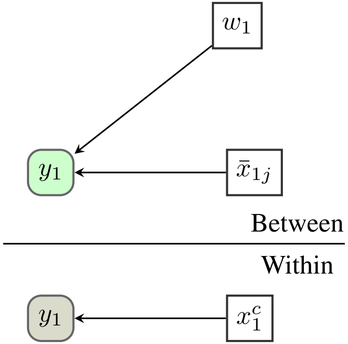
  – β_1: predicted change in y for a one-unit increase of x_1 at the within level only

  – β_2: predicted change in y for a one-unit increase of x_1 at the between level only

  – β_3: predicted change in y for a one-unit increase of w_1

- optionally w_1 can be grandmean-centered (only affecting the intercept)

model 3: diagram and lavaan syntax




```
model <- '
  level: 1
    y1 ~ x1.c
  level: 2
    y1 ~ x1.mean + w1
'

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.c             0.4934    0.0278   17.7382    0.0000

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

Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
   .y1               1.7601    0.0518   33.9502    0.0000


Level 2 []:

Regressions:
                   Estimate   Std.Err   z-value   P(>|z|)
  y1 ~
    x1.mean          0.5343    0.2498    2.1387    0.0325
    w1               0.1598    0.0789    2.0250    0.0429

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

```
   .y1                 0.0151    0.0738    0.2042    0.8382

Variances:
                     Estimate   Std.Err   z-value   P(>|z|)
   .y1                 0.9127    0.1074    8.5005    0.0000
```

lmer version

```
> f <- lmer(y1 ~ 1 + x1.c + x1.mean + w1 + (1 | cluster), data = Demo.twolevel, REM
> summary(f, corr = FALSE)
```

```
Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: y1 ~ 1 + x1.c + x1.mean + w1 + (1 | cluster)
   Data: Demo.twolevel

     AIC      BIC   logLik deviance df.resid
  8905.3   8940.3  -4446.7   8893.3     2494

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.4655 -0.6526  0.0106  0.6238  3.3701

Random effects:
 Groups   Name        Variance Std.Dev.
 cluster  (Intercept) 0.9127   0.9553
 Residual             1.7601   1.3267
Number of obs: 2500, groups:  cluster, 200

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.01508    0.07383   0.204
x1.c         0.49342    0.02782  17.738
x1.mean      0.53432    0.24984   2.139
w1           0.15980    0.07891   2.025
```

adding interaction terms (moderation)

- just like in ordinary regression, interaction terms are products of the main terms, for example:

```
> Demo.twolevel$x1.x2 <- Demo.twolevel$x1 * Demo.twolevel$x2
```

- manifest group-mean centering: center first, then take the product (C1P2)

```
> model <- '
  level: 1
     y1 ~ x1.c + x2.c + x1c.x2c
  level: 2
     y1 ~~ y1
  '
```

- manifest group-mean centering: first the product, then centering (P1C2)

```
> model <- '
  level: 1
     y1 ~ x1.c + x2.c + x1.x2.c
  level: 2
     y1 ~~ y1
  '
```

- literature about P1C2 versus C1P2:

  Loeys, T., Josephy, H., & Dewitte, M. (2018). More precise estimation of lower-level interaction effects in multilevel models. *Multivariate behavioral research, 53*, 335–347.

- latent centering: first add the product term (eg. `x1.x2`) to the data.frame; then specify the product term at both levels

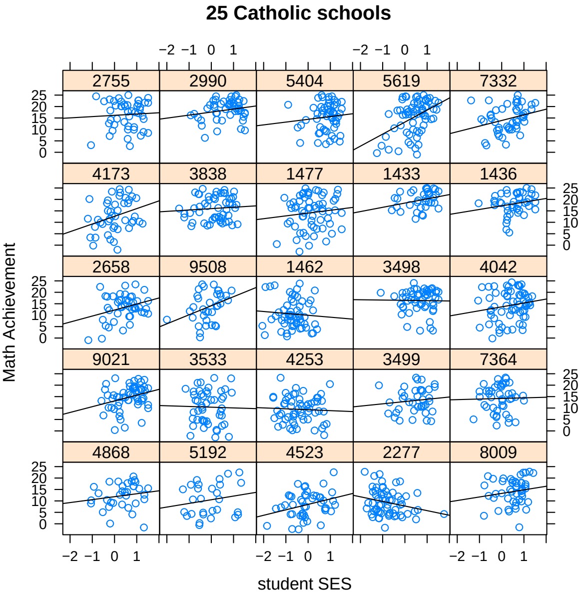
```
> model <- '
  level: 1
     y1 ~ x1 + x2 + x1.x2
  level: 2
     y1 ~ x1 + x2 + x1.x2
  '
> # will not work for this dataset, because x1 and x2 have almost no
> # between level variance
```

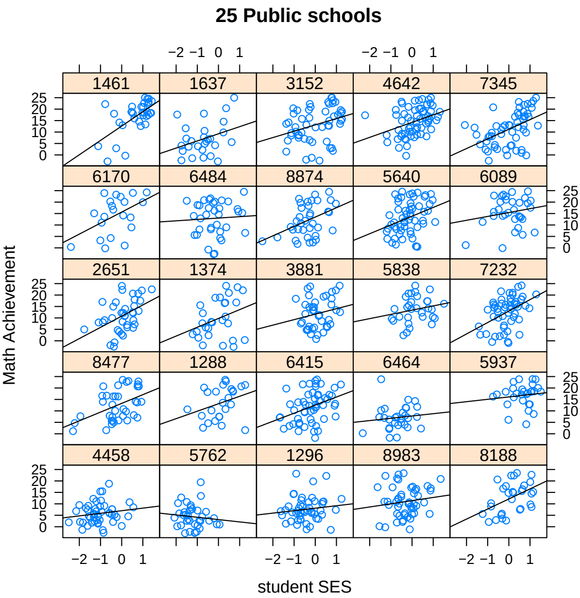
7.4 Two-level regression with random slopes

- the following example is borrowed from Raudenbusch and Bryk (2001)
- the data are from the 1982 "High School and Beyond" survey, and pertain to 7185 U.S. high-school students from 160 schools (70 catholic, 90 public)
- these are the variables that we will use:
  - **school** an ordered factor designating the school that the student attends.
  - **mAch** a numeric vector of Mathematics achievement scores
  - **ses** a numeric vector of socio-economic scores
  - **cses** a numeric vector of centered ses values where the centering is with respect to the meanses for the school
  - **meanses** a numeric vector of mean ses for the school
  - **sector** a factor with levels Public and Catholic
- the aim of the analysis is to determine how students' math achievement scores are related to their family socioeconomic status

exploring the data

```
> library(mlmRev)
> summary(Hsb82)

     school     minrty          sx             ses                  mACh
 2305   :  67   No :5211   Male  :3390   Min.   :-3.758000   Min.   :-2.832
 5619   :  66   Yes:1974   Female:3795   1st Qu.:-0.538000   1st Qu.: 7.275
 4292   :  65                            Median : 0.002000   Median :13.131
 8857   :  64                            Mean   : 0.000143   Mean   :12.748
 4042   :  64                            3rd Qu.: 0.602000   3rd Qu.:18.317
 3610   :  64                            Max.   : 2.692000   Max.   :24.993
 (Other):6795
    meanses                sector           cses
 Min.   :-1.1939459   Public  :3642   Min.   :-3.6507
 1st Qu.:-0.3230000   Catholic:3543   1st Qu.:-0.4479
 Median : 0.0320000                   Median : 0.0160
 Mean   : 0.0001434                   Mean   : 0.0000
 3rd Qu.: 0.3269123                   3rd Qu.: 0.4694
 Max.   : 0.8249825                   Max.   : 2.8561
```


25 Catholic schools
2755
2990
5404
5619
7332
4173
3838
1477
1433
1436
2658
9508
1462
3498
4042
9021
3533
4253
3499
7364
4868
5192
4523
2277
8009
Math Achievement
student SES
−2 −1 0 1
0 5 10 15 20 25


25 Public schools
1461
1637
3152
4642
7345
6170
6484
8874
5640
6089
2651
1374
3881
5838
7232
8477
1288
6415
6464
5937
4458
5762
1296
8983
8188
Math Achievement
student SES
−2 −1 0 1
0
5
10
15
20
25

model 1: a random-coefficients regression model

- 1 level-1 predictor (SES, centered within school), no level-2 predictors
- random intercept and random slopes
- model for the first (student) level (using HLM type notation)

$$y_{ji} = \alpha_{0j} + \alpha_{1j}\,\text{cses}_{ji} + \epsilon_{ji}$$

- model for the second (school) level:

$$\begin{aligned} \alpha_{0j} &= \gamma_{00} + u_{0j} & \text{(the random intercept)} \\ \alpha_{1j} &= \gamma_{10} + u_{1j} & \text{(the random slope)} \end{aligned}$$

- the combined model and the Laird-Ware form:

$$\begin{aligned} y_{ji} &= (\gamma_{00} + u_{0j}) + (\gamma_{10} + u_{1j})\,\text{cses}_{ji} + \epsilon_{ji} \\ &= \gamma_{00} + \gamma_{10}\,\text{cses}_{ji} + u_{0j} + u_{1j}\,\text{cses}_{ji} + \epsilon_{ji} \\ &= \beta_0 + \beta_1 x_{1ji} + b_{0j} + b_{1j} z_{1ji} + \epsilon_{ji} \end{aligned}$$

- the fixed-effect coefficients β_0 and β_1 represent the average within-schools population intercept and slope respectively
- the model has four variance-covariance components:
  - $\mathsf{Var}(b_{0j}) = d_0^2$: the variance among school intercepts
  - $\mathsf{Var}(b_{1j}) = d_1^2$: the variance among school slopes
  - $\mathsf{Cov}(b_{0j}, b_{1j}) = d_{01}$: the covariance between within-school intercepts and slopes
  - $\mathsf{Var}(\epsilon_{ji}) = \sigma^2$: the error variance around the within-school regressions

R code

```
> fit.model3 <- lmer(mAch ~ 1 + cses + (1 + cses | school), data = Hsb82,
                     REML = FALSE)
> summary(fit.model3, correlation = FALSE)
```

```
Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: mAch ~ 1 + cses + (1 + cses | school)
   Data: Hsb82

     AIC      BIC   logLik deviance df.resid
 46723.0  46764.3 -23355.5  46711.0     7179

Scaled residuals:
     Min       1Q   Median       3Q      Max
-3.09688 -0.73198  0.01794  0.75445  2.89902

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 school   (Intercept)  8.6206  2.9361
          cses         0.6782  0.8235   0.02
 Residual             36.7000  6.0581
Number of obs: 7185, groups:  school, 160

Fixed effects:
            Estimate Std. Error t value
(Intercept)  12.6363     0.2437   51.85
cses          2.1932     0.1278   17.16
```

model 2: intercepts-and-slopes-as-outcomes model

- we expand the model by including two level-2 predictors: meanses and sector; the slopes are allowed to vary randomly
- model for the first (student) level:

$$y_{ji} = \alpha_{0j} + \alpha_{1j}\,\text{cses}_{ji} + \epsilon_{ji}$$

- model for the second (school) level:

$$\alpha_{0j} = \gamma_{00} + \gamma_{01}\,\text{meanses}_j + \gamma_{02}\,\text{sector}_j + u_{0j}$$
$$\alpha_{1j} = \gamma_{10} + \gamma_{11}\,\text{meanses}_j + \gamma_{12}\,\text{sector}_j + u_{1j}$$

- the combined model and the Laird-Ware form:

$$
\begin{aligned}
y_{ji} = &(\gamma_{00} + \gamma_{01}\,\text{meanses}_j + \gamma_{02}\,\text{sector}_j + u_{0j}) + \\
&(\gamma_{10} + \gamma_{11}\,\text{meanses}_j + \gamma_{12}\,\text{sector}_j + u_{1j})\,\text{cses}_{ji} + \epsilon_{ji} \\
= &\gamma_{00} + \gamma_{01}\,\text{meanses}_j + \gamma_{02}\,\text{sector}_j + \gamma_{10}\,\text{cses}_{ji} + \\
&\gamma_{11}\,\text{meanses}_j\,\text{cses}_{ji} + \gamma_{12}\,\text{sector}_j\,\text{cses}_{ji} + u_{0j} + u_{1j}\,\text{cses}_{ji} + \epsilon_{ji} \\
= &\beta_0 + \beta_1 x_{1ji} + \beta_2 x_{2ji} + \beta_3 x_{3ji} + \beta_4 (x_{1ji} x_{3ji}) + \beta_5 (x_{2ji} x_{3ji}) + \\
&b_{0j} + b_{1j} z_{1ji} + \epsilon_{ji}
\end{aligned}
$$

R code

```
> fit.model4 <- lmer(mAch ~ 1 + meanses*cses + sector*cses + (1 + cses | school),
                     data = Hsb82, REML = FALSE)
> summary(fit.model4, correlation = FALSE)

Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: mAch ~ 1 + meanses * cses + sector * cses + (1 + cses | school)
   Data: Hsb82

     AIC      BIC   logLik deviance df.resid
 46516.4  46585.2 -23248.2  46496.4     7175
```

```
Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.1610 -0.7244  0.0168  0.7549  2.9581

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 school   (Intercept)  2.31666 1.5221
          cses         0.06512 0.2552   0.48
 Residual             36.72116 6.0598
Number of obs: 7185, groups:  school, 160

Fixed effects:
                    Estimate Std. Error t value
(Intercept)          12.1279     0.1974  61.441
meanses               5.3317     0.3655  14.586
cses                  2.9457     0.1540  19.128
sectorCatholic        1.2269     0.3033   4.046
meanses:cses          1.0427     0.2960   3.522
cses:sectorCatholic  -1.6440     0.2373  -6.926
```

do we need the random slopes?

```
> fit.model4bis <- lmer(mAch ~ 1 + meanses*cses + sector*cses + (1 | school),
                        data = Hsb82, REML = FALSE)
> anova(fit.model4bis, fit.model4)

Data: Hsb82
Models:
fit.model4bis: mAch ~ 1 + meanses * cses + sector * cses + (1 | school)
fit.model4: mAch ~ 1 + meanses * cses + sector * cses + (1 + cses | school)
              npar   AIC   BIC logLik deviance  Chisq Df Pr(>Chisq)
fit.model4bis    8 46513 46568 -23249    46497
fit.model4      10 46516 46585 -23248    46496 1.0016  2     0.6061
```

- no: apparently, the level-2 predictors do a sufficiently good job of accounting for differences in slopes
- statistical note: using a LRT for comparing two models with a different random structure is conservative; better approaches exist (e.g. in the R package RLRsim)

more reading

Hox, J.J., Moerbeek, M., & van den Schoot, R., (2018, 3rd edition). *Multilevel analysis: Techniques and applications.* Routledge.

Snijders, T.A., & Bosker, R.J. (2011, 2nd edition). *Multilevel analysis: An introduction to basic and advanced multilevel modeling*. Sage.

Finch, W.H., Bolin, J.E., & Kelley, K. (2019). *Multilevel modeling using R*. Routledge.

8 Multilevel SEM

8.1 Introduction

- limitations of the multilevel regression model:
  - (mostly) univariate perspective (multivariate is possible but awkward)
  - no measurement models (latent variables)
  - no mediators (only strictly dependent or independent variables)
  - no reciprocal effects, no goodness-of-fit measures, ...
- two evolutions since the late 1980s:
  - the multilevel regression framework was extended to include measurement errors and latent variables (cfr. HLM and MLwiN software)
  - the traditional SEM framework started to incorporate random intercepts and random slopes
- the boundaries between SEM and multilevel regression have gradually disappeared

8.2 History (optional)

Schmidt, W. H. (1969)

- Schmidt, W.H. (1969). Covariance structure analysis of the multivariate random effects model (Doctoral dissertation, University of Chicago, Department of Education).
  - full modeling of within and between covariance matrices
  - provided a computer program for ML estimation
  - balanced data only, no level-2 variables, no meanstructure
  - structured case is described in Schmidt & Wisenbaker (1986)
- Gustafsson, J.E., & Lindström, B. (1979). Analyzing ATI Data By Structural Analysis of Covariance Matrices. (Paper presented at the Annual Meeting of the AERA, San Fransisco, April 8–12, 1979) – Examples 7 + 8

> LISREL also offers great possibilities for conducting such multilevel analyses. It has been shown by Schmidt (1969) that maximum likelihood estimates can be derived of the within- class and between-class covariance matrices, and these can

> be parameterized in LISREL models, to allow separate estimates of parameters at the two levels [...] A great problem, of course, is that there in most studies tend to be few classes (or other higher level units) only, which precludes the possibility of obtaining any stable estimates at the class level. We would like to suggest, however, that in the least within-class analyses should be performed to guard against the possibility that results obtained in non-hierarchical analyses can in fact he accounted for by effects at the class level, which may be more or less artifactual.

- his work was also picked up by Leigh Burstein
  - Burstein, L. (1980). The analysis of multilevel data in educational research and evaluation. *Review of research in education, 8*, 158–233.
  - Burstein worked at the Graduate School of Education (UCLA)
- also cited in Muthén, B.O. (1989). Latent variable modeling in heterogeneous populations. *Psychometrika, 54*, 557–585
  - he reformulated Schmidt's fitting function so that it could be estimated using existing software for multiple-group SEM (e.g., LISCOMP)

Goldstein & McDonald

- Goldstein, H., & McDonald, R.P. (1988). A general model for the analysis of multilevel data. *Psychometrika, 53*, 455–467.
  - very general formulation, including multilevel SEM
  - univariate perspective (multivariate vector = 1st level)
  - can handle missing data, hierchical data, cross-classified data
  - expression of the likelihood, IGLS algorithm is suggested
- McDonald, R.P., & Goldstein, H. (1989). Balanced versus unbalanced designs for linear structural relations in two-level data. *British Journal of mathematical and statistical psychology, 42*, 215–232.
  - multivariate perspective, within-and-between formulation
  - likelihood expression + a computationally tractable (re)expression
  - both for balanced and unbalanced clusters
- McDonald, R.P. (1993). A general model for two-level data with responses missing at random. *Psychometrika*, 58, 575–585.

Muthén

- Muthén, B.O. (1989). Latent variable modeling in heterogeneous populations. *Psychometrika, 54*, 557–585
  - re-expresses the within-part of the likelihood as a sum over different cluster sizes
  - in the balanced case, this leads to a multiple-group SEM fitting function with two groups
- Muthén, B.O. (1990). Mean and covariance structure analysis of hierarchical data. Department of Statistics, UCLA. (unpublished technical report)
  - derivations of Muthén (1989)
  - suggestion: we can use the balanced solution even in the unbalanced case (using an estimate of the average cluster size): estimator = MUML
  - more discussion in Muthén, B.O. (1994). Multilevel covariance structure analysis. *Sociological methods & research, 22*, 376–398.
- standard SEM software could be used (at least for the balanced case)

Lee

- Lee, S.Y. (1990). Multilevel analysis of structural equation models. *Biometrika, 4*, 763–772.
  - statistically more rigorous development of multilevel SEM theory: ML and GLS estimation, inference, goodness-of-fit statistics, constraints
  - suggested using Fisher scoring and Gauss-Newton for optimization
  - no level-2 variables
- Poon & Lee (1992): within-part as sum over different cluster sizes
- Yau, Lee & Poon (1993): three-level setting
- Lee & Poon (1998): using the EM algorithm (by treating the the latent random vectors at the cluster level as missing data)
- Lee, S.Y. (2007). *Structural equation modeling: A Bayesian approach*. John Wiley & Sons.
  - Chapter 9: Bayesian methods for analyzing various two-level SEMs

Bentler

- Liang, J., & Bentler, P.M. (2004). An EM algorithm for fitting two-level structural equation models. *Psychometrika, 69*, 101–122.
  - earlier work: Benter & Liang (2000?), Bentler & Liang (2003), Liang & Bentler (2003)
  - extend the EM algorithm of Lee & Poon (1998) to handle level-2 predictors
  - clever way to avoid a large number of matrix inversions
  - often considered to be the state-of-the-art algorithm for estimating 2-level SEMs with continuous responses
  - no missing data, no random slopes
- perhaps the last technical paper on (continuous) two-level SEM (in the frequentist framework)

8.3 Frameworks (and software) for multilevel SEM

overview

- two-level SEM with random intercepts
  - Mplus (type = twolevel), LISREL, EQS, lavaan
- the gllamm framework: gllamm, (related approach: Latent Gold)
- the Mplus framework: Mplus
- the case-wise likelihood based approach (e.g., Mehta & Neale, 2005)
  - Mplus (type = random), Mx, OpenMx (definition variables)
  - in principle: both continuous and categorical outcomes; random slopes
  - xxM?
- the Bayesian framework
  - Mplus
  - (Open)BUGS, JAGS, Stan

two-level SEM with random intercepts

- an extension of single-level SEM to incorporate random intercepts
- extensive technical literature, starting from the late 1980s (until about 2004)
- available in Mplus, EQS, LISREL, lavaan, . . .
- this is by far the most widely used framework in the applied literature
- advantages:
  - fast, simple, well-understood, plenty of examples
  - well-documented
- disadvantages:
  - continuous outcomes only
  - no random slopes

the Mplus framework

- the Mplus framework has added many extensions to the two-level within/between approach in the last 20 years
  - EM algorithm can handle random slopes and missing data
  - categorical outcomes (with numerical quadrature)
  - multilevel (robust) (D)WLS
  - combination multilevel with complex survey data, mixture modeling, . . .
- advantages:
  - superb implementation
  - user-friendly, familiar (‘multivariate’) approach
- disadvantages:
  - NO technical documentation (about the extensions)
  - black box software

the gllamm framework

- Sophia Rabe-Hesketh, Anders Skrondal and Andrew Pickles
- see `http://www.gllamm.org/`
- an extension of generalized linear mixed models to include (continuous and discrete) latent variables (including a structural part)
- advantages:
  - very well documented, open-source code (written in Stata)
  - handles a wide range of outcome types (normal, categorical, . . .)
  - very general, very flexible
- disadvantages:
  - not easy to specify (complex) models, univariate perspective
  - needs Stata
  - very, very slow (even in the continuous case)

lavaan

- multilevel SEM development started around jan 2017
- introduced in version 0.6-1:
  - standard two-level 'within-and-between' approach
  - continuous responses only, no missing data (for now)
  - no random slopes (for now)
  - using quasi-newton optimization by default
  - em algorithm available using the option `optim.method = "em"`
- future plans: many, but don't ask when it will be ready
  - random slopes, missing data (0.6-8 or 0.6-9)
  - gllamm framework (but more user-friendly)
  - case-wise likelihood approach
  - hybrids

lavaan syntax setup for two-level SEM

$\mathbf{\Sigma}_B$

Between

Within

$\mathbf{\Sigma}_W$

```
model <- '

  level: 1

    # here comes the within level

  level: 2

    # here comes the between level
'

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

useful literature

- the relationship between SEM and multilevel regression:

Curran, P.J. (2003). Have multilevel models been structural equation models all along? *Multivariate Behavioral Research, 38*, 529–569.

Bauer, D.J. (2003). Estimating Multilevel Linear Models as Structural Equation Models. *Journal of Educational and Behavioral Statistics, 28*, 135–167.

Mehta, P.D., and Neale, M.C. (2005). People are variables too: Multilevel structural equations modeling. *Psychological methods, 10*, 259–284.

- books:

Hox, J.J., Moerbeek, M., & van de Schoot, R. (2010). *Multilevel analysis: Techniques and applications*. Routledge.

Heck, R.H., & Thomas, S.L. (2015). *An introduction to multilevel modeling techniques: MLM and SEM approaches using Mplus*. Routledge.

Skrondal, A., & Rabe-Hesketh, S. (2004). *Generalized latent variable modeling: Multilevel, longitudinal, and structural equation models.* CRC Press.

Lee, Sik-Yum (2007). *Structural equation modeling: A Bayesian approach.* John Wiley & Sons.

8.4 The two-level SEM model with random intercepts

- we assume two-level data with individuals (students) nested within clusters (schools)
- in this framework, we decompose the total score of *each* variable into two parts: a within part, and a between part (Cronbach & Webb, 1979):

$$\begin{aligned}\mathbf{y}_{ji} &= (\mathbf{y}_{ji} - \bar{\mathbf{y}}_j) + \bar{\mathbf{y}}_j \\ \mathbf{y}_T &= \mathbf{y}_W + \mathbf{y}_B\end{aligned}$$

  where $j = 1, \ldots, J$ is an index for the clusters, and $i = 1, \ldots, n_j$ is an index for the units within a cluster; $\bar{\mathbf{y}}_j$ is the cluster mean of cluster j

  - both components are treated as unknown (latent) variables
  - the two parts are orthogonal and additive; one of the parts can be zero
- the total covariance (at the population level) can be decomposed as

$$\mathsf{Cov}(\mathbf{y}) = \mathbf{\Sigma}_T = \mathbf{\Sigma}_W + \mathbf{\Sigma}_B$$

two-level SEM: specifying a model for each level

- for a two-level CFA model, we can use

$$\boldsymbol{\Sigma}_W = \boldsymbol{\Lambda}_W \boldsymbol{\Psi}_W \boldsymbol{\Lambda}'_W + \boldsymbol{\Theta}_W$$

  and

$$\boldsymbol{\Sigma}_B = \boldsymbol{\Lambda}_B \boldsymbol{\Psi}_B \boldsymbol{\Lambda}'_B + \boldsymbol{\Theta}_B$$

- if we add a structural (regression) part, we need to add the $(\boldsymbol{I} - \boldsymbol{B})^{-1}$ term to the matrix formulation (as in regular SEM)
- meanstructure
  - within: $\boldsymbol{\mu}_W$ (usually all zero, as the level-1 variables are cluster-centered, except for within-only variables)
  - between: $\boldsymbol{\mu}_B$
- in addition, we can add level-2 covariates ($\mathbf{z}_j$) to the model

8.5 Loglikelihood of a two-level SEM (optional)

notation

- number of clusters: J, number of units per cluster: n_j
- data for cluster j:

$$\mathbf{v}_j = [\mathbf{z}_j, \mathbf{y}_{j1}, \mathbf{y}_{j2}, \ldots, \mathbf{y}_{jn_j}]^T$$

- model implied matrices/vectors: $\mathbf{\Sigma}_{zz}$, $\mathbf{\Sigma}_{zy}$, $\mathbf{\Sigma}_w$, $\mathbf{\Sigma}_b$ and $\boldsymbol{\mu}_b = [\boldsymbol{\mu}_z, \boldsymbol{\mu}_y]^T$
- expectation of $\mathbf{v}_j$:

$$\mathrm{E}[\mathbf{v}_j] = \hat{\mathbf{v}}_j = [\boldsymbol{\mu}_z, \boldsymbol{\mu}_{\boldsymbol{y}}, \boldsymbol{\mu}_{\boldsymbol{y}}, \ldots, \boldsymbol{\mu}_{\boldsymbol{y}}]^T$$

- covariance matrix for $\mathbf{v}_j$:

$$\mathrm{Cov}[\mathbf{v}_j] = \mathbf{V}_j = \begin{bmatrix} \mathbf{\Sigma}_{zz} & \mathbf{1}_{n_j}^T \otimes \mathbf{\Sigma}_{zy} \\ \mathbf{1}_{n_j} \otimes \mathbf{\Sigma}_{yz} & \mathbf{\Sigma}_{yy} \end{bmatrix}$$

where

$$\mathbf{\Sigma}_{yy} = \mathbf{I}_{n_j} \otimes \mathbf{\Sigma}_w + \mathbf{1}_{n_j}\mathbf{1}_{n_j}^T \otimes \mathbf{\Sigma}_b$$

loglikelihood

- assuming multivariate normality, we can write the loglikelihood for cluster j as follows:

$$\text{loglik}_j = -\frac{O_j}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{V}_j| - \frac{1}{2}(\mathbf{v}_j - \hat{\mathbf{v}}_j)^T \mathbf{V}_j^{-1}(\mathbf{v}_j - \hat{\mathbf{v}}_j)$$

  where O_j is the length of $\mathbf{v}_j$, usually $p_z + (n_j \times p_y)$

- the total likelihood over all J clusters:

$$\text{loglik} = \sum_{j=1}^{J} \text{loglik}_j$$

- we can find ML estimates by minimizing the objective function F_{ML} which is minus two times the loglikelihood function, ignoring the constant:

$$F_{ML} = \sum_{j=1}^{J} \ln|\mathbf{V}_j| + (\mathbf{v}_j - \hat{\mathbf{v}}_j)^T \mathbf{V}_j^{-1}(\mathbf{v}_j - \hat{\mathbf{v}}_j)$$

objective function (optional)

- the original objective function:

$$F_{ML} = \sum_{j=1}^{J} \ln |\mathbf{V}_j| + (\mathbf{v}_j - \hat{\mathbf{v}}_j)^T \mathbf{V}_j^{-1} (\mathbf{v}_j - \hat{\mathbf{v}}_j)$$

- for large clusters, the size of $\mathbf{V}_j$ can be formidable
- we should exploit the block-diagonal structure of $\mathbf{V}$
- we define:

$$\mathbf{\Sigma}_{b.z} = (\mathbf{\Sigma}_b - \mathbf{\Sigma}_{yz}\mathbf{\Sigma}_{zz}^{-1}\mathbf{\Sigma}_{zy})$$

- version 1: McDonald & Goldstein (1989), per cluster, using $\boldsymbol{\Sigma}_{b.z}$:

$$
\begin{aligned}
F_{ML} = \sum_{j=1}^{J} \Bigg[& \ln|\boldsymbol{\Sigma}_{zz}| + (n_j - 1)\ln|\boldsymbol{\Sigma}_w| + \ln|\boldsymbol{\Sigma}_w + n_j \cdot \boldsymbol{\Sigma}_{b.z}| \\
& + \mathrm{tr}\left[\left(\boldsymbol{\Sigma}_{zz}^{-1} + n_j \boldsymbol{\Sigma}_{zz}^{-1}\boldsymbol{\Sigma}_{zy}(n_j\boldsymbol{\Sigma}_{b.z} + \boldsymbol{\Sigma}_w)^{-1}\boldsymbol{\Sigma}_{yz}\boldsymbol{\Sigma}_{zz}^{-1}\right)(\mathbf{z}_j - \boldsymbol{\mu}_z)(\mathbf{z}_j \right. \\
& + 2n_j\,\mathrm{tr}\left[-\boldsymbol{\Sigma}_{zz}^{-1}\boldsymbol{\Sigma}_{zy}(n_j\boldsymbol{\Sigma}_{b.z} + \boldsymbol{\Sigma}_w)^{-1}(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)(\mathbf{z}_j - \boldsymbol{\mu}_z)^T\right] \\
& + \mathrm{tr}\left[\boldsymbol{\Sigma}_w^{-1}\mathbf{Y}_j^{(c)T}\mathbf{Y}_j^{(c)}\right] \\
& - n_j\mathrm{tr}\left[\boldsymbol{\Sigma}_w^{-1}(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)^T\right] \\
& + n_j\mathrm{tr}\left[(n_j\boldsymbol{\Sigma}_{b.z} + \boldsymbol{\Sigma}_w)^{-1}(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)^T\right] \Bigg]
\end{aligned}
$$

- version 2: lavaan = McDonald & Goldstein (1989), per cluster size,

$$
\begin{aligned}
F_{ML} = (N-J)\left(\ln|\boldsymbol{\Sigma}_w| + \text{tr}\left[\boldsymbol{\Sigma}_w^{-1} S_{pw}\right]\right) + \\
\sum_{s=1}^{S} n_s \cdot \Bigg[(\ln|\boldsymbol{\Sigma}_{zz}| + \ln|\boldsymbol{\Sigma}_w + n_j \cdot \boldsymbol{\Sigma}_{b.z}|) + \\
\text{tr}\left[\left(\boldsymbol{\Sigma}_{zz}^{-1} + n_j \boldsymbol{\Sigma}_{zz}^{-1}\boldsymbol{\Sigma}_{zy}(n_j\boldsymbol{\Sigma}_{b.z} + \boldsymbol{\Sigma}_w)^{-1}\boldsymbol{\Sigma}_{yz}\boldsymbol{\Sigma}_{zz}^{-1}\right)(\mathbf{z}_j - \boldsymbol{\mu}_z)(\mathbf{z}_j - \boldsymbol{\mu}_z)^T\right] \\
+ 2n_j \,\text{tr}\left(-\boldsymbol{\Sigma}_{zz}^{-1}\boldsymbol{\Sigma}_{zy}(n_j\boldsymbol{\Sigma}_{b.z} + \boldsymbol{\Sigma}_w)^{-1}(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)(\mathbf{z}_j - \boldsymbol{\mu}_z)^T\right) \\
+ n_j \text{tr}\left((n_j\boldsymbol{\Sigma}_{b.z} + \boldsymbol{\Sigma}_w)^{-1}(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)(\bar{\mathbf{y}}_j - \boldsymbol{\mu}_y)^T\right)\Bigg]
\end{aligned}
$$

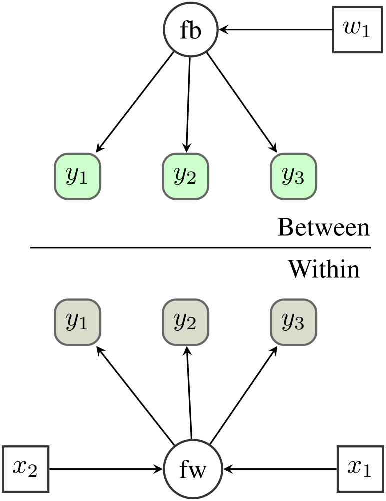
where S_{pw} is the pooled within-clusters covariance matrix:

$$
S_{pw} = \frac{\sum_{j=1}^{J}\sum_{i=1}^{n_j}(\mathbf{y}_{ji} - \bar{\mathbf{y}}_j)(\mathbf{y}_{ji} - \bar{\mathbf{y}}_j)^T}{N-J}
$$

optimization techniques for two-level SEM (optional)

- quasi-newton methods (lavaan, 0.6-1)
  - if the between-level covariance matrix becomes negative (definite), estimation will fail with an error message
  - negative variances are possible
- Fisher scoring, Gauss-Newton (LISREL)
- Expectation-Maximization (EM) (Mplus, EQS, lavaan)
  - in lavaan, use `optim.method = "em"`
  - will usually 'always' converge, if enough iterations are allowed
  - no negative variances
  - but we may not detect problematic models
  - many acceleration schemes are available

diagram + lavaan syntax using Demo.twolevel dataset




```
library(lavaan)

model <- '
    level: 1
        fw =~ y1 + y2 + y3

        fw ~ x1 + x2

    level: 2
        fb =~ y1 + y2 + y3

        fb ~ w1
'

fit <- sem(model, data = Demo.twolevel,
           cluster = "cluster",
           # optim.method = "em",
           fixed.x = FALSE)
```

summary() output

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

lavaan 0.6-7 ended normally after 42 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         25

  Number of observations                          2500
  Number of clusters [cluster]                     200

Model Test User Model:

  Test statistic                                 3.708
  Degrees of freedom                                 6
  P-value (Chi-square)                           0.716

Model Test Baseline Model:

  Test statistic                              2144.729
  Degrees of freedom                                15
  P-value                                        0.000

User Model versus Baseline Model:

  Comparative Fit Index (CFI)                    1.000
```

```
  Tucker-Lewis Index (TLI)                       1.003

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)             -19482.638
  Loglikelihood unrestricted model (H1)     -19480.784

  Akaike (AIC)                               39015.276
  Bayesian (BIC)                             39160.878
  Sample-size adjusted Bayesian (BIC)        39081.446

Root Mean Square Error of Approximation:

  RMSEA                                          0.000
  90 Percent confidence interval - lower         0.000
  90 Percent confidence interval - upper         0.019
  P-value RMSEA <= 0.05                          1.000

Standardized Root Mean Square Residual (corr metric):

  SRMR (within covariance matrix)                0.004
  SRMR (between covariance matrix)               0.023

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                0.777    0.035   22.165    0.000
    y3                0.735    0.034   21.880    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  fw ~
    x1                0.507    0.024   21.452    0.000
    x2                0.406    0.023   17.837    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  x1 ~~
    x2                0.001    0.020    0.042    0.966

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.000
   .y2                0.000
   .y3                0.000
    x1               -0.007    0.020   -0.376    0.707
```

```
    x2                -0.003    0.020   -0.144    0.886
   .fw                 0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                 0.990    0.047   21.144    0.000
   .y2                 1.065    0.040   26.876    0.000
   .y3                 1.010    0.037   27.302    0.000
   .fw                 0.589    0.043   13.738    0.000
    x1                 0.982    0.028   35.355    0.000
    x2                 1.011    0.029   35.355    0.000


Level 2 [cluster]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fb =~
    y1                 1.000
    y2                 0.713    0.053   13.352    0.000
    y3                 0.582    0.048   12.003    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  fb ~
    w1                 0.140    0.078    1.789    0.074

Intercepts:
```

```
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.015    0.075    0.207    0.836
   .y2               -0.022    0.060   -0.372    0.710
   .y3               -0.047    0.054   -0.870    0.384
    w1                0.052    0.066    0.794    0.427
   .fb                0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.052    0.049    1.070    0.285
   .y2                0.122    0.032    3.800    0.000
   .y3                0.151    0.028    5.368    0.000
   .fb                0.902    0.119    7.587    0.000
    w1                0.870    0.087   10.000    0.000
```

the unrestricted within and between variance/covariance matrices

```
> lavInspect(fit, "h1")

$within
$within$cov
   y1    y2    y3    x1    x2
y1 1.998
y2 0.787 1.672
y3 0.748 0.562 1.556
x1 0.488 0.392 0.376 0.982
x2 0.414 0.320 0.297 0.001 1.011

$within$mean
    y1     y2     y3     x1     x2
 0.001 -0.002 -0.002 -0.007 -0.003


$cluster
$cluster$cov
   y1    y2    y3    w1
y1 0.971
y2 0.653 0.587
y3 0.536 0.383 0.464
w1 0.120 0.116 0.032 0.870

$cluster$mean
    y1     y2     y3     w1
 0.017 -0.018 -0.045  0.052
```

the model-implied within and between variance/covariance matrices

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

$within
$within$cov
   y1    y2    y3    x1    x2
y1 1.998
y2 0.783 1.673
y3 0.742 0.576 1.555
x1 0.498 0.387 0.366 0.982
x2 0.411 0.319 0.302 0.001 1.011

$within$mean
    y1     y2     y3     x1     x2
-0.005 -0.004 -0.004 -0.007 -0.003


$cluster
$cluster$cov
   y1    y2    y3    w1
y1 0.971
y2 0.655 0.588
y3 0.535 0.381 0.462
w1 0.122 0.087 0.071 0.870

$cluster$mean
    y1     y2     y3     w1
 0.023 -0.017 -0.043  0.052
```

the icc for all variables

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

   y1    y2    y3    x1    x2
0.327 0.260 0.230 0.000 0.000
```

8.6 The status of a latent variable in a two-level SEM

- when a latent variable, representing a hypothetical construct, is introduced in a two-level model, we need to carefully reflect on the 'status' of this latent variable
  - are the indicators measured at the within or the between level?
  - is the construct of (theoretical) interest at the within level, the between level, or both?
  - how can we interpret the 'meaning' of the construct at the within/between level?
- based on the answers on these questions, we need to create the latent variable in a different way at the within and/or the between level
- this is (still today) a big source of confusion (and bad practices) in the literature

different types of latent variables

- we will discuss five different construct types:

  1. within-only construct
     - in this case, if we have no other level-2 variables, we may as well use a single-level SEM based on a pooled within-cluster covariance matrix
  2. between-only construct
  3. shared between-level construct
  4. configural (or contextual) construct
  5. shared and configural construct

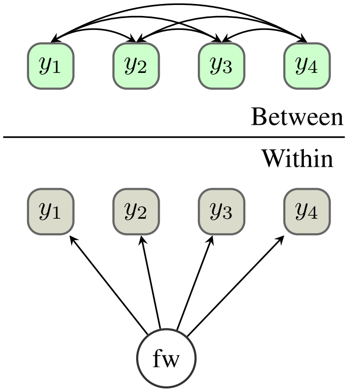
- reference:

Stapleton, L.M., Yang, J.S., & Hancock, G.R. (2016). Construct meaning in multilevel settings. *Journal of Educational and Behavioral Statistics, 41*, 481–520.

within-only construct

- indicators of the latent variable are measured at the within level
- level at which construct is of interest: within level only
- interpretation at the within level: construct explains the covariances between its indicators measured at the within level
- interpretation at the between level: not relevant
- although the construct only ‘exists’ at the within level, we may still observe ‘spurious’ between-level variation in the sample
- example: construct represents ‘lactose intolerance’
  - items inquire about the degree of severity of physical reactions after consuming products containing lactose
  - construct can not be a school-level characteristic, although we may observe differences (on average) across schools

diagram and lavaan syntax

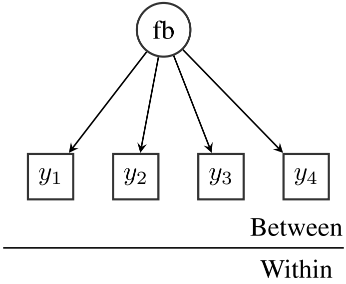



```
model <- '

  level: 1

    fw =~ y1 + y2 + y3 + y4

  level: 2

    y1 ~~ y1 + y2 + y3 + y4
    y2 ~~ y2 + y3 + y4
    y3 ~~ y3 + y4
    y4 ~~ y4
'
```

between-only construct

- indicators of the latent variable are measured at the between level
- level at which construct is of interest: between level only
- interpretation at the within level: not relevant (does not ‘exist’ at the within level)
- interpretation at the between level: construct explains the covariances between its indicators measured at the between level
- example: construct reflects self-reported ‘leadership style’ measured by a questionnaire filled in by the school principles

diagram and lavaan syntax

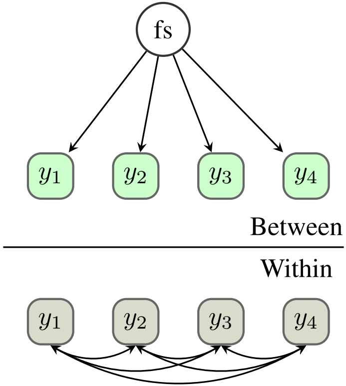



```
model <- '
  level: 1
    # perhaps other level-1 variables
  level: 2
    fb =~ y1 + y2 + y3 + y4
'
```

shared (or reflective) between-level construct

- indicators of the latent variable are measured at the within level
- level at which construct is of interest: between level only
- interpretation at the within level: none
- interpretation at the between level: construct represents a characteristic of the cluster
- example: construct reflects ‘instructional quality’ (a classroom characteristic) as perceived by students
  - each student in each classroom was asked to judge the ‘instructional quality’ of the teacher of that classroom
  - we are interested in the ‘average’ responses of the individual students within each classroom
  - responses within each classroom should be highly correlated (high agreement) if indeed ‘instructional quality’ is a shared construct

diagram and lavaan syntax

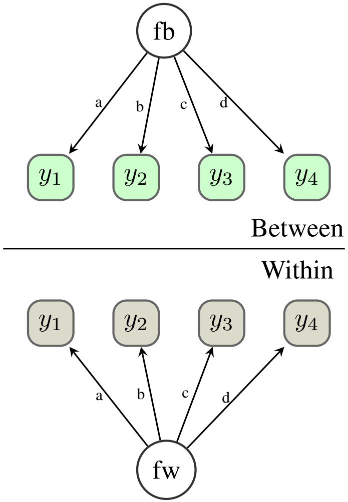



```
model <- '

  level: 1

    y1 ~~ y1 + y2 + y3 + y4
    y2 ~~ y2 + y3 + y4
    y3 ~~ y3 + y4
    y4 ~~ y4

  level: 2

    fs =~ y1 + y2 + y3 + y4

'
```

configural (or formative) construct

- indicators of the latent variable are measured at the within level
- level at which construct is of interest: both within and between level
- interpretation at the within/between level: construct explains the covariances of the within/between part of its indicators
- the configural construct (at the between level) represents the *aggregate* of the measurements of individuals within a cluster
- example: reading motivation:
  - at the individual level (within cluster)
  - at the school level (average student motivation within a school)
- the cluster itself is not seen as the source/reason for variability of an individual construct
- therefore, between-cluster loadings are fixed to be the same as within-cluster loadings (cross-level measurement invariance)

diagram and lavaan syntax

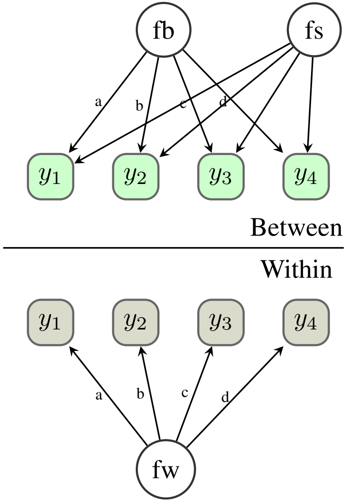



```
model <- '

  level: 1

    fw =~ a*y1 + b*y2 + c*y3 + d*y4

  level: 2

    fb =~ a*y1 + b*y2 + c*y3 + d*y4

    # optional (cross-level invariance)
    # y1 ~~ 0*y1
    # y2 ~~ 0*y2
    # y3 ~~ 0*y3
    # y4 ~~ 0*y4
'
```

shared + configural construct

- indicators of the latent variable are measured at the within level
- level at which construct is of interest: within and between level
- interpretation at the within level: construct explains the covariances of the within part of its indicators
- interpretation at the between level: both the configural construct and the shared construct explain the covariances of the within/between part of its indicators
- example: reading motivation for each child in a classroom is rated by the classroom teacher (using multiple items)
  - some teachers tend to rate more positively as compared to others
  - the 'shared' construct reflects the rater effect
  - the 'configural' construct reflects the average reading motivation in a classroom

diagram and lavaan syntax

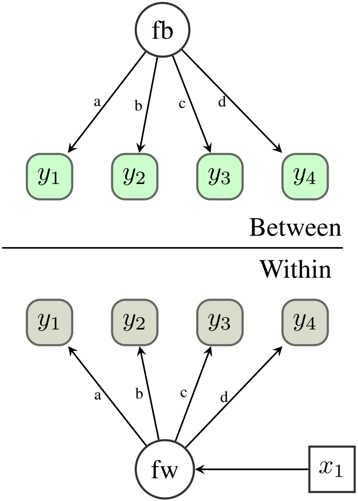



```
model <- '

  level: 1

    fw =~ a*y1 + b*y2 + c*y3 + d*y4

  level: 2

    fb =~ a*y1 + b*y2 + c*y3 + d*y4

    fs =~ y1 + y2 + y3 + y4

    # fb and fs must be orthogonal
    fs ~~ 0*fb

'
```

8.7 The status of observed covariates in a two-level SEM

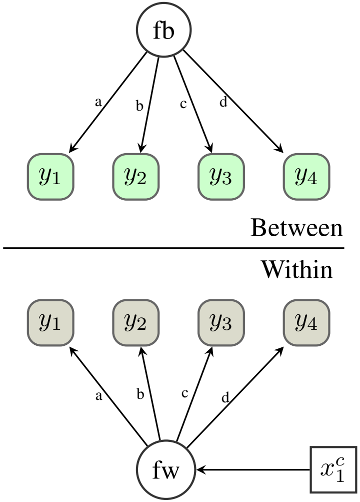
- when observed covariates are added in a two-level model, we again need to carefully reflect on the ‘status’ of these covariates
  - are the covariates measured at the within or the between level?
  - if they are measured at the within level, does it make sense to split this covariate into a within and a between part?
- based on the answers on these questions, we can make a distinction between three types of covariates:
  1. within-only covariates (uncentered, or group-mean centered)
  2. between-only covariates
  3. level-1 covariates with a within and a between part (manifest or latent centering)

adding a within-only covariate (uncentered)



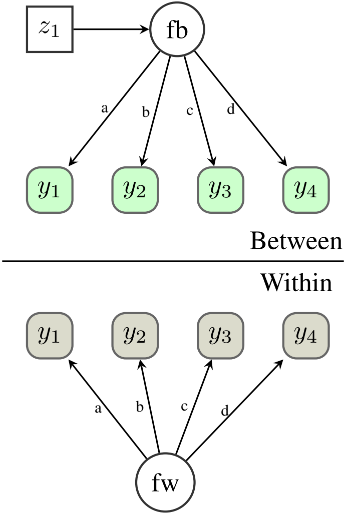

```
model <- '
  level: 1
    fw =~ a*y1 + b*y2 + c*y3 + d*y4
    fw ~ x1
  level: 2
    fb =~ a*y1 + b*y2 + c*y3 + d*y4
'
```

adding a within-only covariate (group-mean centered)



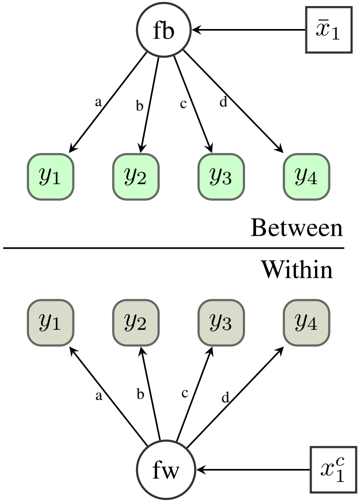

```
model <- '
  level: 1
    fw =~ a*y1 + b*y2 + c*y3 + d*y4
    fw ~ x1.c
  level: 2
    fb =~ a*y1 + b*y2 + c*y3 + d*y4
'
```

adding a between-only covariate



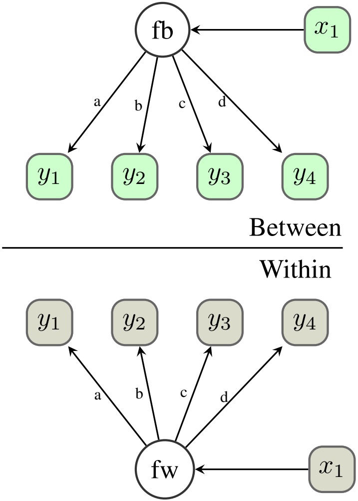

```
model <- '
  level: 1
    fw =~ a*y1 + b*y2 + c*y3 + d*y4
  level: 2
    fb =~ a*y1 + b*y2 + c*y3 + d*y4
    fb ~ z1
'
```

adding a level-1 covariate with a within and a between part (manifest)




```
model <- '
  level: 1
    fw =~ a*y1 + b*y2 + c*y3 + d*y4
    fw ~ x1.c
  level: 2
    fb =~ a*y1 + b*y2 + c*y3 + d*y4
    fb ~ x1.mean
'
```

adding a level-1 covariate with a within and a between part (latent)




```
model <- '
  level: 1
    fw =~ a*y1 + b*y2 + c*y3 + d*y4
    fw ~ x1
  level: 2
    fb =~ a*y1 + b*y2 + c*y3 + d*y4
    fb ~ x1
'
```

8.8 Evaluating model fit

- if no random slopes are involved, we can fit an unrestricted (saturated) model: we estimate all the elements of $\mathbf{\Sigma}_W$, $\mathbf{\Sigma}_B$ and $\boldsymbol{\mu}_B$
- then, we can compute the standard 'χ^2' goodness-of-fit test statistic as:

$$T = -2(L_0 - L_1)$$

  where L_0 and L_1 are the loglikelihood of the restricted (user-specified) model (h0) and the unrestricted model (h1) respectively

  - under various optimal conditions, this statistic follows a chi-square distribution
  - the degrees of freedom are computed as in a two-group SEM model: the difference between the number of (non-redundant) sample statistics for each level, and the number of free model parameters

- in principle, fit measures like CFI/TLI, RMSEA, SRMR, ... can be computed in a similar way as in a single-level SEM

evaluating fit (2)

- unfortunately, a recent simulation study showed that CFI, TLI, and RMSEA were not sensitive to Level-2 model misspecification:

  Hsu, H.Y., Kwok, O.M., Lin, J.H., & Acosta, S. (2015). Detecting misspecified multilevel structural equation models with common fit indices: a Monte Carlo study. *Multivariate behavioral research, 50*, 197–215.

- there seems to be a growing sentiment that 'global' fit indices may not be very useful in a multilevel setting

- an alternative approach is to assess the fit per level:
  - we could compute the SRMR for each level
  - we could fit a single-level model separately for each level, and look at the traditional fit measures to judge the model fit for that level

8.9 Example: two-level CFA

- we use an example from this book (Chapter 14):

  Hox, J.J., Moerbeek, M., & van de Schoot, R. (2010). *Multilevel analysis: Techniques and applications*. Routledge.

- the (simulated) data are the scores on six intelligence measures of 399 children from 60 (large) families, patterned after a real dataset collected by Van Peet, A.A.J. (1992)

- the six intelligence measures are: word list, cards, matrices, figures, animals, and occupations

- the data have a two-level structure, with children nested within families

- if intelligence is strongly influenced by shared genetic and environmental influences in the families, we may expect strong between-family effects

- the ICCs of the 6 measures range from 0.36 to 0.49

exploring the data

```
> FamIQData <- read.table("FamIQData.dat")
> names(FamIQData) <- c("family", "child", "wordlist", "cards", "matrices",
                        "figures", "animals", "occupats")
> summary(FamIQData)

     family          child           wordlist          cards
 Min.   : 1.00   Min.   : 1.00   Min.   :12.00   Min.   :11.00
 1st Qu.:16.00   1st Qu.: 2.00   1st Qu.:27.00   1st Qu.:26.50
 Median :33.00   Median : 4.00   Median :30.00   Median :30.00
 Mean   :31.78   Mean   : 4.04   Mean   :29.95   Mean   :29.84
 3rd Qu.:48.00   3rd Qu.: 6.00   3rd Qu.:33.00   3rd Qu.:33.00
 Max.   :60.00   Max.   :12.00   Max.   :45.00   Max.   :44.00
    matrices        figures          animals          occupats
 Min.   :15.00   Min.   :17.00   Min.   :15.00   Min.   :15.00
 1st Qu.:26.00   1st Qu.:27.00   1st Qu.:27.00   1st Qu.:27.00
 Median :30.00   Median :30.00   Median :30.00   Median :30.00
 Mean   :29.73   Mean   :30.08   Mean   :30.11   Mean   :30.01
 3rd Qu.:33.00   3rd Qu.:33.00   3rd Qu.:34.00   3rd Qu.:33.00
 Max.   :46.00   Max.   :44.00   Max.   :46.00   Max.   :43.00


> # various cluster sizes
> table(table(FamIQData$family))

 4  5  6  7  8  9 10 11 12
 3 16 11 12 11  4  1  1  1
```

analytic procedure

- fitting a two-level model is often a stepwise procedure; below are the steps used by Joop Hox
- model 0: as a preliminary step, an EFA was carried out on the pooled within-clusters covariance matrix $\mathbf{S}_{PW}$
  - it was concluded that a 2-factor model fitted well at the within level
  - not shown here
- model 1: a two-factor model at the within level + a null model at the between level
  - a null model implies: zero variances and covariances for all (6) variables
  - if this model fits well, we would conclude that there is no between family structure at all
- model 2: a two-factor model at the within level + an independence model at the between level

  - independence model implies: estimated variances but zero covariances
  - if this model holds, there is family-level variance, but no substantively interesting structural model

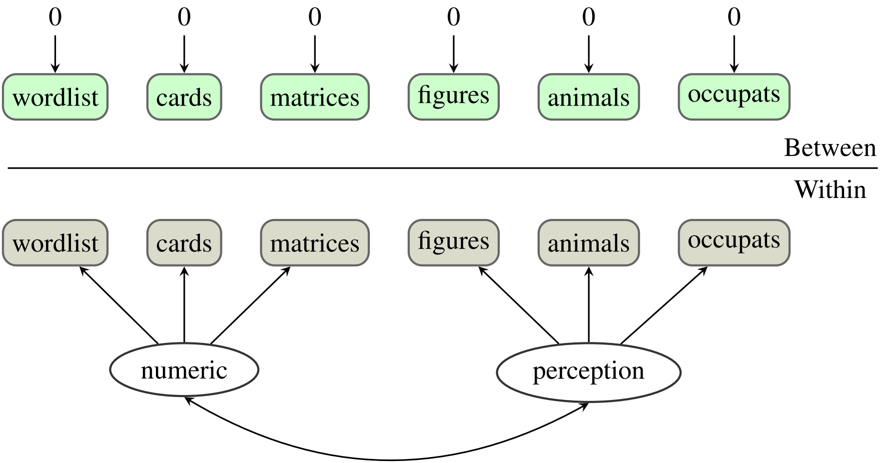
- model 3: a two-factor model at the within level + a saturated model at the between level

  - the factors at the within-level in this model correspond to what we have called 'within-only' constructs

- models 4a and 4b: in his book, Joop Hox goes on and fits a model with a one-factor model for the between part (4a), and a model with a two-factor model for the between part (4b)

  - the two-factor model seems no improvement over the one-factor model
  - model 4a (with a general factor at the between level) is kept as the final model

model 1: a 2-factor within model + null between model

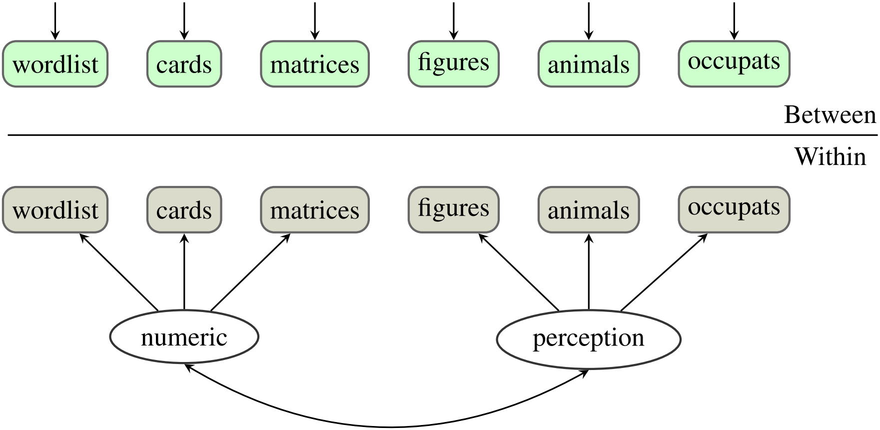


lavaan syntax

```
> model1 <- '
    level: 1
      numeric    =~ wordlist + cards + matrices
      perception =~ figures + animals + occupats
    level: 2
      wordlist ~~ 0*wordlist
      cards    ~~ 0*cards
      matrices ~~ 0*matrices
      figures  ~~ 0*figures
      animals  ~~ 0*animals
      occupats ~~ 0*occupats
  '
> fit1 <- sem(model1, data = FamIQData, cluster = "family",
              std.lv = TRUE, verbose = FALSE)
> # summary(fit1)
> fit1

lavaan 0.6-7 ended normally after 36 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         19

  Number of observations                           399
  Number of clusters [family]                       60
```

```
Model Test User Model:

  Test statistic                               323.649
  Degrees of freedom                                29
  P-value (Chi-square)                           0.000
```

model 2: a 2-factor within model + independence between model



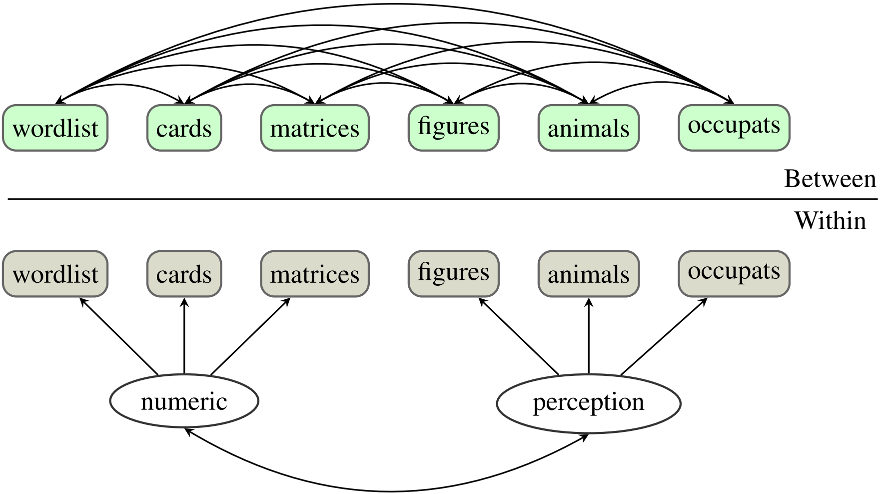
lavaan syntax model 2

```
> model2 <- '
    level: 1
      numeric    =~ wordlist + cards + matrices
      perception =~ figures + animals + occupats
    level: 2
      wordlist ~~ wordlist
      cards    ~~ cards
      matrices ~~ matrices
      figures  ~~ figures
      animals  ~~ animals
      occupats ~~ occupats
  '
> fit2 <- sem(model2, data = FamIQData, cluster = "family",
              std.lv = TRUE, verbose = FALSE)
> # summary(fit2)
> fit2
```

```
lavaan 0.6-7 ended normally after 43 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         25

  Number of observations                           399
  Number of clusters [family]                       60
```

```
Model Test User Model:

  Test statistic                               177.206
  Degrees of freedom                                23
  P-value (Chi-square)                           0.000
```

model 3: a 2-factor within model, with saturated between part



lavaan syntax model 3

```
> model3 <- '
    level: 1
      numeric    =~ wordlist + cards + matrices
      perception =~ figures + animals + occupats
    level: 2
      # saturated
      wordlist ~~ cards + matrices + figures + animals + occupats
      cards    ~~ matrices + figures + animals + occupats
      matrices ~~ figures + animals + occupats
      figures  ~~ animals + occupats
      animals  ~~ occupats
  '
> fit3 <- sem(model3, data = FamIQData, cluster = "family",
              std.lv = TRUE, verbose = FALSE)
> summary(fit3)
```

```
lavaan 0.6-7 ended normally after 157 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         40

  Number of observations                           399
  Number of clusters [family]                       60

Model Test User Model:
```

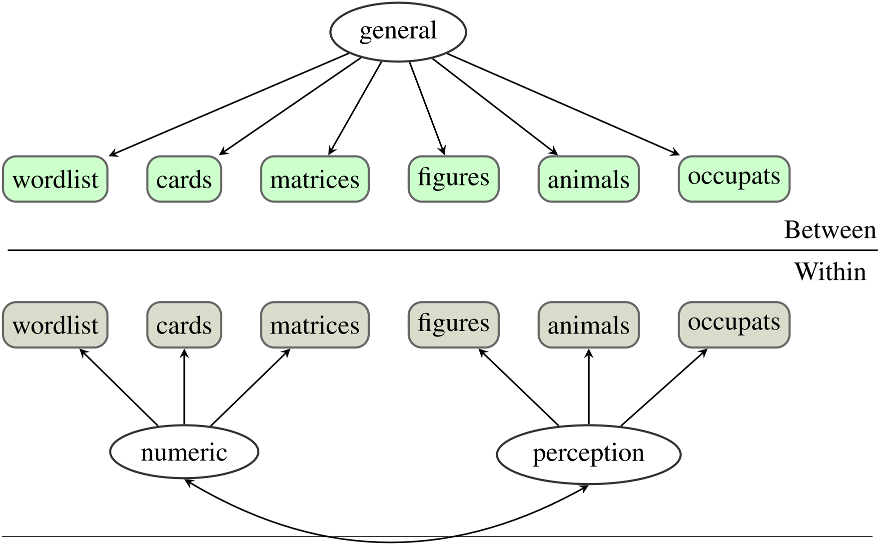
```
  Test statistic                                 6.716
  Degrees of freedom                                 8
  P-value (Chi-square)                           0.568

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian


Level 1 [within]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  numeric =~
    wordlist          3.155    0.203   15.558    0.000
    cards             3.156    0.196   16.113    0.000
    matrices          3.032    0.199   15.207    0.000
  perception =~
    figures           3.091    0.205   15.069    0.000
    animals           3.192    0.195   16.397    0.000
    occupats          2.774    0.183   15.139    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  numeric ~~
```

```
    perception        0.386    0.058    6.691    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .wordlist          0.000
   .cards             0.000
   .matrices          0.000
   .figures           0.000
   .animals           0.000
   .occupats          0.000
    numeric           0.000
    perception        0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .wordlist          6.234    0.739    8.433    0.000
   .cards             5.344    0.693    7.705    0.000
   .matrices          6.443    0.714    9.025    0.000
   .figures           6.856    0.757    9.053    0.000
   .animals           4.851    0.696    6.968    0.000
   .occupats          5.338    0.604    8.835    0.000
    numeric           1.000
    perception        1.000


Level 2 [family]:

Covariances:
```

```
                   Estimate  Std.Err  z-value  P(>|z|)
 .wordlist ~~
   .cards             9.272    2.225    4.168    0.000
   .matrices          8.515    2.077    4.100    0.000
   .figures           8.410    2.053    4.097    0.000
   .animals           9.700    2.195    4.419    0.000
   .occupats         10.428    2.357    4.425    0.000
 .cards ~~
   .matrices          7.997    2.018    3.964    0.000
   .figures           8.424    2.035    4.140    0.000
   .animals          10.000    2.203    4.540    0.000
   .occupats         10.418    2.337    4.457    0.000
 .matrices ~~
   .figures           7.733    1.902    4.067    0.000
   .animals           8.022    1.966    4.081    0.000
   .occupats          9.000    2.142    4.203    0.000
 .figures ~~
   .animals           8.980    2.177    4.125    0.000
   .occupats          9.750    2.333    4.179    0.000
 .animals ~~
   .occupats         11.080    2.489    4.451    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .wordlist         29.890    0.470   63.547    0.000
   .cards            29.892    0.465   64.308    0.000
   .matrices         29.732    0.439   67.746    0.000
   .figures          30.047    0.459   65.476    0.000
```

```
   .animals          30.135    0.471   63.956    0.000
   .occupats         29.967    0.509   58.892    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .wordlist         10.727    2.456    4.368    0.000
   .cards            10.559    2.397    4.404    0.000
   .matrices          9.097    2.123    4.285    0.000
   .figures          10.051    2.321    4.330    0.000
   .animals          10.956    2.466    4.442    0.000
   .occupats         13.473    2.874    4.688    0.000
```

model 4a: a 2-factor within model + general factor between



lavaan syntax model 4a

```
> model4a <- '
    level: 1
      numeric    =~ wordlist + cards + matrices
      perception =~ figures + animals + occupats
    level: 2
      general    =~ wordlist + cards + matrices +
                         figures + animals + occupats
  '
> fit4a <- sem(model4a, data = FamIQData, cluster = "family",
               std.lv = TRUE, verbose = FALSE)
> summary(fit4a)

lavaan 0.6-7 ended normally after 45 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         31

  Number of observations                           399
  Number of clusters [family]                       60

Model Test User Model:

  Test statistic                                11.927
  Degrees of freedom                                17
  P-value (Chi-square)                           0.805
```

```
Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian


Level 1 [within]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  numeric =~
    wordlist          3.175    0.202   15.711    0.000
    cards             3.144    0.194   16.167    0.000
    matrices          3.054    0.199   15.349    0.000
  perception =~
    figures           3.095    0.204   15.146    0.000
    animals           3.188    0.194   16.438    0.000
    occupats          2.782    0.183   15.215    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  numeric ~~
    perception        0.382    0.057    6.740    0.000

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

```
   .wordlist         0.000
   .cards            0.000
   .matrices         0.000
   .figures          0.000
   .animals          0.000
   .occupats         0.000
    numeric          0.000
    perception       0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .wordlist          6.194    0.737    8.406    0.000
   .cards             5.403    0.692    7.804    0.000
   .matrices          6.417    0.714    8.992    0.000
   .figures           6.847    0.757    9.049    0.000
   .animals           4.881    0.696    7.009    0.000
   .occupats          5.324    0.603    8.823    0.000
    numeric           1.000
    perception        1.000


Level 2 [family]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  general =~
    wordlist          3.057    0.393    7.785    0.000
    cards             3.054    0.389    7.843    0.000
```

```
    matrices        2.632    0.381    6.904    0.000
    figures         2.806    0.398    7.048    0.000
    animals         3.204    0.383    8.371    0.000
    occupats        3.439    0.415    8.292    0.000

Intercepts:
                 Estimate  Std.Err  z-value  P(>|z|)
   .wordlist       29.891    0.468   63.847    0.000
   .cards          29.890    0.466   64.096    0.000
   .matrices       29.749    0.435   68.462    0.000
   .figures        30.044    0.458   65.536    0.000
   .animals        30.134    0.471   64.042    0.000
   .occupats       29.967    0.508   59.012    0.000
    general         0.000

Variances:
                 Estimate  Std.Err  z-value  P(>|z|)
   .wordlist        1.253    0.569    2.201    0.028
   .cards           1.323    0.586    2.258    0.024
   .matrices        1.935    0.669    2.891    0.004
   .figures         2.158    0.714    3.022    0.003
   .animals         0.656    0.487    1.347    0.178
   .occupats        1.581    0.624    2.536    0.011
    general         1.000
```

8.10 Example: two-level SEM

- we use an example from this book (Chapter 15):

  Hox, J.J., Moerbeek, M., & van de Schoot, R. (2010). *Multilevel analysis: Techniques and applications*. Routledge.

- based on a study by Schijf and Dronker (1991): they collected data from 1559 pupils (1382 after listwise deletion) in 58 schools

- pupil variables: father's occupational status (focc), father's education (feduc), mother's education (meduc), the result of the GALO school achievement test (galo), and the teacher's advice about secondary education (advice)

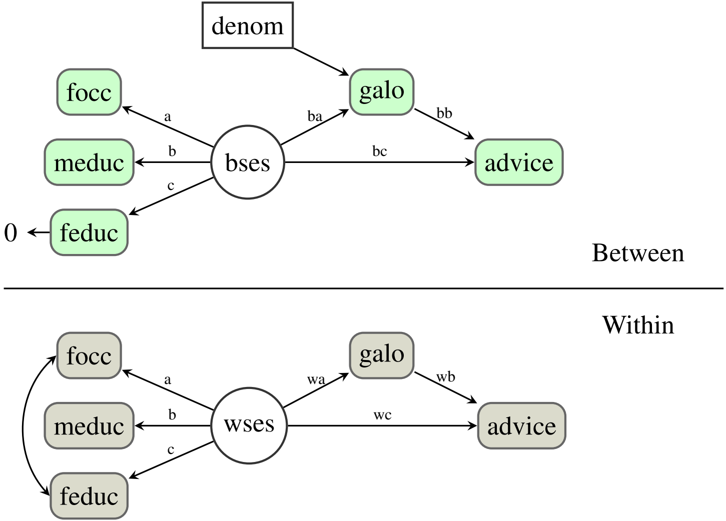
- at the school level, we have one variable: the school's denomination (denom) coded as 1=Protestant, 2=Nondenominational, 3=Catholic

- the main research question is whether the school's denomination affects the GALO score and (indirectly) the teacher's advice, after the other variables have been accounted for

modeling strategy

- a latent variable is constructed to reflect the socio-economic status (ses) using the variables focc, meduc and feduc as indicators
  - we will construct a configural latent variable for ses at the between level (using equality constraints for the loadings)
- preliminary analysis (using the pooled within-clusters covariance matrix only) revealed that a residual correlation is needed between the indicators focc and feduc at the within level
- in addtion, it was decided to fix the residual variance of `feduc` to zero at the between level
- a secondary question is whether the effect of ses on advice is direct or indirect
  - we label the various regression paths, and compute product terms to compute the indirect effect
  - both at the within and the between level

the model


denom
focc
galo
a
ba
bb
meduc
b
bses
bc
advice
c
0
feduc
Between
Within
focc
galo
a
wa
wb
meduc
b
wses
wc
advice
c
feduc

exploring the data

```
> Galo <- read.table("Galo.dat")
> names(Galo) <- c("school", "sex", "galo", "advice", "feduc", "meduc",
                   "focc", "denom")
> Galo[Galo == 999] <- NA
> Galo$denom1 <- ifelse(Galo$denom == 1, 1, 0)
> Galo$denom2 <- ifelse(Galo$denom == 2, 1, 0)
> summary(Galo)
```

```
     school           sex             galo           advice
 Min.   : 1.00   Min.   :1.000   Min.   : 53.0   Min.   :0.000
 1st Qu.:16.00   1st Qu.:1.000   1st Qu.: 94.0   1st Qu.:2.000
 Median :30.00   Median :2.000   Median :103.0   Median :2.000
 Mean   :29.87   Mean   :1.509   Mean   :102.3   Mean   :3.121
 3rd Qu.:43.00   3rd Qu.:2.000   3rd Qu.:111.0   3rd Qu.:4.000
 Max.   :58.00   Max.   :2.000   Max.   :143.0   Max.   :6.000
                                                 NA's   :7
     feduc           meduc            focc           denom
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
 1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:2.000
 Median :4.000   Median :2.000   Median :3.000   Median :2.000
 Mean   :4.002   Mean   :2.966   Mean   :3.336   Mean   :2.007
 3rd Qu.:6.000   3rd Qu.:5.000   3rd Qu.:5.000   3rd Qu.:2.000
 Max.   :9.000   Max.   :9.000   Max.   :6.000   Max.   :3.000
 NA's   :89      NA's   :61      NA's   :117
     denom1           denom2
 Min.   :0.0000   Min.   :0.0000
```

```
 1st Qu.:0.0000   1st Qu.:0.0000
 Median :0.0000   Median :1.0000
 Mean   :0.1501   Mean   :0.6928
 3rd Qu.:0.0000   3rd Qu.:1.0000
 Max.   :1.0000   Max.   :1.0000

> table(table(Galo$school))

10 12 13 14 19 20 21 22 23 24 25 26 27 28 29 30 32 33 34 35 36 37 42 46
 1  2  1  3  1  2  3  3  1  6  3  3  4  2  1  4  1  4  5  1  2  2  1  2
```

lavaan syntax

```
> model <- '
      level: within
          wses =~ a*focc + b*meduc + c*feduc
          # residual correlation
          focc ~~ feduc

          advice ~ wc*wses + wb*galo
          galo   ~ wa*wses

      level: between
          bses =~ a*focc + b*meduc + c*feduc
          feduc ~~ 0*feduc

          advice ~ bc*bses + bb*galo
          galo   ~ ba*bses + denom1 + denom2

      # defined parameters
      wi := wa * wb
      bi := ba * bb
  '
> fit <- sem(model, data = Galo, cluster = "school", std.lv = TRUE)
> summary(fit)

lavaan 0.6-7 ended normally after 108 iterations

  Estimator                                         ML
```

```
  Optimization method                              NLMINB
  Number of free parameters                            29
  Number of equality constraints                        3

                                                     Used       Total
  Number of observations                             1382        1559
  Number of clusters [school]                          58

Model Test User Model:

  Test statistic                                   26.221
  Degrees of freedom                                   19
  P-value (Chi-square)                              0.124

Parameter Estimates:

  Standard errors                                Standard
  Information                                    Observed
  Observed information based on                   Hessian


Level 1 [within]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  wses =~
    focc      (a)     0.748    0.038   19.558    0.000
    meduc     (b)     1.282    0.047   27.570    0.000
```

```
    feduc      (c)    1.674    0.057   29.205    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  advice ~
    wses      (wc)    0.119    0.027    4.489    0.000
    galo      (wb)    0.086    0.002   44.740    0.000
  galo ~
    wses      (wa)    4.200    0.371   11.325    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
 .focc ~~
   .feduc             0.257    0.086    2.986    0.003

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .focc              0.000
   .meduc             0.000
   .feduc             0.000
   .advice            0.000
   .galo              0.000
    wses              0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .focc              1.186    0.065   18.132    0.000
   .meduc             2.021    0.120   16.900    0.000
```

```
   .feduc             1.582    0.167    9.462    0.000
   .advice            0.574    0.022   25.512    0.000
   .galo            125.024    5.123   24.403    0.000
    wses              1.000


Level 2 [school]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  bses =~
    focc       (a)    0.748    0.038   19.558    0.000
    meduc      (b)    1.282    0.047   27.570    0.000
    feduc      (c)    1.674    0.057   29.205    0.000

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  advice ~
    bses      (bc)    0.274    0.069    3.958    0.000
    galo      (bb)    0.062    0.011    5.443    0.000
  galo ~
    bses      (ba)    5.121    0.591    8.672    0.000
    denom1           -5.153    1.602   -3.216    0.001
    denom2           -0.511    1.267   -0.403    0.687

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .focc              3.251    0.108   30.201    0.000
```

```
   .meduc            2.839    0.178   15.964    0.000
   .feduc            3.862    0.228   16.948    0.000
   .advice          -3.238    1.164   -2.782    0.005
   .galo           103.333    1.335   77.378    0.000
    bses             0.000

Variances:
                  Estimate  Std.Err  z-value  P(>|z|)
   .feduc            0.000
   .focc             0.032    0.016    2.010    0.044
   .meduc            0.021    0.025    0.844    0.398
   .advice           0.015    0.008    1.905    0.057
   .galo             5.746    2.057    2.793    0.005
    bses             1.000

Defined Parameters:
                  Estimate  Std.Err  z-value  P(>|z|)
    wi               0.360    0.032   11.146    0.000
    bi               0.317    0.068    4.698    0.000
```

9 Alternative ways to analyze multilevel data with SEM

- some alternative ways to analyze multilevel data with SEM:

  1. the 'wide data' approach: we arrange data in the wide format, and then use single-level SEM to analyze our model
  2. the 'survey' approach: we analyze the data (in long format) as if there where no clusters, but we use cluster-robust standard errors
  3. the two-stage approach: multilevel software (e.g., MLwiN) is used to estimate the (saturated) within and between covariance matrix; analysis by multigroup SEM (Goldstein, 1987)
  4. the pseudo-balanced approach: we pretend the data is balanced, and use a special estimator to fit a multigroup SEM (MUML)
  5. ...

why should you know about these alternatives?

- they may enhance your understanding of:
  - SEM
  - multilevel regression
  - multilevel SEM
  - the relationships between the different modeling frameworks
- depending on your data, model and research questions, they may be easier to set up, have less convergence problems, and the results may be easier to interpret and report
- in some cases, they may safe the day

9.1 The 'wide data' approach

- wonderful paper about this:

  Bauer, D.J. (2003). Estimating Multilevel Linear Models as Structural Equation Models. *Journal of Educational and Behavioral Statistics, 28*, 135–167.

- recent summary and extension to categorical data:

  Barendse, M.T., & Rosseel, Y. (2020). Multilevel Modeling in the 'Wide Format' Approach with Discrete Data: A Solution for Small Cluster Sizes. *Structural Equation Modeling: A Multidisciplinary Journal, 1–26.*

- first approach: using classic SEM to mimic multilevel regression models
  - the random intercepts and random slopes are represented by latent variables
  - the factor loadings of the random intercept are fixed to 1.0

  - the factor loadings of the random slope are fixed to the values of the predictor
  - only feasible if the predictor has a limited number of possible values (e.g. binary, or timepoint 1, 2, 3, or 4)
  - most importantly: only if the values for the predictor are the same for all units ('balanced design')
  - typical example: growth curve model
  - advantage: single-level analysis, model fit (although care is needed to specify the saturated model), flexible error structure, ...

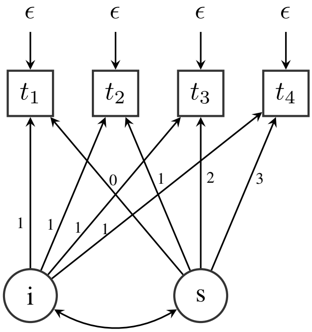
- second approach: calculate a model-implied covariance matrix (and mean vector) for each individual
  - needs special software (like OpenMx or Mplus)
  - predictor can be continuous, design does not need to be balanced

- because we are in the SEM context, we can extend these approaches to include latent variables, mediators, ...

- can be useful if:
  - the cluster sizes are (very) small
  - the number of variables (per unit) is relatively small
  - the data is (almost) balanced
  - the wide data still has many more rows (N) then columns (P)

example: a growth curve model with 4 time-points

- random intercept and random slope



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

R code: using SEM in wide format

```
> library(lavaan)
> head(Demo.growth[,c("t1","t2","t3","t4")], n = 4)

          t1        t2        t3        t4
1  1.7256454  2.142401  2.773172  2.515956
2 -1.9841595 -4.400603 -6.016556 -7.029618
3  0.3195183 -1.269117  1.560016  2.868530
4  0.7769485  3.531371  3.138211  5.363741


> model.slope <- '
    int   =~ 1*t1 + 1*t2 + 1*t3 + 1*t4
    slope =~ 0*t1 + 1*t2 + 2*t3 + 3*t4

    # intercepts (fixed effects)
    int   ~ 1
    slope ~ 1

    # random intercept, random slope
    int   ~~ int
    slope ~~ slope
    int   ~~ slope

    # force same variance for all (compound symmetry)
    t1 ~~ v1*t1
    t2 ~~ v1*t2
```

```
      t3 ~~ v1*t3
      t4 ~~ v1*t4
  '
> fit.slope <- lavaan(model.slope, data = Demo.growth)
> summary(fit.slope)

lavaan 0.6-7 ended normally after 24 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                          9
  Number of equality constraints                     3

  Number of observations                           400

Model Test User Model:

  Test statistic                                 9.678
  Degrees of freedom                                 8
  P-value (Chi-square)                           0.288

Parameter Estimates:

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

Latent Variables:
```

```
                    Estimate  Std.Err  z-value  P(>|z|)
  int =~
    t1                 1.000
    t2                 1.000
    t3                 1.000
    t4                 1.000
  slope =~
    t1                 0.000
    t2                 1.000
    t3                 2.000
    t4                 3.000

Covariances:
                    Estimate  Std.Err  z-value  P(>|z|)
  int ~~
    slope              0.627    0.069    9.129    0.000

Intercepts:
                    Estimate  Std.Err  z-value  P(>|z|)
    int                0.617    0.077    8.029    0.000
    slope              1.005    0.042   24.013    0.000
   .t1                 0.000
   .t2                 0.000
   .t3                 0.000
   .t4                 0.000

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

```
    int                1.928    0.169   11.439    0.000
    slope              0.576    0.050   11.540    0.000
   .t1          (v1)   0.622    0.031   20.000    0.000
   .t2          (v1)   0.622    0.031   20.000    0.000
   .t3          (v1)   0.622    0.031   20.000    0.000
   .t4          (v1)   0.622    0.031   20.000    0.000
```

R code: using lmer

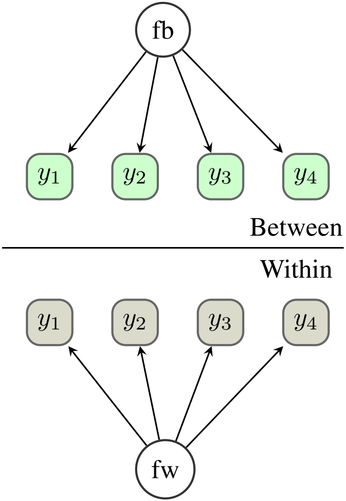
```
> # wide to long
> id    <- rep(1:400,  each = 4)
> score <- lav_matrix_vecr(Demo.growth[,1:4])
> time  <- rep(0:3, times = 400)
> growth.long <- data.frame(id = id, score = score, time = time)
> head(growth.long)

  id     score time
1  1  1.725645    0
2  1  2.142401    1
3  1  2.773172    2
4  1  2.515956    3
5  2 -1.984160    0
6  2 -4.400603    1
```

```
> library(lme4)
> fit.lmer <- lmer(score ~ 1 + time + (1 + time | id), data = growth.long,
                  REML = FALSE)
> summary(fit.lmer, correlation = FALSE)

Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: score ~ 1 + time + (1 + time | id)
   Data: growth.long
```

```
     AIC      BIC   logLik deviance df.resid
  5523.7   5556.0  -2755.9   5511.7     1594

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.62396 -0.51865 -0.00867  0.51881  2.83705

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 id       (Intercept) 1.9279   1.3885
          time        0.5765   0.7592   0.59
 Residual             0.6223   0.7889
Number of obs: 1600, groups:  id, 400

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.61716    0.07687   8.029
time         1.00519    0.04186  24.013
```

example 2: 1-factor model, cluster size = 3

- model in the multilevel SEM framework:



multilevel SEM syntax

```
> longData <- read.table("FCovRIcovWB.dat")
> names(longData) <- c("y1", "y2","y3", "y4", "x", "clus")
> model.long <- '
      level: 1
          fw =~ y1 + y2 + y3 + y4
      level: 2
          fb =~ y1 + y2 + y3 + y4
          y1 ~~ 0*y1
          y2 ~~ 0*y2
          y3 ~~ 0*y3
          y4 ~~ 0*y4
  '
> fit.long <- sem(model.long, data = longData, cluster = "clus",
                  fixed.x = FALSE)
> summary(fit.long)

lavaan 0.6-7 ended normally after 28 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         16

  Number of observations                          1200
  Number of clusters [clus]                        400

Model Test User Model:
```

```
  Test statistic                                 6.432
  Degrees of freedom                                 8
  P-value (Chi-square)                           0.599

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                1.039    0.059   17.552    0.000
    y3                0.942    0.052   18.136    0.000
    y4                0.985    0.058   17.024    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                0.491    0.027   17.880    0.000
   .y2                0.473    0.028   16.995    0.000
   .y3                0.481    0.026   18.443    0.000
   .y4                0.521    0.028   18.506    0.000
    fw                0.558    0.051   10.910    0.000


Level 2 [clus]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fb =~
    y1                1.000
    y2                0.886    0.098    9.087    0.000
    y3                0.977    0.095   10.328    0.000
    y4                0.871    0.098    8.847    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               -0.040    0.038   -1.045    0.296
   .y2               -0.049    0.037   -1.335    0.182
   .y3               -0.034    0.037   -0.906    0.365
   .y4               -0.034    0.037   -0.926    0.354
    fb                0.000
```

```
Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.000
   .y2                0.000
   .y3                0.000
   .y4                0.000
    fb                0.241    0.046    5.294    0.000
```

wide-format syntax

```
> wideData <- matrix(lav_matrix_vecr(longData[,1:5]), 400, 15, byrow = TRUE)
> wideData <- as.data.frame(wideData)
> names(wideData) <- paste(rep(c("y1","y2","y3","y4","x"), 3),
                           rep(1:3, each = 5), sep = ".")
> model.wide <- '
      # WITHIN #

      # within factors, common loadings, common (zero) means, common variance
      fw1 =~ 1*y1.1 + lw2*y2.1 + lw3*y3.1 + lw4*y4.1
      fw2 =~ 1*y1.2 + lw2*y2.2 + lw3*y3.2 + lw4*y4.2
      fw3 =~ 1*y1.3 + lw2*y2.3 + lw3*y3.3 + lw4*y4.3
      fw1 ~~ fvw*fw1
      fw2 ~~ fvw*fw2
      fw3 ~~ fvw*fw3

      # uncorrelated fw1, fw2, fw3
      fw1 ~~ 0*fw2 + 0*fw3; fw2 ~~ 0*fw3

      # within intercepts (fixed to zero)
      y1.1 + y2.1 + y3.1 + y4.1 ~ 0*1
      y1.2 + y2.2 + y3.2 + y4.2 ~ 0*1
      y1.3 + y2.3 + y3.3 + y4.3 ~ 0*1

      # common residual variances
      y1.1 ~~ rw1*y1.1; y1.2 ~~ rw1*y1.2; y1.3 ~~ rw1*y1.3
      y2.1 ~~ rw2*y2.1; y2.2 ~~ rw2*y2.2; y2.3 ~~ rw2*y2.3
```

```
    y3.1 ~~ rw3*y3.1; y3.2 ~~ rw3*y3.2; y3.3 ~~ rw3*y3.3
    y4.1 ~~ rw4*y4.1; y4.2 ~~ rw4*y4.2; y4.3 ~~ rw4*y4.3

    # BETWEEN #

    # between version of y1,y2,y3,y4
    by1 =~ 1*y1.1 + 1*y1.2 + 1*y1.3
    by2 =~ 1*y2.1 + 1*y2.2 + 1*y2.3
    by3 =~ 1*y3.1 + 1*y3.2 + 1*y3.3
    by4 =~ 1*y4.1 + 1*y4.2 + 1*y4.3

    # between intercepts
    by1 + by2 + by3 + by4 ~ 1

    # optional: zero residual variances
    by1 ~~ 0*by1; by2 ~~ 0*by2; by3 ~~ 0*by3; by4 ~~ 0*by4

    # between factor
    fb =~ by1 + by2 + by3 + by4

    # not correlated with the within lvs
    fb ~~ 0*fw1 + 0*fw2 + 0*fw3

  '
> fit.wide <- sem(model.wide, data = wideData, information = "observed")
> summary(fit.wide)

lavaan 0.6-7 ended normally after 26 iterations
```

```
  Estimator                                         ML
  Optimization method                           NLMINB
  Number of free parameters                         32
  Number of equality constraints                    16

  Number of observations                           400

Model Test User Model:

  Test statistic                                69.728
  Degrees of freedom                                74
  P-value (Chi-square)                           0.619

Parameter Estimates:

  Standard errors                             Standard
  Information                                 Observed
  Observed information based on                Hessian

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fw1 =~
    y1.1              1.000
    y2.1      (lw2)   1.039    0.059   17.552    0.000
    y3.1      (lw3)   0.942    0.052   18.136    0.000
    y4.1      (lw4)   0.985    0.058   17.024    0.000
  fw2 =~
```

```
    y1.2                1.000
    y2.2      (lw2)     1.039    0.059   17.552    0.000
    y3.2      (lw3)     0.942    0.052   18.136    0.000
    y4.2      (lw4)     0.985    0.058   17.024    0.000
  fw3 =~
    y1.3                1.000
    y2.3      (lw2)     1.039    0.059   17.552    0.000
    y3.3      (lw3)     0.942    0.052   18.136    0.000
    y4.3      (lw4)     0.985    0.058   17.024    0.000
  by1 =~
    y1.1                1.000
    y1.2                1.000
    y1.3                1.000
  by2 =~
    y2.1                1.000
    y2.2                1.000
    y2.3                1.000
  by3 =~
    y3.1                1.000
    y3.2                1.000
    y3.3                1.000
  by4 =~
    y4.1                1.000
    y4.2                1.000
    y4.3                1.000
  fb =~
    by1                 1.000
    by2                 0.886    0.098    9.087    0.000
```

```
    by3               0.977    0.095   10.328    0.000
    by4               0.871    0.098    8.847    0.000

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  fw1 ~~
    fw2               0.000
    fw3               0.000
  fw2 ~~
    fw3               0.000
  fw1 ~~
    fb                0.000
  fw2 ~~
    fb                0.000
  fw3 ~~
    fb                0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1.1              0.000
   .y2.1              0.000
   .y3.1              0.000
   .y4.1              0.000
   .y1.2              0.000
   .y2.2              0.000
   .y3.2              0.000
   .y4.2              0.000
   .y1.3              0.000
```

```
   .y2.3              0.000
   .y3.3              0.000
   .y4.3              0.000
   .by1              -0.040    0.038   -1.045    0.296
   .by2              -0.049    0.037   -1.335    0.182
   .by3              -0.034    0.037   -0.906    0.365
   .by4              -0.034    0.037   -0.926    0.354
    fw1               0.000
    fw2               0.000
    fw3               0.000
    fb                0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
    fw1      (fvw)    0.558    0.051   10.910    0.000
    fw2      (fvw)    0.558    0.051   10.910    0.000
    fw3      (fvw)    0.558    0.051   10.910    0.000
   .y1.1     (rw1)    0.491    0.027   17.880    0.000
   .y1.2     (rw1)    0.491    0.027   17.880    0.000
   .y1.3     (rw1)    0.491    0.027   17.880    0.000
   .y2.1     (rw2)    0.473    0.028   16.995    0.000
   .y2.2     (rw2)    0.473    0.028   16.995    0.000
   .y2.3     (rw2)    0.473    0.028   16.995    0.000
   .y3.1     (rw3)    0.481    0.026   18.443    0.000
   .y3.2     (rw3)    0.481    0.026   18.443    0.000
   .y3.3     (rw3)    0.481    0.026   18.443    0.000
   .y4.1     (rw4)    0.521    0.028   18.506    0.000
   .y4.2     (rw4)    0.521    0.028   18.506    0.000
```

```
   .y4.3      (rw4)     0.521    0.028   18.506    0.000
   .by1                 0.000
   .by2                 0.000
   .by3                 0.000
   .by4                 0.000
    fb                  0.241    0.046    5.294    0.000
```

(optional) wide-format syntax saturated model

```
> model.sat <- '
      # WITHIN #

      # common variances
      y1.1 ~~ vw1*y1.1; y1.2 ~~ vw1*y1.2; y1.3 ~~ vw1*y1.3
      y2.1 ~~ vw2*y2.1; y2.2 ~~ vw2*y2.2; y2.3 ~~ vw2*y2.3
      y3.1 ~~ vw3*y3.1; y3.2 ~~ vw3*y3.2; y3.3 ~~ vw3*y3.3
      y4.1 ~~ vw4*y4.1; y4.2 ~~ vw4*y4.2; y4.3 ~~ vw4*y4.3

      # common covariances
      y1.1 ~~ cw12*y2.1 + cw13*y3.1 + cw14*y4.1; y2.1 ~~ cw23*y3.1 + cw24*y4.1; y3
      y1.2 ~~ cw12*y2.2 + cw13*y3.2 + cw14*y4.2; y2.2 ~~ cw23*y3.2 + cw24*y4.2; y3
      y1.3 ~~ cw12*y2.3 + cw13*y3.3 + cw14*y4.3; y2.3 ~~ cw23*y3.3 + cw24*y4.3; y3

      # within means (fixed to zero)
      y1.1 + y2.1 + y3.1 + y4.1 ~ 0*1
      y1.2 + y2.2 + y3.2 + y4.2 ~ 0*1
      y1.3 + y2.3 + y3.3 + y4.3 ~ 0*1
      # BETWEEN #

      # between version of y1,y2,y3,y4
      by1 =~ 1*y1.1 + 1*y1.2 + 1*y1.3
      by2 =~ 1*y2.1 + 1*y2.2 + 1*y2.3
      by3 =~ 1*y3.1 + 1*y3.2 + 1*y3.3
      by4 =~ 1*y4.1 + 1*y4.2 + 1*y4.3
```

```
    # between intercepts
    by1 + by2 + by3 + by4 ~ 1

    # between variances
    by1 ~~ by1; by2 ~~ by2; by3 ~~ by3; by4 ~~ by4

    # between covariances
    by1 ~~ by2 + by3 + by4
    by2 ~~ by3 + by4
    by3 ~~ by4
  '
> fit.sat <- sem(model.sat, data = wideData)
> lavTestLRT(fit.sat, fit.wide)

Chi-Squared Difference Test

         Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit.sat  66 12565 12660 62.948
fit.wide 74 12555 12619 69.728     6.7799       8     0.5606
```

(Note: the small discrepancy between the value of this chi-square test statistic and the one obtained from the multilevel fit, is due to the fact that in the multilevel setting, the saturated model is computed using an EM algorithm, which uses slightly different stopping criteria.)

9.2 The ‘survey’ (design-based) approach

- literature:

  Oberski, D.L. (2014). lavaan.survey: An R package for complex survey analysis of structural equation models. *Journal of Statistical Software, 57*, 1–27.

  Stapleton, L.M., McNeish, D.M., & Yang, J.S. (2016). Multilevel and single-level models for measured and latent variables when data are clustered. *Educational Psychologist*, 51, 317–330.

- mostly used if all variables (and constructs) are at the within-level only (but we could include level-2 predictors too)
- we treat the clustering as a (sampling) nuisance
- less assumptions are needed compared to the multilevel approach
- standard errors are design-based (‘cluster-robust’ using a sandwich type estimator)

- allows for incorporation of clustering, stratification, unequal probability weights, finite population correction, and multiple imputation (see lavaan.survey package)

example with lavaan

```
> model <- ' # no levels!
      fw1 =~ y1 + y2 + y3
      fw2 =~ y4 + y5 + y6
  '
> fit.robust <- sem(model, data = Demo.twolevel, cluster = "cluster")
> summary(fit.robust, header = FALSE)

Parameter Estimates:

  Standard errors                              Robust.cluster
  Information                                        Observed
  Observed information based on                       Hessian

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fw1 =~
    y1                1.000
    y2                0.733    0.033   22.016    0.000
    y3                0.653    0.035   18.764    0.000
  fw2 =~
    y4                1.000
    y5                0.750    0.046   16.147    0.000
    y6                0.712    0.045   15.700    0.000

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

```
  fw1 ~~
    fw2               0.372    0.097    3.847    0.000

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.025    0.084    0.296    0.767
   .y2               -0.024    0.066   -0.369    0.712
   .y3               -0.024    0.059   -0.400    0.689
   .y4                0.064    0.089    0.717    0.473
   .y5                0.078    0.073    1.073    0.283
   .y6                0.012    0.075    0.164    0.870
    fw1               0.000
    fw2               0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                1.019    0.082   12.412    0.000
   .y2                1.205    0.051   23.779    0.000
   .y3                1.178    0.053   22.121    0.000
   .y4                0.995    0.068   14.552    0.000
   .y5                1.187    0.047   25.270    0.000
   .y6                1.134    0.047   23.929    0.000
    fw1               1.969    0.143   13.788    0.000
    fw2               1.388    0.167    8.316    0.000
```

10 Last slide

- be careful with a small number of clusters (may lead to biased results)

  McNeish, D.M., & Stapleton, L.M. (2016). The effect of small sample size on two-level model estimates: A review and illustration. *Educational Psychology Review, 28*, 295–314.

- topics not discussed in this workshop:
  - construct reliability in the multilevel setting
  - multilevel mediation
  - random slopes in a multilevel SEM
  - multilevel SEM with categorical outcomes
  - missing data